A COMPARISON OF TRANSFORMER AND LSTM ENCODER DECODER MODELS FOR ASR

Albert Zeyer[1,2], *Parnia Bahar*[1,2], *Kazuki Irie*[1], *Ralf Schlüter*[1], *Hermann Ney*[1,2]

[1]Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52074 Aachen, Germany,
[2]AppTek GmbH, 52062 Aachen, Germany

{zeyer, bahar, irie, schlueter, ney}@cs.rwth-aachen.de

ABSTRACT

We present competitive results using a Transformer encoder-decoder-attention model for end-to-end speech recognition needing less training time compared to a similarly performing LSTM model. We observe that the Transformer training is in general more stable compared to the LSTM, although it also seems to overfit more, and thus shows more problems with generalization. We also find that two initial LSTM layers in the Transformer encoder provide a much better positional encoding. Data-augmentation, a variant of SpecAugment, helps to improve both the Transformer by 33% and the LSTM by 15% relative. We analyze several pretraining and scheduling schemes, which is crucial for both the Transformer and the LSTM models. We improve our LSTM model by additional convolutional layers. We perform our experiments on LibriSpeech 1000h, Switchboard 300h and TED-LIUM-v2 200h, and we show state-of-the-art performance on TED-LIUM-v2 for attention based end-to-end models. We deliberately limit the training on LibriSpeech to 12.5 epochs of the training data for comparisons, to keep the results of practical interest, although we show that longer training time still improves more. We publish all the code and setups to run our experiments.

***Index Terms*—** Transformer, LSTM, attention, end-to-end ASR

1. INTRODUCTION

End-to-end automatic speech recognition (ASR) tries to simplify the modeling aspect, the training pipeline, and the decoding algorithm [1–8]. It also tries to reduce any assumptions being made about the data, and thus better performance can potentially be expected compared with the conventional system, the hybrid hidden Markov model (HMM) - neural network (NN) approach on phoneme level [9–12]. The encoder-decoder attention model [6, 13, 14] is one possible end-to-end model which we will focus on in this work. A recent development in end-to-end models has gained lots of interest, as the performance is improving and it is finally overtaking the conventional hybrid approach [15, 16] if they are trained on moderate amounts of training data.

Initial works on encoder-decoder models were usually based on long short-term memory (LSTM) [17] networks [18]. Later, convolutional networks have been used as well [19–21]. More recently, the Transformer model was introduced [22], which is based on self-attention [23–25]. The motivation for self-attention is two-fold: It allows for more direct information flow across the whole sequence, and thus also more direct gradient flow. Also, it allows for faster training, as most operations can be calculated in parallel for the standard cross-entropy loss. Since then, the Transformer model has become a very successful model in various applications [26, 27]. The Transformer as an auto-regressive language model has been suggested in [28–32]. The Transformer LM for ASR has been used in [21, 33]. More recent work suggested to use the Transformer directly as the acoustic model [34] or as an end-to-end ASR model [34–42].

We want to extend on that work, and provide a comparison between Transformer models and LSTM models for end-to-end ASR. We show that initial LSTM layers instead of the standard positional encoding greatly improved our Transformer performance. Also, we show the importance of data augmentation, especially for the Transformer model. A comparison between Transformer and LSTM has been done in [43–46]. Combining the components of these two architectures have also been studied [44] in which better results than both individual models have been obtained. A pretraining scheme like BERT as a contextual representation based on a large Transformer model [26] is commonly used in many applications. A similar pretraining scheme to the Transformer as we do has been done in [47].

We also extend on our LSTM baseline by adding initial convolutional layers, and a slightly more stable MLP attention variant, by reparameterization of the outer weights. Also, we explore a faster LSTM variant where the decoder LSTM is decoupled from the loop, and thus allows for faster training.

2. LSTM-BASED ASR BASELINE

Our LSTM baseline is based on [14, 48, 49], but Listen-Attend-and-Spell [6] and others are similar. Our LSTM baseline model consists of a stacked bidirectional LSTM encoder, with max-pooling in time operations optionally between the

978-1-7281-0306-8/19/$31.00 ©2019 IEEE

bidirectional LSTM (BLSTM) layers. For the input sequence x_1^T, we end up with the encoder state

$$h_1^{T'} = (\mathrm{BLSTM}_{\#\mathrm{enc}} \circ \cdots \circ \text{max-pool}_1 \circ \mathrm{BLSTM}_1)(x_1^T),$$

where $T' = T/\#\mathrm{red}.$ for the time reduction factor $\#\mathrm{red}$, and $\#\mathrm{enc}$ is the number of encoder layers, with $\#\mathrm{enc} \geq 2$. In our experiments, the first max-pooling has pool-size 3, and the second and last max-pooling has pool-size 2, i.e. we get a total time reduction factor $\#\mathrm{red} = 6$.

