Interspeech 2022
18-22 September 2022, Incheon, Korea

Efficient Training of Neural Transducer for Speech Recognition

Wei Zhou, Wilfried Michel, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52074 Aachen, Germany
AppTek GmbH, 52062 Aachen, Germany
{zhou,michel,schlueter,ney}@cs.rwth-aachen.de

Abstract

As one of the most popular sequence-to-sequence modeling approaches for speech recognition, the RNN-Transducer has achieved evolving performance with more and more sophisticated neural network models of growing size and increasing training epochs. While strong computation resources seem to be the prerequisite of training superior models, we try to overcome it by carefully designing a more efficient training pipeline. In this work, we propose an efficient 3-stage progressive training pipeline to build highly-performing neural transducer models from scratch with very limited computation resources in a reasonable short time period. The effectiveness of each stage is experimentally verified on both Librispeech and Switchboard corpora. The proposed pipeline is able to train transducer models approaching state-of-the-art performance with a single GPU in just 2-3 weeks. Our best conformer transducer achieves 4.1% WER on Librispeech test-other with only 35 epochs of training.

Index Terms: speech recognition, neural transducer, efficient training

1. Introduction

Recently, sequence-to-sequence modeling becomes the major trend of automatic speech recognition (ASR). Among other popular approaches such as connectionist temporal classification (CTC) [1] and attention-based encoder-decoder (AED) models [2, 3], the recurrent neural network transducer (RNN-T) [4] receives the most interest due to its good performance and streaming nature.

The RNN-T model definition allows a direct from-scratch training by summing over all alignment paths matching the output sequence. This simplicity usually comes at the cost of high memory and computation requirement. Meanwhile, the performance of transducer models has largely evolved with more and more sophisticated neural network (NN) architectures [5, 6, 7]. The best performing systems usually adopt a large NN trained for many epochs. All of these seem to indicate that strong computation resources have become the prerequisite of training highly-performing transducer models in a reasonable time. Although this can be true to some extent, we try to overcome it by carefully designing a more efficient training pipeline.

More specifically, we propose a fast 3-stage progressive training pipeline for neural transducer models in this work. We provide detailed recipes and design principles of each stage, which largely reduce the time and computation costs. Experiments on both Librispeech (LBS) [8] and Switchboard (SWB) [9] corpora show that our proposed pipeline is able to train transducer models approaching state-of-the-art (SOTA) performance from scratch with a single GPU in just 2-3 weeks.

2. Model

In this work, we mainly focus on the strictly monotonic version of RNN-T [10] which closely matches the nature of speech as desired for many real-time applications, although most of the proposed approaches can also work for standard RNN-T. Let X denote the acoustic feature sequence of a speech utterance and $h_1^T = f^{\text{enc}}(X)$ denote the encoder output, which transforms the input into high-level representations. Let a_1^S denote the output label sequence of length $S \leq T$, whose sequence posterior is defined as:

$$\begin{aligned} P_{\text{RNNT}}(a_1^S|X) &= \sum_{y_1^T:\mathcal{B}^{-1}(a_1^S)} P_{\text{RNNT}}(y_1^T|h_1^T) \\ &= \sum_{y_1^T:\mathcal{B}^{-1}(a_1^S)} \prod_{t=1}^{T} P_{\text{RNNT}}(y_t|\mathcal{B}(y_1^{t-1}), h_1^T) \end{aligned} \quad (1)$$

Here y_1^T is the blank ϵ-augmented alignment sequence, which is uniquely mapped to a_1^S via the collapsing function $\mathcal{B}$ to remove all blanks. A limited context dependency [11, 12, 13] can also be introduced to further simplify the model in Eq. (1). This can be better expressed from a lattice representation of the transducer label topology [4, 10]. By denoting y_1^{t-1} as a path reaching a node $(t-1, s-1)$, we have:

$$P_{\text{RNNT}}(y_t|\mathcal{B}(y_1^{t-1}), h_1^T) = P_{\text{RNNT}}(y_t|a_1^{s-1}, h_t) = P_{\text{RNNT}}(y_t|a_{s-k}^{s-1}, h_t)$$

where k is the label context size, and y_1^t reaches $(t, s-1)$ if $y_t = \epsilon$ or (t, s) otherwise.

