ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) | 978-1-7281-7605-5/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICASSP39728.2021.9414784

IMPROVED NEURAL LANGUAGE MODEL FUSION FOR STREAMING RECURRENT NEURAL NETWORK TRANSDUCER

Suyoun Kim, Yuan Shangguan, Jay Mahadeokar, Antoine Bruguier
Christian Fuegen, Michael L. Seltzer, Duc Le

Facebook AI, USA

ABSTRACT

Recurrent Neural Network Transducer (RNN-T), like most end-to-end speech recognition model architectures, has an implicit neural network language model (NNLM) and cannot easily leverage unpaired text data during training. Previous work has proposed various fusion methods to incorporate external NNLMs into end-to-end ASR to address this weakness. In this paper, we propose extensions to these techniques that allow RNN-T to exploit external NNLMs during both training and inference time, resulting in 13-18% relative Word Error Rate improvement on Librispeech compared to strong baselines. Furthermore, our methods do not incur extra algorithmic latency and allow for flexible plug-and-play of different NNLMs without re-training. We also share in-depth analysis to better understand the benefits of the different NNLM fusion methods. Our work provides a reliable technique for leveraging unpaired text data to significantly improve RNN-T while keeping the system streamable, flexible, and lightweight.

Index Terms— RNN-T, language model fusion, streaming end-to-end speech recognition, leveraging unpaired text

1. INTRODUCTION

Recurrent Neural Network Transducer (RNN-T) [1–5] has become one of the most popular model architectures for on-device streaming automatic speech recognition (ASR) over the last few years. Compared to traditional hybrid ASR systems, RNN-T is much more compact due to the lack of external n-gram language models (LMs) and decision trees. Compared to other end-to-end ASR approaches such as encoder-decoder with attention [6–10], RNN-T is easier to stream and generally works better in low-latency scenarios where the entire utterance is not available up front and partial decoding results need to be emitted during decoding.

Although RNN-T has the advantage of streamability over encoder-decoder based ASR models, a recent study [11] found that the prediction network of RNN-T, often thought of as an implicit LM, shows poor results in modeling long-range linguistic information. In addition, the model's end-to-end nature makes it difficult to leverage unpaired text data to further improve performance. Many recent works have investigated methods for using text-only data to improve encoder-decoder based ASR models, including fusion with an external neural network LM (NNLM) [12–16]. There has been comparatively limited work on NNLM fusion for RNN-T with streaming constraints, where shallow fusion remains the most popular and effective technique [13,17,18]. A recent study [11] tried pre-training the prediction network of RNN-T with unpaired text, but did not get any WER improvement.

In this work, we explore NNLM fusion methods for RNN-T that are applied on-the-fly during first pass decoding, thus avoiding additional algorithmic latency and keeping the model latency low. We propose extensions to the original cold NNLM fusion to increase its flexibility and effectiveness within the RNN-T framework. Our combined cold and shallow NNLM fusion method achieves **13-18%** relative WER improvement on the widely used Librispeech dataset over our strong baselines. In addition, our method allows for flexible plug-and-play of different NNLMs without the need for re-training, which could be very useful for rapid domain adaptation and dynamically adjusting to resource constraints. Lastly, we provide in-depth analysis to better understand the benefits of the different NNLM fusion methods.

2. NNLM FUSION FOR RNN-T

2.1. RNN-T Overview

RNN-T [1] consists of three major sub-networks: encoder, predictor, and joiner. The encoder transforms an input sequence of audio feature vectors $\mathbf{x} = (x_1, \ldots, x_T)$ into a sequence of acoustic embeddings $\mathbf{h}^{enc}$:

$$\mathbf{h}^{enc} = f^{enc}(\mathbf{x}) = (h_1^{enc}, \ldots, h_{T'}^{enc}) \quad (1)$$

where T' may be different from T. The predictor, which is analogous to an LM, transforms a sequence of previous tokens $(y_1, \ldots, y_{u-1})$ into an embedding vector h_u^{pred}:

$$h_u^{pred} = f^{pred}(y_1, \ldots, y_{u-1}) \quad (2)$$

Finally, the joiner combines the encoder embedding h_t^{enc} and predictor embedding h_u^{pred} to estimate the logits $z_{t,u}$:

$$z_{t,u} = f^{join}(h_t^{enc}, h_u^{pred}) \quad (3)$$

$$P(.|x_1, \ldots, x_t, y_1, \ldots, y_{u-1}) = \text{softmax}(z_{t,u}) \quad (4)$$

Additional details on the RNN-T training objective and decoding procedure can be found in [1].

2.2. Shallow Fusion

Shallow fusion is the most popular technique for combining RNN-T with an external NNLM trained on text-only data [13]. In shallow fusion, the NNLM is incorporated via log-linear interpolation at inference time, and the decoding problem of finding the best hypothesis y^* becomes:

$$y^* = \arg\max_y \log P_{\text{RNNT}}(y|x) + \lambda \log P_{LM}(y) \quad (5)$$

where λ is a hyperparameter that controls the relative importance of the external NNLM ($\lambda = 0$ corresponds to normal RNN-T decoding without shallow fusion). Note that unlike encoder-decoder models, we cannot directly interpolate the log probability of RNN-T's joiner output and NNLM output at each output time step because RNN-T allows emission of blank symbols which are not modeled in external NNLMs. In our implementation, we interpolate external NNLM scores with RNN-T scores during beam search when the model outputs a non-blank output symbol. This interpolation happens on-the-fly and the overall system remains streamable.