In the decoder, we use the **MLP attention** [5, 13], i.e. the attention energies $e_{i,t} \in \mathbb{R}$ for encoder time-step t and decoder step i are defined as

$$e_{i,t} = v^\top \tanh(W[s_i, h_t, \beta_{i,t}]),$$

where v is a trainable vector, W a trainable matrix, s_i the current decoder state, h_t the encoder state, and $\beta_{i,t}$ is the attention weight feedback. The attention weight feedback is a variant of the fertility score [50], which is defined as

$$\beta_{i,t} = \sigma(v_\beta^\top h_t) \cdot \sum_{k=1}^{i-1} \alpha_{k,t},$$

where v_β is a trainable vector. Then the attention weights are defined as

$$\alpha_i = \mathrm{softmax}_t(e_i)$$

and the attention context vector is given as

$$c_i = \sum_t \alpha_{i,t} h_t.$$

The decoder state is a recurrent function implemented as

$$s_i = \mathrm{LSTMCell}(s_{i-1}, y_{i-1}, c_{i-1})$$

and the final prediction probability for the output symbol y_i is given as

$$p(y_i | y_{i-1}, x_1^T) = \mathrm{softmax}(\mathrm{MLP}_{\mathrm{readout}}(s_i, y_{i-1}, c_i)).$$

In our case we use $\mathrm{MLP}_{\mathrm{readout}} = \mathrm{linear} \circ \mathrm{maxout} \circ \mathrm{linear}$.

For LibriSpeech and TED-LIUM, we use CTC as an additional loss on top the encoder [51], mostly to help the initial convergence, alongside with the pretraining, but also for regularization, but the CTC output is not used in decoding.

2.1. Pretraining

We follow a similar pretraining scheme as in [48], which has been shown to help convergence. Specifically, we start with a small encoder (small in depth and width, i.e. number of layers and hidden dimensions) and then grow it step by step (i.e. we add layer by layer, and increase the dimension sizes). Alongside the growth of the model, we also use learning rate warmup and scheduling of regularization such as dropout [52] and label smoothing [53], which we keep low or disabled initially and then later enable. Also, it can help to have a larger batch size (or use accumulated gradients) initially. Alongside this scheduling, some curriculum learning [54] can also be important. All of that is to make the convergence initially more stable and easier.

In our experiments, training does not converge at all if we do not use any pretraining. Although if we do it too much (i.e. too long), it also hurts performance, thus the pretraining phase needs a bit of tuning. Usually tuning the pretraining for some fast initial convergence is enough, and does not take much work and time, as you are looking at the initial training phase only.

2.2. Initial convolutional network

Inspired by [19], we experimented with adding convolutional layers as the initial encoder layers, directly on the features, before the BLSTM layers. In our experiments, it made the initial convergence somewhat more unstable (we needed to increase the pretraining time), but it can help the overall performance in some cases (see Table 2).

2.3. Stable MLP attention projection (ExpV)

We can reformulate the attention energies $e_{i,t}$ as

$$e_{i,t} = {v'}^\top \tanh(W'[\ldots]),$$

where $v' \geq 0$ element-wise. We simply set $v' = \mathrm{abs}(v)$, and for all negative elements in v, we multiply the corresponding row in W, such that we get W'. Thus, if we restrict all elements in v to be positive, we keep the same expressiveness. This inspired us to use

$$e_{i,t} = \exp(v'')^\top \tanh(W'[\ldots])$$

for any v''.

2.4. Decoupled decoder LSTM

For faster training time, and inspired by [40], and similar to the Transformer, we also investigate a variant, where the decoder LSTM does only depend on the ground truth, instead of the attention context. This allows for the direct use of very fast LSTM implementations. Besides, we don't use weight feedback and we use dot-attention instead of MLP-attention, such that we can parallelize everything else in the decoder (this is automatically being optimized by RETURNN) (see Table 4 for train speed). Interestingly, we observed that these models tend to overfit more, similar to the Transformer. If overfitting is not dominant, this model performs good, but worse than the LSTM baseline (see Table 2).

3. TRANSFORMER-BASED ASR MODEL

Our Transformer model is based on the original model [22] consisting of stacked layers. The encoder is composed of a multihead self-attention layer followed by a feedforward layer. We apply layer normalization [55] before each layer and dropout followed by a residual connection afterward. The decoder is constructed similar to the encoder with an additional multi-head attention layer to couple the representation of encoder states and decoder ones. As for the LSTM encoder, we also apply some time reduction in the encoder to adopt long speech sequences. As we have shown before that a stacked BLSTM with interleaved max-pooling in time works well, we utilize the same idea here, specifically 2 layers of BLSTM and max-pooling operations. As the LSTM is able to learn a positional encoding by itself, we discard the standard positional encoding from the encoder that initially proposed in [22] to preserve the order of a sequence. We also discard it from the decoder, as shown in [33] since it is not needed for the auto-regressive model. Therefore, we define

$$\begin{aligned}{h'}_1^{T'} = (&\mathrm{dropout}_{\mathrm{embed}} \circ \mathrm{linear}_{\mathrm{embed}} \circ \\ &\text{max-pool}_2 \circ \mathrm{BLSTM}_2 \circ \text{max-pool}_1 \circ \mathrm{BLSTM}_1)(x_1^T),\end{aligned}$$

where max-pool_1 has pool-size 3, and max-pool_2 has pool-size 2, i.e. $T' = T/\#\mathrm{red}$ with time reduction factor $\#\mathrm{red} = 6$. As stated, it follows the standard Transformer encoder, i.e.