For decoding using the RNN-T model with an external language model (LM), a general formulation of the maximum a posteriori (MAP) decision rule can be given as:

$$X \rightarrow \widehat{\mathcal{W}(a_1^S)} = \arg\max_{\mathcal{W}(a_1^S,S)} \; P_{\text{LM}}^{\lambda_1}(\mathcal{W}(a_1^S)) \cdot \frac{P_{\text{RNNT}}(a_1^S|X)}{P_{\text{RNNT-ILM}}^{\lambda_2}(a_1^S)} \quad (2)$$

where $P_{\text{RNNT-ILM}}$ is the internal LM (ILM) / sequence prior of P_{RNNT}. Here λ_1 and λ_2 are scales applied in common practice, where $\lambda_2 = 0$ leads to the simple shallow fusion (SF) approach [14]. The additional function $\mathcal{W}$ is introduced for a more general coverage of different label choices for a. More specifically, $\mathcal{W}$ is the lexical mapping or identity function when a represents phonemes or (sub)words, correspondingly.

3. Training Pipeline & Recipes

In a standard pipeline, the encoder of transducer NN is usually pretrained with the CTC [1] objective, which can be time consuming when the encoder size grows large. The complete transducer NN is then trained with the full-sum (FS) loss:

$$\mathcal{L}_{\text{FS}} = -\log P_{\text{RNNT}}(a_1^S|X) \quad (3)$$

which is usually very time consuming w.r.t. computation cost and convergence to optimal performance. It also has a high memory consumption due to the FS over all alignment paths.

To further improve the FS-trained transducer model, the minimum Bayes risk (MBR) sequence discriminative training is commonly applied due to its consistency with the ASR evaluation metric. For simplicity, we denote a_1^S as $\boldsymbol{a}$ and omit the variable length S for the MBR criterion:

10.21437/Interspeech.2022-829

$$\mathcal{L}_{\text{MBR}} = \sum_{\boldsymbol{a}\in\mathcal{H}} \frac{P_{\text{seq}}(\boldsymbol{a}|X)}{\sum_{\boldsymbol{a'}\in\mathcal{H}} P_{\text{seq}}(\boldsymbol{a'}|X)} \cdot R(\boldsymbol{a}, \boldsymbol{a}^r) \tag{4}$$

Here $\mathcal{H}$ is the hypotheses space, which is usually approximated with an N-best list. The risk function R is typically the Levenshtein distance w.r.t. the ground truth sequence $\boldsymbol{a}^r$. The P_{seq} is often chosen to be P_{RNNT} directly [15, 16], but may also include different LMs as in Eq. (2) [17]. This objective requires generating $\mathcal{H}$ by (semi-) on-the-fly decoding with the model in training, and computing FS for all label sequences in $\mathcal{H}$. Therefore, the MBR training has even much higher complexity and cost in terms of both time and memory.

In this section, we present the proposed 3-stage progressive training pipeline with detailed recipes. All hyper-parameters are extensively tuned on LBS only, which generalize well on other public corpora such as SWB and TED-LIUM Release 2 [18]. This efficient pipeline allows to train highly-performing transducer models from scratch with very limited computation resources in a reasonable short time period. We use conformer [7] transducer throughout this work, while the recipes also generalize well on common NN structures such as bidirectional long short-term memory [19] (BLSTM).

3.1. Stage 1: Viterbi training

As a first step, we directly apply the frame-wise cross-entropy (CE) training to a from-scratch initialized transducer NN:

$$\mathcal{L}_{\text{Viterbi}} = -\log P_{\text{RNNT}}(\hat{y}_1^T|h_1^T) = \sum_{t=1}^{T} -\log P_{\text{RNNT}}(\hat{y}_t|\mathcal{B}(\hat{y}_1^{t-1}), h_1^T)$$

where the Viterbi alignment $\hat{y}_1^T$ is obtained by training a very small CTC model. For label loops in the CTC alignment, we simply take the last frame of the segment as the label position.

As shown in [11, 20], such Viterbi training for transducer allows an easy integration of various regularization methods and auxiliary losses to improve performance and stability:

- 0.2 label smoothing [21] on $\mathcal{L}_{\text{Viterbi}}$
- $\mathcal{L}_{\text{enc}} = \sum_{t=1}^{T} -\log P(\hat{y}_t|h_t)$: an auxiliary CE loss w.r.t. $\hat{y}_1^T$ to the encoder output by introducing an additional softmax layer only in training, which is further weighted by a 1.0 focal loss factor [22]. The same can also be applied to the encoder middle layer with a smaller scale (default 0.3) [23].
- $\mathcal{L}_{\text{boost}}$: another auxiliary loss to boost the importance of output label sequence by simply excluding all blank frames in $\mathcal{L}_{\text{Viterbi}}$:

  $$\mathcal{L}_{\text{boost}} = \sum_{t=1}^{T} -\log P_{\text{RNNT}}(\hat{y}_t|\mathcal{B}(\hat{y}_1^{t-1}), h_1^T) \cdot (1 - \delta_{\hat{y}_t,\epsilon})$$

  By default, we further apply a scale $\alpha = 5$ to this loss.
- chunking: each utterance can also be split into windows with 50% overlap, where the window length can be adjusted based on the label context size of the model.