$$\begin{aligned}h_1^{T'} = (&\text{layer-norm} \circ \\ &\mathrm{TrafoEncLayer}_{\#\mathrm{enc}} \circ \cdots \circ \mathrm{TrafoEncLayer}_1)({h'}_1^{T'}),\end{aligned}$$

where h_1^T is the final encoder output, #enc is the number of Transformer encoder layers, and each layer is defined as

$$\mathrm{TrafoEncLayer} = \mathrm{SelfAttBlock} \circ \mathrm{FFBlock},$$

where

$$\begin{aligned}&\mathrm{SelfAttBlock}(z) = \\ &\quad z + (\mathrm{dropout} \circ \mathrm{linear} \circ \text{self-attention} \circ \text{layer-norm})(z), \\ &\mathrm{FFBlock}(z) = \\ &\quad z + (\mathrm{dropout} \circ \mathrm{linear} \circ \mathrm{relu} \circ \mathrm{linear} \circ \text{layer-norm})(z).\end{aligned}$$

The decoder predicts the next output y_i as before, i.e. it defines

$$\begin{aligned}&p(y_i | y_1^{i-1}, h_1^{T'}(x_1^T)) = \\ &\quad (\mathrm{softmax} \circ \mathrm{linear} \circ \text{layer-norm} \circ \\ &\quad \mathrm{TrafoDecLayer}_{\#\mathrm{dec}}(h) \circ \cdots \circ \mathrm{TrafoDecLayer}_1(h) \circ \\ &\quad \mathrm{linear})(y_1^{i-1}),\end{aligned}$$

where $\#\mathrm{dec}$ is the number of decoder layers. Each decoder layer is defined as

$$\begin{aligned}&\mathrm{TrafoDecLayer} = \\ &\quad \mathrm{FFBlock} \circ \mathrm{EncAttBlock} \circ \mathrm{MaskedSelfAttBlock},\end{aligned}$$

where $\mathrm{FFBlock}$ is as before, $\mathrm{MaskedSelfAttBlock}$ is like $\mathrm{SelfAttBlock}$ but it used $\text{masked-self-attention}$, and

$$\begin{aligned}&\mathrm{EncAttBlock}(z) = \\ &\quad z + (\mathrm{dropout} \circ \mathrm{linear} \circ \mathrm{attention}(h) \circ \text{layer-norm})(z).\end{aligned}$$

All three types of attention we use, $\mathrm{attention}(h)$, self-attention and $\text{masked-self-attention}$ are simply dot-attention, and the detailed formula can be seen in [22], or in our configuration files. We also use dropout on the attention weights.

3.1. Pretraining

Following the experience with pretraining from the LSTM model, we apply a similar pretraining scheme for the Transformer. However, as we observed a more stable and faster initial convergence, we increased the pretrain construction speed. Also, as the decoder is deeper now, we let it grow, like the encoder. Specifically, we start with a single layer in the encoder and decoder each, and double the number of layers after every pretrain iteration. At the same time, we linearly increase the hidden dimensions. Completely disabling the pretraining does not converge in our experiments; having a slower pretraining scheme decreased the performance.

3.2. Training behaviour

Compared to the LSTM models, the Transformer model trains **faster**. Also, it usually trains **more stable**, i.e. the convergence is much faster initially, and it does also converge without CTC, in contrast to the LSTM. We also observed that the Transformer model tends to **overfit more** compared to the LSTM (compare Figure 1). Overall, we get similar performance with the Transformer compared to the LSTM (compare Table 2), if a moderate amount of training data provided.

The initial LSTM layers are quite important (compared to the standard positional encoding without LSTMs), for better convergence and also final performance. CTC also helps in the same way, but to a lesser extent (compare Figure 2). We also compare a different number of layers in the encoder and the decoder (see Table 1), and in general see that increased depth helps up to some limit [1].

4. DATA AUGMENTATION

We closely follow SpecAugment [16] as our data augmentation technique. We randomly apply masking in consecutive frames in the time axis, and consecutive dimensions in the feature axis. Our time warping was optional (and later also mostly disabled for performance reasons) and changed the speed uniformly on the whole sequence, on the raw audio level. This is done always on-the-fly, i.e. the model never sees the original data during training, and a different permuted variant in every step. We can confirm that it severely reduces overfitting. The effect varies though, depending on the model and on the dataset. In general, if a particular setup had much overfitting problems, it can help, to some degree. In some cases it completely solves the overfitting, in others it just reduces it. And there is a limit of how much data augmentation is still effective. This is first by the stability of the initial convergence, which can be solved though by some scheduling of the augmentation. However, even then, at some point, the performance just degraded by more augmentation. Still, in all cases, adding some data augmentation improved the setup (see Table 2, Figure 1, Figure 5).

5. LANGUAGE MODEL FUSION

We train Transformer language models [28–33] on the extra text-only data and the transcriptions. We apply the language model in decoding by shallow fusion [56,57]. We use a single scale parameter to weight the log probability of the language

[1] However, many of our experiments were conducted with the initially selected baseline architecture of 12 layers in both the encoder and the decoder.

model, which we tune on the development set WER.

Following [40], we study the end-of-sentence (EOS)-penalty in the context of language model shallow fusion. The EOS penalty allows preventing short hypotheses when a large beam size [58] is used in shallow fusion. The exact penalty variant we use is as follows: We introduce a factor γ (which we tune on the dev set) and we only allow emitting the EOS when its log probability is higher than γ times the log probability of the non-EOS word with the highest probability.

6. EXPERIMENTS

The experiments were performed on LibriSpeech (1000h) [59], Switchboard (300h) [60], and TED-LIUM release 2 (200h) [61]. All experiments were performed using RETURNN [62][2]. Language models for shallow fusion are selected based on their perplexity on the development set.