The above is our default recipe which yields the final loss as:

$$\mathcal{L}_{\text{Viterbi}}^{\text{final}} = \mathcal{L}_{\text{Viterbi}} + \mathcal{L}_{\text{enc}} + \alpha\mathcal{L}_{\text{boost}}$$

We also add a gradient clipping of 20 to ensure stability. We observe that this Viterbi training is very robust to varying quality of the alignment CTC model. The simplified criterion allows an easy learning, which gives fast convergence to good performance with only a small number of epochs.

In contrast to the standard pipeline, we obtain several gains on speed and memory in this stage:

1. We only need to train a very small CTC model till some reasonable performance to generate the Viterbi alignment.
2. The Viterbi training itself is much faster than FS (3 times faster in our setup) and needs less number of epochs to reach a similar good performance.
3. Much less memory is needed, which allows a much larger batch size. Besides speed-up, this also leads to a more reliable BatchNorm operation in the conformer model.

However, after quickly reaching a good local optimum, this Viterbi training barely improves further even with much longer training. This is mainly due to the limitation of fixed path training, which restricts the model to further converge to a better optimum. The problem can only be slightly eased with realignment, which nevertheless makes the training less efficient.

3.2. Stage 2: full-sum fine-tuning

As a result, we directly switch to FS fine-tuning using Eq. (3) as the 2nd stage after the model quickly converges in the Viterbi training stage. Without the aforementioned restriction, it gives the model more freedom to converge to a better optimum, which is also more consistent with the FS nature of transducer model definition. Since the starting point is already a good base model, this stage only needs a very small amount of epochs to achieve further decent improvement. This largely reduces the time cost of the FS training in the standard pipeline. Due to memory consumption, we have to accumulate gradients from several smaller batches to form a usual-size mini-batch for update. Therefore, we freeze the parameters in BatchNorm operation from this stage on, which is also more consistent with testing.

3.3. Stage 3: fast sequence training for LM integration

3.3.1. Objective

We also apply MBR sequence discriminative training as the 3rd stage to further improve the transducer model from stage 2. We aim to directly prepare the transducer model for external LM integration in this stage. As explained in [24], FS-trained transducer models usually have a quite high blank probability, which forces a rather small scale λ_1 for the external LM in SF decoding. Otherwise, the results will be biased towards shorter sequences with high deletion (Del) errors. This weakens the LM's discrimination power, which is usually handled in decoding by applying ILM correction [25, 26, 24] and/or heuristics such as length reward [27, 28]. However, such approaches require very careful tuning and a high quality of ILM estimation, which can be more problematic for transducers with limited context.

Therefore, we tend to directly tackle the fundamental issue by fine-tuning the transducer model with the MBR criterion towards an LM-aware distribution that further suppresses blank / boosts label around label positions while maintains the blank dominance elsewhere. Ideally, even with SF decoding, this allows a reasonably larger λ_1 to suppress substitution (Sub) and/or insertion (Ins) errors, while Del errors stay the same or decrease further. This should also make the scale-tuning easier for ILM correction if it is still necessary. Thus, we apply LM SF 1-pass decoding to generate the N-best list. We optimize λ_1 on the dev set using the base transducer model from stage 2, which usually gives higher Del errors than Ins errors. The P_{seq} in the MBR criterion of Eq. (4) is then defined accordingly:

$$P_{\text{seq}}(\boldsymbol{a}|X) = \left(P_{\text{RNNT}}(a_1^S|X) \cdot P_{\text{LM}}^{\lambda_1}(\mathcal{W}(a_1^S))\right)^{\beta}$$

where β is the global renormalization scale (typically $\beta = \frac{1}{\lambda_1}$ in our setup) and P_{LM} is frozen in the training. We always include $\boldsymbol{a}^r$ into the N-best and adopt 0.05-0.1 FS loss for stability, which yields the final loss as:

$$\mathcal{L}_{\text{MBR}}^{\text{final}} = \mathcal{L}_{\text{MBR}} + \alpha\mathcal{L}_{\text{FS}}$$

3.3.2. Static N-best generation

In [15], it is observed that a small N-best list usually does not change much within a certain training period, which allows a semi-on-the-fly N-best decoding using every new sub-epoch

model. Similarly, we also generate N-best separately and recompute the exact FS of P_{RNNT} for each sequence in the N-best in training. We observe that even with such semi-on-the-fly decoding, most improvement of the MBR training only comes in the first quarter of one full epoch. Based on these, we further simplify the pipeline by only using the stage-2 base model to decode random 25% of data for MBR training in this stage.