Table 1. LibriSpeech 1000h results. Comparing different encoder and decoder depths (number of layers).

Depth		**WER** [%]			
Enc.	**Dec.**	**dev**		**test**	
		clean	**other**	**clean**	**other**
10	10	4.49	11.69	4.56	11.97
12	6	4.59	11.73	4.65	12.13
18	6	4.52	11.74	4.88	12.10
12	12	4.55	11.70	4.71	11.90
12	16	4.57	11.59	4.63	11.93
14	14	4.44	11.61	4.59	**11.77**
16	16	4.42	**11.42**	**4.44**	12.13
18	18	**4.31**	11.58	4.54	12.20
20	20	4.34	11.43	4.60	11.83

6.1. LibriSpeech (1000h)

The LibriSpeech ASR used here is based on [14, 48, 49]. We limit the training time by a fixed number of epochs over the training dataset in all experiments, if not noted otherwise. Specifically, we train for exactly 12.5 epochs (∼12.5k hours of audio), which takes about 3-7 days on a single Nvidia 1080 Ti GPU (except for the 'LongTrain' experiments, which trained for 25 epochs). Note that this is much less compared to what the authors of SpecAugment [16] do for their best system. They train for over 600 epochs, which takes about 24 days on 32 TPUs [63][3]. We also observed that we can further improve our performance through longer training. All our results are collected in Table 2. As the total training time (in hours) can be of importance, we visualize our various experiments w.r.t. training time and WER in Figure 3, which shows also our trend during the past 1.5 years of tuning. While this highly depends on our experience, we also publish most of the relevant configs. In Figure 4, we compare Transformer based and LSTM based models, and while the LSTM has similar performance, in some cases slightly better performance, the Transformer often takes less training time. An important method is data augmentation, as can be seen in Figure 5.

Table 2. LibriSpeech 1000h results (12.5 epochs). Comparing Transformer to LSTM, and its decoupled decoder LSTM (DecLSTM) variant; comparing to other results in the literature; data augmentation most similar to SpecAugment; initial convolutional network (Conv); stable MLP attention projection (ExpV); LongTrain trains for twice as long (25 epochs).

Method	**LM**	**WER** [%]			
		dev		**test**	
		clean	**other**	**clean**	**other**
HMM					
LSTM [49]	Trafo	1.9	4.5	2.3	5.0
Attention					
CNN/LSTM [40]	CNN	3.01	8.86	3.28	9.84
LSTM, SpecAug [16]	none			2.8	6.8
+ LM [16]	LSTM			2.5	5.8
Transformer	none	5.74	16.43	5.84	17.21
+ SpecAug		4.31	10.98	4.49	11.50
+ LongTrain		**3.77**	**10.03**	**4.02**	**10.31**
+ LM-EOS	Trafo	**2.51**	**6.66**	**2.81**	**7.42**
LSTM	none	4.45	13.32	4.55	13.83
+ SpecAug		4.05	11.19	4.19	11.82
+ Conv		3.93	10.72	4.08	11.27
+ ExpV		4.12	10.90	4.36	11.30
+ LongTrain		**3.64**	**9.93**	**3.94**	**10.30**
+ LM	Trafo	2.72	6.89	**2.98**	7.75
+ LM-EOS	Trafo	**2.56**	**6.81**	3.03	**7.46**
+ LM-EOS	LSTM	2.77	7.50	3.10	8.19
DecLSTM	none	4.85	14.23	4.90	15.10
+ SpecAug		4.37	11.09	4.57	13.20

For our Transformer experiments, we use a hidden dimension of 512 except for the feedforward sublayer of size 2048. We set the number of heads to 8 and stack 12 layers both in the encoder and the decoder. We use dropout and label smoothing of 0.1. Similar to the LSTM models, we use the Adam optimizer [64] with a variant of Newbob learning rate scheduling. A warmup period with increasing learning rate is used.

For shallow fusion experiments, we train a 24-layer Transformer language model with a feedforward dimension of 4096 and 1024 self-attention total dimension with 8 heads on the 800M-word text only data. We carry out shallow fusion with both LSTM and Transformer attention baseline models. We first carry out shallow fusion for the LSTM baseline using 12, 32, and 64 beam sizes, where 32 gave the best development WERs. While we find the EOS penalty (LM-EOS) to give some improvements for the LSTM baseline (Table 2) with a beam size of 64 and the penalty of $\gamma = 1.0$, no improvement

[2] https://github.com/rwth-i6/returnn

[3] Personal communication with the authors of [16].

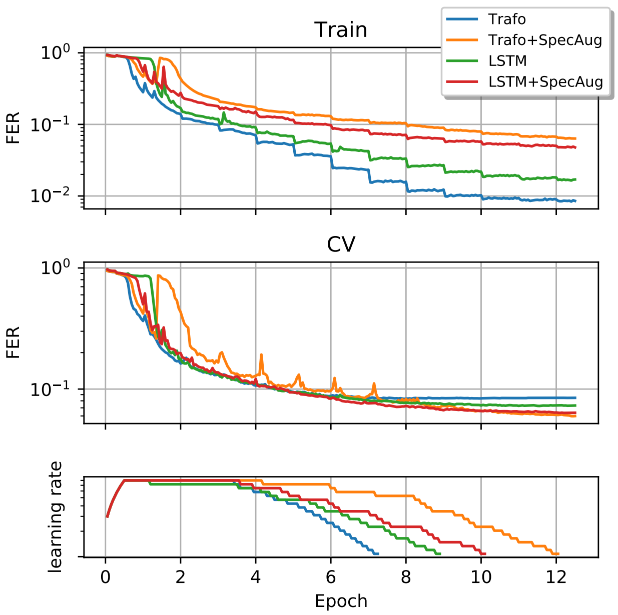

Fig. 1. The plot shows the frame-error-rate (FER) on the train and cross-validation (CV) from LibriSpeech, and the learning rate, all on log-scale, corresponding to Table 2. The learning rate scheduling is dependent on the relative change of the CV score.