More specifically, we generate lattices for this subset of data to obtain the N-best list. For reuse purpose, the lattices can be slightly larger to allow an N-best size increment and to contain more unique label sequences after duplicates removal, e.g. homophones when a represents phonemes. This becomes a once-only and completely offline process which can be parallelized to many CPUs or a few GPUs if available. Different tuning experiments can then reuse the same lattices and finish within a few hours. Additionally, this lattice generation process can be further split into several sub-epochs, where the first training experiment can start as soon as the first sub-epoch of data is decoded. In this case, training and decoding can run in parallel with minor synchronization effort. This whole process has a strong similarity to the lattice-based MBR training in the classical hybrid systems [29, 30].

Our N-best-based MBR training allows a dynamic N to account for lattices containing less hypotheses than N, which can happen for some rather simple or short utterances. We also find that it is safe to use a slightly weaker LM with reduced context/model size to further speed up the lattice generation process. However, this weaker LM can not be too distinct from the final recognition LM so that the small N-best list still contains representative error patterns. With a largely reduced complexity, the 3rd stage MBR training gives some further improvement for LM integration within a very short time, which completes the proposed training pipeline.

3.4. Learning rate scheduling & epochs

Inspired by [28], we also apply the one-cycle learning rate (OCLR) policy [31] to enhance the effectiveness of our training pipeline. We try to more closely study the original OCLR design and adapt it to a simple formulation that requires least tuning effort and works very well in general with common adaptive optimizer. This can be expressed as a 3-phase LR scheduling within one training stage:

P1. 0-45% of total steps: linear increase LR from LR_1 to LR_{peak} with $\text{LR}_1 = \text{LR}_{\text{peak}}/10$

P2. 45-90% of total steps: linear decrease LR from LR_{peak} to LR_2 with $\text{LR}_2 = \text{LR}_{\text{peak}}/10$

P3. 90-100% of total steps: linear decrease LR from LR_2 to LR_{final} with $\text{LR}_{\text{final}} = 1e^{-6}$ typically

As claimed in [31], P1 and P2 share the same spirit as curriculum learning [32] and simulated annealing [33], respectively. And P3 tries to find the optima of local optimum with a quick decay to a much smaller LR. With this schedule, only LR_{peak} needs to be tuned, whose optimum is usually very close to the peak LR used in a common constant + exponential decay policy. We observe that the described OCLR schedule gives consistently better performance than the common one under the same training epochs. For stage 1 training, we use this OCLR schedule exactly. For stage 2, we adjust P1 with $\text{LR}_1 = \text{LR}_{\text{peak}}$ and P2 with $\text{LR}_2 = \text{LR}_{\text{peak}}/5$. For stage 3, we simply use a constant LR $1e^{-5}$. By default, we use a mini-batch size of 10-15k input frames, $5e^{-6}$ L2 regularization and 0.1 dropout, which works generally well with this schedule. In terms of epochs, we find that a reasonable range is to use 10-30k hours divided by the corpus duration for stage 1, and 50-75% of that for stage 2.

Table 1: *Number of epochs (#ep) for different transducer models in stage 1 and 2 training, and corresponding WER [%] results on the LBS dev-clean/other sets and SWB Hub5'00 set. All results with LM are from shallow fusion (SF) decoding only.*

Train Stage	Phoneme Transducer						ADSM Transducer			
	LM	LBS			SWB		LM	LBS		
		#ep	clean	other	#ep	Hub5'00		#ep	clean	other
1	4gram	20	2.9	6.9	50	11.4	-	30	3.1	8.4
2	4gram	15	2.6	6.0	36	10.7	-	15	2.7	7.3
2	Trafo	15	1.8	4.1	36	9.9	Trafo	15	1.9	4.6

Table 2: *Stage-1 Viterbi training using various CTC models for alignment generation and correspondingly-trained phoneme transducer models (all 20 epochs). WER [%] results using 4gram LM SF decoding on the LBS dev-other set.*

Alignment CTC Model				Stage-1 Phoneme Transducer		
NN	size	#ep	WER	Encoder NN	size	WER
BLSTM	6×512	20	9.7	BLSTM	6×512	8.1
					6×640	7.7
		30	9.4	VGG-Conf.	12 blocks	6.9
BLSTM-Conf.	3×512 6 blocks		9.0			6.9
VGG-Conf.	12 blocks		7.6			6.9

4. Experiments

4.1. Setups

Detailed experiments are conducted on the 960h LBS corpus [8] and the 300h SWB corpus [9]. We evaluate the proposed training pipeline on context-1 transducer models using phonemes for both corpora, and full-context transducer models using 5k acoustic data-driven subword modeling (ADSM) units [34] for LBS. Additionally, we reduce the LBS phoneme inventory in the official lexicon by unifying stressed phonemes, e.g. (AA0, AA1, AA2): AA, which we find harmless for the overall performance. Following [11], we further apply end-of-word augmentation to the phoneme sets of both corpora. For SWB, we use Hub5'00 as dev set, and Hub5'01 and RT'03 as test sets.

Similar as [23], we use a 12×512 conformer (Conf.) [7] encoder with an initial VGG network [35]. The VGG network contains 3 3-by-3 convolutional layers with number of filters 32, 64 and 64, respectively. The last 2 convolutional layers use a temporal stride of 2 to achieve a total subsampling of factor 4. Additionally, we swap the convolution and multi-headed self-attention modules in the conformer blocks as this gives slightly better performance in our setup. For the prediction network, we use 2×640 feed-forward network and 2×640 LSTM layers for context-1 and full-context transducer models, respectively. The standard additive joint network is used, which contains a linear-tanh layer of size 1024 and a final linear-softmax layer.

We use gammatone features (LBS: 50-dim; SWB: 40-dim) [36], and specaugment [37] except for stage 3. All model training is performed on single GTX 1080 Ti GPU. By default, we apply 1-pass LM SF decoding, where word-level LM is used for phoneme transducers. The word-level transformer (Trafo) LMs are the same as in [38] for LBS and [39] (sentence-wise) for SWB, while the ADSM Trafo LM is the same as in [34].

4.2. Stage 1: Viterbi training

By default, we use a 6×512 BLSTM for the alignment CTC model, which adopts the same subsampling via max-pooling layers in the middle. For LBS and SWB, we train the CTC model for 20 epochs and 35 epochs, respectively. After generating the Viterbi alignment, we apply the stage-1 training on transducer models. The LR_{peak} for phoneme and ADSM transducers are $8e^{-4}$ and $3e^{-4}$, respectively. The corresponding

Table 3: *Efficiency illustration (on single GTX 1080 Ti GPU) of the proposed training with phoneme transducer; vs. standard training under similar WER [%] on the LBS dev sets*

Model Train	LBS					SWB		
	#ep	hour/ep	$\sum$ hour	clean	other	#ep	hour/ep	$\sum$ hour
standard	40	18.3	732	2.6	6.1	-		
stage 1	20	6	-			50	1.8	-
+ stage 2	+ 15	18.3	394.5	2.6	6.0	+ 36	5.5	288

Table 4: *WER [%] and LM integration effect of stage 3 training on the stage-2 transducer models; results evaluated with SF and ILM correction on the LBS dev-other and SWB Hub5'00 sets; further detailed scales and Sub/Del/Ins error rate [%] on LBS.*

Model	Train Stage	LM	LBS						SWB
			λ_1	λ_2	dev-other				Hub5'00
					WER	Sub	Del	Ins	WER
phoneme Transd.	2	Trafo	0.9	0	4.1	3.1	0.6	0.4	9.9
		+ ILM	1.0	0.2	3.9	3.1	0.4	0.4	9.7
	3	Trafo	1.3	0	**3.7**	3.0	0.4	0.4	9.3
		+ ILM	1.4	0.1	**3.7**	2.9	0.4	0.4	**9.2**
ADSM Transd.	2	Trafo	0.6	0	4.6	3.5	0.7	0.4	-
		+ ILM	0.8	0.4	4.0	3.2	0.4	0.4	
	3	Trafo	0.9	0	4.2	3.4	0.4	0.4	
		+ ILM	1.2	0.3	4.0	3.2	0.4	0.4	

epochs and word error rate (WER) results are shown in Table 1.

We use the LBS phoneme transducer to illustrate the robustness of this training stage. Different CTC models are evaluated for alignment generation, which covers different NN structures, epochs and accuracy as shown in Table 2. We see that the correspondingly Viterbi-trained conformer-transducer models all achieve the same performance, where the same recipe also works well for BLSTM-transducer models. We believe the cost of the alignment CTC model can be further reduced, which we didn't investigate much here.

4.3. Stage 2: full-sum fine-tuning

Based on cross-validation score or intermediate recognition, the best model checkpoints of stage 1 are selected for stage-2 training. For SWB, we increase dropout to 0.2 from this stage on to avoid overfitting with the fine-tuning process. We find LR_{peak} around $5e^{-5}$ to be a good optimum for all models. The corresponding number of epochs and WER results of this training stage are shown in Table 1. We see that decent improvements are achieved in all cases with only a small amount of epochs.

In Table 3, we show the efficiency of the proposed training pipeline using phoneme transducers, which is compared with the standard pipeline on LBS. We use the well-trained VGG-Conf. CTC model from Table 2 for encoder initialization and directly train a transducer model using Eq. (3). The same OCLR schedule as in stage 1 is applied. We train this standard model for 40 epochs till it reaches the same performance as our stage-2 model. Based on the total time used, our proposed pipeline achieves a 46% relative speed-up.