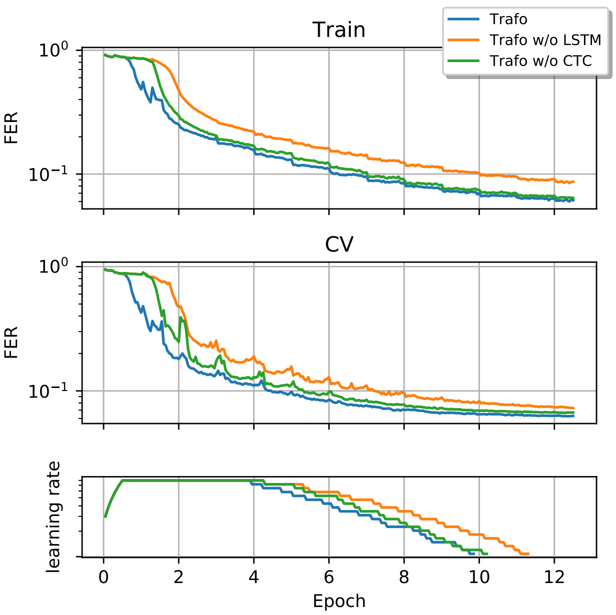

Fig. 2. The plot shows the frame-error-rate (FER) on the train and cross-validation (CV) from LibriSpeech, and the learning rate, all on log-scale. The baseline Transformer is with LSTM and with CTC and gets 11.71% WER on dev-other. Transformer w/o CTC gets 12.46% WER on dev-other.

was obtained for the Transformer baseline.

Table 3. Switchboard 300h results.

Method	**LM**	**WER** [%]		
		Hub5'00		**Hub5'01**
		SWB	**CH**	
HMM				
LSTM [14]	LSTM	8.3	17.3	
Attention				
Transformer [39]	none	10.4	18.6	
LSTM, SpecAug [16]	none	7.2	14.6	
+ LM [16]	LSTM	6.8	14.1	
Transformer	none	16.0	30.5	22.4
+ SpecAug		11.2	22.7	16.3
+ 36Enc		10.6	22.3	15.5
LSTM	none	11.9	23.7	17.7
+ SpecAug		**9.9**	21.5	**14.7**
+ ExpV		10.4	21.4	15.1
+ Conv		10.1	**20.6**	**14.7**
+ LM-EOS	Trafo	**9.3**	**20.3**	**14.2**

6.2. Switchboard (300h)

Our setup is based on [48]. We train for 33 epochs over the training data, which takes 2-4 days on a single 1080 GPU. We have started our Transformer experiments similar to the LibriSpeech configuration with a stack of 12 layers, however, we increase the dropout to 0.2, and apply it on various parts of the model, also embedding layer. Furthermore, we decrease the feedforward hidden size to 1024. The batch size and gradient accumulation are set such that we update every 25k-30k subwords. We also vary the number of encoder layers and dropout ratio and report the results in Table 3. In all the Transformer's configs, we utilize the CTC loss as we observed that the model converges faster, nevertheless, it converges without CTC. For shallow fusion, we selected a 30-layer Transformer language model with a feedforward hidden size of 2048 and a self-attention dimension of 512 with 8 heads. The introduction of the EOS penalty (LM-EOS) gave marginal improvements with a beam size of 32. We note that the improvements from the language model in this task with smaller LM training data (26M words) is rather limited compared with what we obtain for LibriSpeech.

6.3. TED-LIUM release 2 (200h)

The TED-LIUM setup is very much based on the LibriSpeech setup. We note that it took only 2 days to prepare the whole training and recognition pipeline, which is one of the advantages of such an end-to-end system, and also shows that the LibriSpeech baseline is stable across datasets. We publish the whole setup with all scripts for the whole preparation, training and recognition pipeline.[4] We train for 37 epochs over the training data, which takes 1-2 days on a single 1080

[4] https://github.com/rwth-i6/returnn-experiments/tree/master/2019-asr-e2e-trafo-vs-lstm

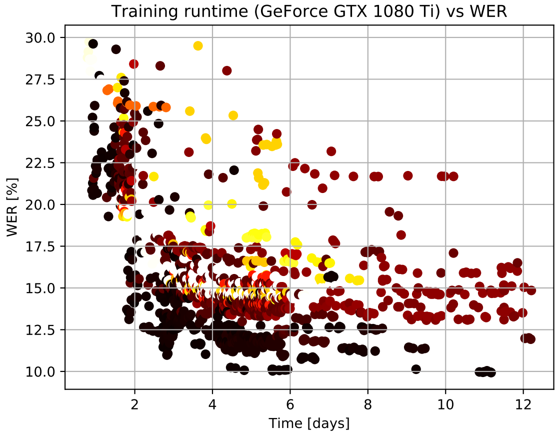

Fig. 3. The plot shows the WER and training runtime on a single Nvidia 1080 GPU of most of our LibriSpeech experiments, starting from Feb 2018 until Jun 2019 (date taken from our Git history). The darker the color, the more recent the experiment.

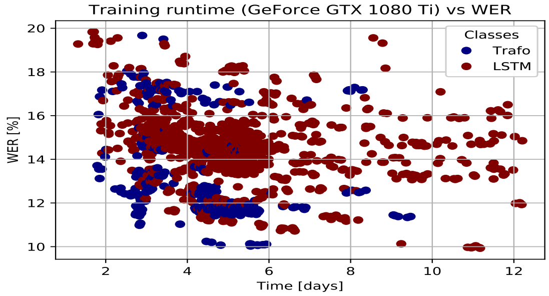

Fig. 4. Most of our LibriSpeech experiments, like Figure 3, showing Transformer vs. LSTM.

GPU. Experimental results are shown in Table 4. The Transformer model has been trained using data augmentation similar to those in the Switchboard setup. The decoupled decoder LSTM experiment is based on the LibriSpeech setup. We observe very high overfitting for this model, which can maybe be compensated by more regularization or data augmentation. For shallow fusion, we train a 1K-BPE level 30-layer Transformer with a feedforward dimension of 4096, total self attention dimension of 768 with 12 heads, on the 270M-word text only data provided with TED-LIUM release 2. Again, the use of EOS-penalty with a larger beam size of 32 gives slight improvements over the baseline shallow fusion with a beam size of 12 (further increasing the beam size to 64 did not result in any improvement).

7. CONCLUSION

We have shown competitive results with Transformer models for end-to-end ASR, which can be trained faster and more stable than LSTMs. Overfitting and generalization might be a problem, though, which can partly be overcome with data-augmentation. Data-augmentation, in general, seemed to be extremely helpful, especially for the Transformer models whose overfitting issue is much more severe than those of LSTM. Moreover, pretraining along with time reduction leads to faster convergence, and higher performance boost. The CTC auxiliary loss helps training in both cases.

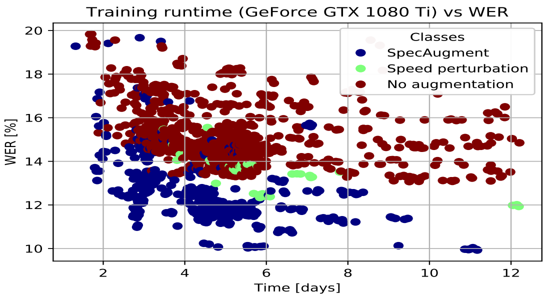

Fig. 5. Most of our LibriSpeech experiments, like Figure 3, showing effect of data augmentation.

Table 4. TED-LIUM-v2 results. [1] is a Transformer with an LSTM on top. [2] is a Transformer with interleaved LSTM. The training speed is with a single Nvidia 1080 Ti GPU. We use our own native CUDA LSTM implementation. All our models use a variant of SpecAugment.

Method	**LM**	**WER** [%]		**Train speed**
		dev	**test**	[char/sec]
HMM				
LSTM [65]	RNN	7.1	7.7	
Attention				
LSTM [35]	none	14.6	14.7	1.1k
Transformer[1] [35]	none	16.4	17.5	2.4k
Transformer[2] [35]	none	15.3	16.7	1.5k
Transformer	none	14.7	12.5	4.2k
+ 36Enc		13.9	12.0	2.2k
LSTM	none	12.4	10.8	3.3k
+ Conv		12.6	10.8	2.9k
+ ExpV		**11.7**	**10.5**	3.1k
+ LM	Trafo	10.5	9.0	-
+ LM-EOS	Trafo	**10.3**	**8.8**	-
DecLSTM	none	18.3	16.0	3.9k

8. ACKNOWLEDGEMENT

This work has received funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation programme (grant agreement No 694537, project "SEQCLAS") and from a Google Focused Award. The work reflects only the authors' views and none of the funding parties is responsible for any use that may be made of the information it contains.

9. REFERENCES

[1] Dario Amodei, Rishita Anubhai, Eric Battenberg, Carl Case, Jared Casper, Bryan Catanzaro, Jingdong Chen,

Mike Chrzanowski, Adam Coates, Greg Diamos, et al., "Deep speech 2: End-to-end speech recognition in english and mandarin," *arXiv preprint arXiv:1512.02595*, 2015.

[2] Hagen Soltau, Hank Liao, and Haşim Sak, "Neural speech recognizer: Acoustic-to-word LSTM model for large vocabulary speech recognition," in *Proc. Interspeech*, 2017, pp. 3707–3711.

[3] Kartik Audhkhasi, Bhuvana Ramabhadran, George Saon, Michael Picheny, and David Nahamoo, "Direct acoustics-to-word models for english conversational speech recognition," in *Proc. Interspeech*, 2017, pp. 959–963.

[4] Ronan Collobert, Christian Puhrsch, and Gabriel Synnaeve, "Wav2letter: an end-to-end convnet-based speech recognition system," *arXiv preprint arXiv:1609.03193*, 2016.