4.4. Stage 3: fast sequence training for LM integration

The best models from stage 2 are then further trained in stage 3 with the fast MBR training described in Section 3.3. As the weaker LM for lattice generation, we use the same recognition 4gram LM for phoneme transducers, and a 1×1024 LSTM LM trained on the LM text data for the ADSM transducer. We filter out utterances longer than 16s in time, or containing more than 190 phonemes or 90 ADSM labels in the transcription. The MBR training uses an N-best of size 4. Additionally for the SWB phoneme transducer, we exclude the non-speech labels for sequence uniqueness and risk computation. Besides WER, we also evaluate LM integration effect using SF and ILM correction for decoding, where the zero-encoder ILM approach [25, 26] is applied for the latter. The detailed results are shown in Table 4.

With LM SF decoding, consistent improvements are achieved by the stage-3 training upon all the stage-2 transducer models. As claimed in Section 3.3, we see that the higher Del errors are reduced to a balanced level with Ins, while the LM scale λ_1 is increased to a more reasonable value to further suppress the Sub errors. With ILM correction, the stage-2 transducer models already achieve better results. Yet the stage-3 training brings further 5% relative improvements to both phoneme transducers. However, the benefit of ILM correction for the full-context ADSM transducer becomes much smaller after stage-3 training, which leads to no overall improvement comparing to the stage-2 ADSM transducer with ILM correction. It seems that the proposed fast sequence training is more helpful when the ILM is less powerful, e.g. limited context. We will further investigate this joint effect in future work.

Table 5: *Overall WER [%] results on LBS vs. literature including number of parameters (#pm) and epochs (#ep)*

Work	Model			LM	dev		test	
	Approach	#pm	#ep		clean	other	clean	other
[40]	RNN AED	360M	600	LSTM	-		2.2	5.2
[41]	Trafo Hybrid	81M	100	Trafo			2.3	4.9
[42]	Trafo CTC	124M	200				2.1	4.2
[7]	Conf. Transd.	119M	-	LSTM			1.9	3.9
this	ADSM Transd.	87M	45	Trafo	1.8	4.0	1.9	4.4
	phoneme Transd.	75M	35	4gram	2.5	5.7	2.9	6.2
				Trafo	1.7	3.7	2.1	4.1

Table 6: *Overall WER [%] results on SWB vs. literature.*
(: cross-utterance evaluation with combined LSTM and Trafo LMs)*

Work	Model			LM	Hub5'00	Hub5'01	RT'03
	Approach	#pm	#ep				
[23]	Conf. Hybrid	88M	-	Trafo	10.3	9.7	-
[28]	RNN Transd.	57M	100	LSTM	9.7	10.1	12.6
[43]	RNN AED	280M	250		9.8	10.1	12.0
[44]	Conf. AED	68M	250	+ Trafo	8.4^*	8.5^*	9.9^*
this	phoneme Transd.	75M	86	4gram	10.3	10.6	11.8
				Trafo	9.2	9.4	10.5

4.5. Overall performance

Finally, we evaluate the overall performance of our best phoneme transducer models from stage 3 and best ADSM transducer model from stage 2. We switch the ILM correction for ADSM transducer to the mini-LSTM ILM approach [45] as done in [24]. All scales are optimized on the dev sets. We compare these results with other top systems from the literature in Table 5 and Table 6 for LBS and SWB, respectively. Our best systems are competitive with SOTA results, while our models are smaller and/or trained with much less epochs.

5. Conclusions

In this work, we presented an efficient 3-stage progressive training pipeline for both phoneme and subword neural transducer models. The provided detailed recipes and design principles of each stage are experimentally verified on both LBS and SWB corpora. Compared with the standard pipeline, our proposed training pipeline achieves a large reduction on time and computation costs as well as complexity. This allows us to build conformer transducer models approaching SOTA performance from scratch with a single GPU in just 2-3 weeks.

Acknowledgements

This work was partially supported by the Google Faculty Research Award for "Label Context Modeling in Automatic Speech Recognition", and by NeuroSys which, as part of the initiative "Clusters4Future", is funded by the Federal Ministry of Education and Research BMBF (03ZU1106DA).

We thank Zuoyun Zheng for training the ADSM LMs.

6. References

[1] A. Graves, S. Fernández, F. J. Gomez, and J. Schmidhuber, "Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks," in *Proc. ICML*, 2006, pp. 369–376.

[2] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-End Attention-based Large Vocabulary Speech Recognition," in *Proc. ICASSP*, 2016, pp. 4945–4949.