[5] Jan Chorowski, Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: first results," *arXiv preprint arXiv:1412.1602*, 2014.

[6] William Chan, Navdeep Jaitly, Quoc V. Le, and Oriol Vinyals, "Listen, attend and spell: A neural network for large vocabulary conversational speech recognition," in *ICASSP*, 2016.

[7] Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, Anuroop Sriram, and Zhenyao Zhu, "Exploring neural transducers for end-to-end speech recognition," in *ASRU*, Okinawa, Japan, Dec. 2017, pp. 206–213.

[8] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *ICASSP*, 2016, pp. 4945–4949.

[9] Hervé Bourlard and Nelson Morgan, *Connectionist speech recognition: a hybrid approach*, vol. 247, Springer, 1994.

[10] Anthony J Robinson, "An application of recurrent nets to phone probability estimation," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298–305, 1994.

[11] Hasim Sak, Andrew W. Senior, and Françoise Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Interspeech*, Singapore, Sept. 2014, pp. 338–342.

[12] Albert Zeyer, Patrick Doetsch, Paul Voigtlaender, Ralf Schlüter, and Hermann Ney, "A comprehensive study of deep bidirectional lstm rnns for acoustic modeling in speech recognition," in *ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 2462–2466.

[13] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *Proc. ICLR*, 2015.

[14] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Improved training of end-to-end attention models for speech recognition," in *Interspeech*, Hyderabad, India, Sept. 2018.

[15] Chung-Cheng Chiu, Tara N Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J Weiss, Kanishka Rao, Ekaterina Gonina, et al., "State-of-the-art speech recognition with sequence-to-sequence models," in *ICASSP*, 2018, pp. 4774–4778.

[16] Daniel S Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Interspeech*, Graz, Austria, Sept. 2019, pp. 2613–2617.

[17] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "Learning phrase representations using RNN encoder-decoder for statistical machine translation," *EMNLP*, 2014.

[19] Yu Zhang, William Chan, and Navdeep Jaitly, "Very deep convolutional networks for end-to-end speech recognition," in *ICASSP*, New Orleans, LA, USA, Mar. 2017, pp. 4845–4849.

[20] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin, "Convolutional sequence to sequence learning," in *ICML*. JMLR. org, 2017, pp. 1243–1252, arXiv:1705.03122.

[21] Jason Li, Vitaly Lavrukhin, Boris Ginsburg, Ryan Leary, Oleksii Kuchaiev, Jonathan M Cohen, Huyen Nguyen, and Ravi Teja Gadde, "Jasper: An end-to-end convolutional neural acoustic model," *arXiv preprint arXiv:1904.03288*, 2019.

[22] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin, "Attention is all you need," in *NIPS*, 2017, pp. 6000–6010.

[23] Jianpeng Cheng, Li Dong, and Mirella Lapata, "Long short-term memory-networks for machine reading," in *EMNLP*, Austin, TX, USA, Nov. 2016, pp. 551–561.

[24] Zhouhan Lin, Minwei Feng, Cicero Nogueira dos Santos, Mo Yu, Bing Xiang, Bowen Zhou, and Yoshua Bengio, "A structured self-attentive sentence embedding," *ICLR*, Apr. 2017.

[25] Ankur P. Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit, "A decomposable attention model for natural language inference," in *EMNLP*, Austin, TX, USA, Nov. 2016, pp. 2249–2255.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, "BERT: pre-training of deep bidirectional Transformers for language understanding," in *Proc. NAACL-HLT*, Minneapolis, MN, USA, June 2019, pp. 4171–4186.

[27] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le, "XLNet: Generalized autoregressive pretraining for language understanding," *CoRR*, vol. abs/1906.08237, 2019.

[28] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever, "Improving language understanding by generative pre-training," *OpenAI Blog*, 2018.

[29] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever, "Language models are unsupervised multitask learners," *OpenAI Blog*, 2019.

[30] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *ACL*, Florence, Italy, July 2019.

[31] Alexei Baevski and Michael Auli, "Adaptive input representations for neural language modeling," in *ICLR*, New Orleans, LA, USA, May 2019.

[32] Rami Al-Rfou, Dokook Choe, Noah Constant, Mandy Guo, and Llion Jones, "Character-level language modeling with deeper self-attention," in *AAAI Conf. on AI.*, Honolulu, HI, USA, Jan. 2019.

[33] Kazuki Irie, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "Language modeling with deep Transformers," in *Interspeech*, Graz, Austria, Sept. 2019, pp. 3905–3909.

[34] Daniel Povey, Hossein Hadian, Pegah Ghahremani, Ke Li, and Sanjeev Khudanpur, "A time-restricted self-attention layer for ASR," in *ICASSP*, 2018, pp. 5874–5878.

[35] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel, "Self-attentional acoustic models," in *Proc. Interspeech*, 2018, pp. 3723–3727.

[36] Linhao Dong, Shuang Xu, and Bo Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *ICASSP*, 2018, pp. 5884–5888.

[37] Julian Salazar, Katrin Kirchhoff, and Zhiheng Huang, "Self-attention networks for connectionist temporal classification in speech recognition," in *Proc. ICASSP*, Brighton, UK, May 2019, pp. 7115–7119.

[38] Abdelrahman Mohamed, Dmytro Okhonko, and Luke Zettlemoyer, "Transformers with convolutional context for ASR," *arXiv preprint arXiv:1904.11660*, 2019.

[39] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Muller, and Alex Waibel, "Very deep self-attention networks for end-to-end speech recognition," in *Interspeech*, Graz, Austria, Sept. 2019, pp. 66–70.

[40] Awni Hannun, Ann Lee, Qiantong Xu, and Ronan Collobert, "Sequence-to-sequence speech recognition with time-depth separable convolutions," in *Interspeech*, Graz, Austria, Sept. 2019, pp. 3785–3789.

[41] Jie Li, Xiaorui Wang, Yan Li, et al., "The speechtransformer for large-scale mandarin chinese speech recognition," in *ICASSP*. IEEE, 2019, pp. 7095–7099.

[42] Linhao Dong, Feng Wang, and Bo Xu, "Self-attention aligner: A latency-control end-to-end model for asr using self-attention network and chunk-hopping," *arXiv preprint arXiv:1902.06450*, 2019.

[43] Surafel M Lakew, Mauro Cettolo, and Marcello Federico, "A comparison of Transformer and recurrent neural networks on multilingual neural machine translation," in *International Conference on Computational Linguistics*, 2018.

[44] Mia Xu Chen, Orhan Firat, Ankur Bapna, Melvin Johnson, Wolfgang Macherey, George Foster, Llion Jones, Mike Schuster, Noam Shazeer, Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Zhifeng Chen, Yonghui Wu, and Macduff Hughes, "The best of both worlds: Combining recent advances in neural machine translation," in *Proc. ACL*, Melbourne, Australia, July 2018, pp. 76–86.

[45] Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al., "Gmail smart compose: Real-time assisted writing," *arXiv preprint arXiv:1906.00080*, 2019.

[46] Shigeki Karita, Nanxin Chen, Tomoki Hayashi, Takaaki Hori, Hirofumi Inaguma, Ziyan Jiang, Masao Someki, Nelson Enrique Yalta Soplin, Ryuichi Yamamoto, Xiaofei Wang, et al., "A comparative study on Transformer vs RNN in speech applications," in *ASRU*, 2019.

[47] Linyuan Gong, Di He, Zhuohan Li, Tao Qin, Liwei Wang, and Tieyan Liu, "Efficient training of bert by progressively stacking," in *ICML*, 2019, pp. 2337–2346.

[48] Albert Zeyer, André Merboldt, Ralf Schlüter, and Hermann Ney, "A comprehensive analysis on attention models," in *IRASL Workshop, NeurIPS*, Montreal, Canada, Dec. 2018.

[49] Christoph Lüscher, Eugen Beck, Kazuki Irie, Markus Kitza, Wilfried Michel, Albert Zeyer, Ralf Schlüter, and Hermann Ney, "RWTH ASR systems for librispeech: Hybrid vs attention," in *Interspeech*, Graz, Austria, Sept. 2019, pp. 231–235.

[50] Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li, "Modeling coverage for neural machine translation," in *ACL*, 2016.

[51] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William Chan, "Advances in joint CTC-attention based end-to-end speech recognition with a deep CNN encoder and RNN-LM," in *Interspeech*, 2017.

[52] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Mach. Learn. Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[53] Gabriel Pereyra, George Tucker, Jan Chorowski, Lukasz Kaiser, and Geoffrey E. Hinton, "Regularizing neural networks by penalizing confident output distributions," *CoRR*, vol. abs/1701.06548, 2017.

[54] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston, "Curriculum learning," in *Proc. ICML*, Montreal, Canada, June 2009, pp. 41–48.

[55] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016, Version 1.

[56] Çağlar Gülçehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loïc Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "On using monolingual corpora in neural machine translation," *Computer Speech & Language*, vol. 45, pp. 137–148, Sept. 2017.

[57] Shubham Toshniwal, Anjuli Kannan, Chung-Cheng Chiu, Yonghui Wu, Tara N Sainath, and Karen Livescu, "A comparison of techniques for language model integration in encoder-decoder speech recognition," in *Proc. SLT*, Athens, Greece, Dec. 2018.

[58] Kenton Murray and David Chiang, "Correcting length bias in neural machine translation," in *Proc. WMT*, Belgium, Brussels, Oct. 2018, pp. 212–223.

[59] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *ICASSP*. IEEE, 2015, pp. 5206–5210.

[60] John J Godfrey, Edward C Holliman, and Jane McDaniel, "Switchboard: Telephone speech corpus for research and development," in *Proc. ICASSP*, San Francisco, CA, USA, Mar. 1992, vol. 1, pp. 517–520.

[61] Anthony Rousseau, Paul Deléglise, and Yannick Estève, "Enhancing the TED-LIUM corpus with selected data for language modeling and more TED talks," in *LREC*, 2014, pp. 3935–3939.

[62] Albert Zeyer, Tamer Alkhouli, and Hermann Ney, "RETURNN as a generic flexible neural toolkit with application to translation and speech recognition," in *ACL*, Melbourne, Australia, July 2018.

[63] Norman P Jouppi, Cliff Young, et al., "In-datacenter performance analysis of a tensor processing unit," in *Proc. ISCA*, Toronto, Canada, June 2017, pp. 1–12.

[64] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014, Version 9.

[65] Kyu J Han, Akshay Chandrashekaran, Jungsuk Kim, and Ian Lane, "The CAPIO 2017 conversational speech recognition system," *arXiv preprint arXiv:1801.00059*, 2018.