[3] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, "Listen, Attend and Spell: A Neural Network for Large Vocabulary Conversational Speech Recognition," in *Proc. ICASSP*, 2016, pp. 4960–4964.

[4] A. Graves, "Sequence Transduction with Recurrent Neural Networks," 2012, https://arxiv.org/abs/1211.3711.

[5] Q. Zhang, H. Lu, H. Sak, A. Tripathi, E. McDermott, S. Koo, and S. Kumar, "Transformer Transducer: A Streamable Speech Recognition Model with Transformer Encoders and RNN-T Loss," in *Proc. ICASSP*, 2020, pp. 7829–7833.

[6] W. Han, Z. Zhang, Y. Zhang, J. Yu, C. Chiu, J. Qin, A. Gulati, R. Pang, and Y. Wu, "ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context," in *Proc. Interspeech*, 2020, pp. 3610–3614.

[7] A. Gulati, J. Qin, C. Chiu, N. Parmar, Y. Zhang, J. Yu, W. Han, S. Wang, Z. Zhang, Y. Wu, and R. Pang, "Conformer: Convolution-augmented Transformer for Speech Recognition," in *Proc. Interspeech*, 2020, pp. 5036–5040.

[8] V. Panayotov, G. Chen, D. Povey, and S. Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[9] J. J. Godfrey, E. C. Holliman, and J. McDaniel, "SWITCHBOARD: Telephone Speech Corpus for Research and Development," in *Proc. ICASSP*, vol. 1, 1992, pp. 517–520.

[10] A. Tripathi, H. Lu, H. Sak, and H. Soltau, "Monotonic Recurrent Neural Network Transducer and Decoding Strategies," in *IEEE ASRU*, 2019, pp. 944–948.

[11] W. Zhou, S. Berger, R. Schlüter, and H. Ney, "Phoneme Based Neural Transducer for Large Vocabulary Speech Recognition," in *Proc. ICASSP*, 2021, pp. 5644–5648.

[12] M. Ghodsi, X. Liu, J. Apfel, R. Cabrera, and E. Weinstein, "Rnn-Transducer with Stateless Prediction Network," in *Proc. ICASSP*, Barcelona, Spain, 2020, pp. 7049–7053.

[13] R. Prabhavalkar, Y. He, D. Rybach, S. Campbell, A. Narayanan, T. Strohman, and T. N. Sainath, "Less is More: Improved RNN-T Decoding Using Limited Label Context and Path Merging," in *Proc. ICASSP*, Toronto, Canada, 2021, pp. 5659–5663.

[14] C. Gulcehre et al., "On Using Monolingual Corpora in Neural Machine Translation," 2015, http://arxiv.org/abs/1503.03535.

[15] J. Guo, G. Tiwari, J. Droppo, M. V. Segbroeck, C. Huang, A. Stolcke, and R. Maas, "Efficient Minimum Word Error Rate Training of RNN-Transducer for End-to-End Speech Recognition," in *Proc. Interspeech*, 2020, pp. 2807–2811.

[16] C. Weng, C. Yu, J. Cui, C. Zhang, and D. Yu, "Minimum Bayes Risk Training of RNN-Transducer for End-to-End Speech Recognition," in *Proc. Interspeech*, 2020, pp. 966–970.

[17] Z. Meng, Y. Wu, N. Kanda, L. Lu, X. Chen, G. Ye, E. Sun, J. Li, and Y. Gong, "Minimum Word Error Rate Training with Language Model Fusion for End-to-End Speech Recognition," in *Proc. Interspeech*, 2021, pp. 2596–2600.

[18] A. Rousseau, P. Deléglise, and Y. Estève, "Enhancing the TED-LIUM Corpus with Selected Data for Language Modeling and More TED Talks," in *Proc. LREC*, 2014, pp. 3935–3939.

[19] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[20] A. Zeyer, A. Merboldt, R. Schlüter, and H. Ney, "A New Training Pipeline for an Improved Neural Transducer," in *Proc. Interspeech*, 2020, pp. 2812–2816.

[21] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the Inception Architecture for Computer Vision," in *CVPR*, 2016, pp. 2818–2826.

[22] T. Lin, P. Goyal, R. B. Girshick, K. He, and P. Dollár, "Focal Loss for Dense Object Detection," in *IEEE International Conference on Computer Vision ICCV*, 2017, pp. 2999–3007.

[23] M. Zeineldeen, J. Xu, C. Lüscher, W. Michel, A. Gerstenberger, R. Schlüter, and H. Ney, "Conformer-based Hybrid ASR System for Switchboard Dataset ," in *Proc. ICASSP*, 2022.

[24] W. Zhou, Z. Zheng, R. Schlüter, and H. Ney, "On Language Model Integration for RNN Transducer based Speech Recognition," in *Proc. ICASSP*, Singapore, May 2022.

[25] E. Variani, D. Rybach, C. Allauzen, and M. Riley, "Hybrid Autoregressive Transducer (HAT)," in *Proc. ICASSP*, 2020, pp. 6139–6143.

[26] Z. Meng, S. Parthasarathy, E. Sun, Y. Gaur, N. Kanda, L. Lu, X. Chen, R. Zhao, J. Li, and Y. Gong, "Internal Language Model Estimation for Domain-Adaptive End-to-End Speech Recognition," in *IEEE SLT*, 2021, pp. 243–250.

[27] E. McDermott, H. Sak, and E. Variani, "A Density Ratio Approach to Language Model Fusion in End-to-End Automatic Speech Recognition," in *IEEE ASRU*, 2019, pp. 434–441.

[28] G. Saon, Z. Tüske, D. Bolaños, and B. Kingsbury, "Advancing RNN Transducer Technology for Speech Recognition," in *Proc. ICASSP*, 2021, pp. 5654–5658.

[29] K. Veselý, A. Ghoshal, L. Burget, and D. Povey, "Sequence-discriminative training of deep neural networks," in *Proc. Interspeech*, 2013, pp. 2345–2349.

[30] W. Zhou, W. Michel, K. Irie, M. Kitza, R. Schlüter, and H. Ney, "The RWTH ASR system for TED-LIUM release 2: Improving Hybrid HMM with SpecAugment," in *Proc. ICASSP*, Barcelona, Spain, 2020, pp. 7839–7843.

[31] L. N. Smith and N. Topin, "Super-Convergence: Very Fast Training of Neural Networks using Large Learning Rates," in *Artificial Intelligence and Machine Learning for Multi-Domain Operations Applications*, vol. 11006, 2019, pp. 369 – 386.

[32] Y. Bengio, J. Louradour, R. Collobert, and J. Weston, "Curriculum learning," in *Proc. ICML*, vol. 382, 2009, pp. 41–48.

[33] E. H. L. Aarts and J. H. M. Korst, *Simulated Annealing and Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing*, 1990.

[34] W. Zhou, M. Zeineldeen, Z. Zheng, R. Schlüter, and H. Ney, "Acoustic Data-Driven Subword Modeling for End-to-End Speech Recognition," in *Proc. Interspeech*, 2021, pp. 2886–2890.

[35] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," in *Int. Conf. on Learning Representations (ICLR)*, 2015.

[36] R. Schlüter, I. Bezrukov, H. Wagner, and H. Ney, "Gammatone Features and Feature Combination for Large Vocabulary Speech Recognition," in *Proc. ICASSP*, 2007, pp. 649–652.

[37] B. Zoph, C.-C. Chiu, D. S. Park, E. D. Cubuk, Q. V. Le, W. Chan, and Y. Zhang, "SpecAugment: A Simple Augmentation Method for Automatic Speech Recognition," in *Proc. Interspeech*, 2019, pp. 2613–2617.

[38] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Language Modeling with Deep Transformers," in *Proc. Interspeech*, Graz, Austria, 2019, pp. 3905–3909.

[39] K. Irie, A. Zeyer, R. Schlüter, and H. Ney, "Training Language Models for Long-Span Cross-Sentence Evaluation," in *IEEE ASRU*, 2019, pp. 419–426.

[40] D. S. Park et al., "Specaugment on Large Scale Datasets," in *Proc. ICASSP*, 2020, pp. 6879–6883.

[41] Y. Wang et al., "Transformer-Based Acoustic Modeling for Hybrid Speech Recognition," in *Proc. ICASSP*, 2020, pp. 6874–6878.

[42] F. Zhang, Y. Wang, X. Zhang, C. Liu, Y. Saraf, and G. Zweig, "Faster, Simpler and More Accurate Hybrid ASR Systems Using Wordpieces," in *Proc. Interspeech*, 2020, pp. 976–980.

[43] Z. Tüske, G. Saon, K. Audhkhasi, and B. Kingsbury, "Single Headed Attention based Sequence-to-sequence Model for State-of-the-Art Results on Switchboard," in *Proc. Interspeech*, 2020, pp. 551–555.

[44] Z. Tüske, G. Saon, and B. Kingsbury, "On the Limit of English Conversational Speech Recognition," in *Proc. Interspeech*, 2021, pp. 2062–2066.

[45] M. Zeineldeen, A. Glushko, W. Michel, A. Zeyer, R. Schlüter, and H. Ney, "Investigating Methods to Improve Language Model Integration for Attention-based Encoder-Decoder ASR Models," in *Proc. Interspeech*, 2021, pp. 2856–2860.