2.3. Cold Fusion

One limitation of shallow fusion is that the external NNLM is only applied during inference. Cold fusion [12, 14] is a method originally proposed for encoder-decoder models where a pre-trained external NNLM is fused directly into the decoder network by combining their hidden states during training time. Similar to the decoder network of encoder-decoder models, the prediction network of RNN-T is analogous to an LM. Our proposed cold fusion method for RNN-T extends Equation (2) to combine the predictor embedding and NNLM output as follows:

$$s_u^{LM} = \text{softmax}(z_u^{LM}) \quad (6)$$

$$h_u^{LM} = f^{LM}(s_u^{LM}) \quad (7)$$

$$g_u = \text{sigmoid}(W_g[h_u^{pred}; h_u^{LM}] + b_g) \quad (8)$$

$$h_u^{CF} = f^{CF}(g_u \odot [h_u^{pred}; h_u^{LM}]) \quad (9)$$

where z_u^{LM} is the external NNLM's predicted logits over non-blank output symbols for the next time step given a sequence of previously emitted tokens $(y_1, \ldots, y_{u-1})$, f^{LM} is the LM projection network, and f^{CF} is the combined projection network. The final embedding h_u^{CF} has the same dimension as h_u^{pred} and replaces the latter in Equation (3).

Some of the key differences between our cold fusion approach and the original cold fusion formulation are:

1. We adopt an iterative training procedure instead of training the network from scratch. This means the RNN-T and NNLM are first pre-trained separately; the cold fusion RNN-T is then finetuned with the frozen NNLM for a few epochs. We will show in Section 4.2 that this technique is crucial for cold fusion to work.

2. We use the NNLM's logits z_u^{LM} instead of its hidden state. This allows us to swap in a different NNLM during inference without re-training the whole network. Example scenarios where this ability could be useful are switching to a lightweight NNLM where computation power is limited, or plugging in a domain-specific NNLM for improved accuracy. We will showcase this flexibility in Section 4.3.

3. We apply a fine gating mechanism on top of the concatenated predictor and NNLM output, as shown in Equation (8) and (9). This gating mechanism can increase the predictor network's modeling power by using multiplicative interaction [19], together with additional linguistic information from the external NNLM.

With cold fusion, the system remains streamable since NNLM scores are combined on-the-fly in first pass decoding.

3. EXPERIMENTS

Data: We conduct experiments on the widely used Librispeech [22] dataset which consists of 960 hours of labeled speech and an additional text-only corpus containing 810M words. We use 80-dim globally z-normalized logMel filterbank coefficients as acoustic features, derived from 25ms FFT windows with a 10ms frame shift. We apply the Librispeech Double policy without time warping from SpecAugment [23] during training. We also perform speed perturbation [24] of the training data and produce three versions of each audio with speed factors 0.9, 1.0, and 1.1; as a result, the training data size is tripled. The output targets are 5000 unigram WordPieces [25] generated by the SentencePiece toolkit [26], plus an additional blank symbol.

RNN-T Encoder: We consider two of the most popular streaming encoder architectures for RNN-T, Long-Short Term Memory (LSTM) and Latency Controlled Bidirectional LSTM (LC-BLSTM) [27] as our baselines in this work. The LSTM encoder stacks 11 contiguous feature frames as input, consists of eight layers with 1024 cells each, and has a 640-dim linear projection after every LSTM layer. The LC-BLSTM encoder works on single feature frames without stacking, has 24-frame lookahead (i.e., 240ms), 120-frame chunk size, and comprises eight layers with 640 cells each. Both encoders subsample the input by a factor of four and

Encoder Arch.	Params	Lookahead	LM Fusion	test-clean	test-other
BLSTM [20]	∼126M	inf (non-streaming)	None	3.2	7.8
Transformer [21]	139M	30ms (feature stacking)	None	4.2	11.3
		∼1.1s (2 frames/layer)		3.0	7.7
LSTM (ours)	65M	100ms (feature stacking)	None	4.0	10.1
			SF	3.3	8.6
			CF	3.8	9.4
			SF+CF	**3.3**	**8.3**
LC-BLSTM (ours)	99M	240ms (right context)	None	3.2	8.0
			SF	2.8	7.0
			CF	3.0	7.6
			SF+CF	**2.8**	**6.8**

Table 1: Librispeech WER comparison between vanilla RNN-T baselines, shallow fusion (SF), cold fusion (CF), combined fusion (SF+CF), and relevant published results using sequence transducers.

produce 1024-dim embeddings. In both networks, the predictor contains two LSTM layers with 512 cells each, and the joiner has a single linear layer. The total trainable parameters are 65M (LSTM) and 99M (LC-BLSTM). We train the models for 80 epochs using Adam [28]; the learning rate is fixed at 0.0004 for the first 60 epochs, then drops by a factor of 0.8 after every subsequent epoch.

External NNLM: We employ a 4-layer LSTM network with 2048 cells in each layer, interleaved with 640-dim linear projection, totaling 53M trainable parameters. We train the model on the 810M text-only corpus (broken down into WordPieces) with Cross Entropy loss for 40 epochs using the Adam optimizer [28]. The learning rate is fixed at 0.0004 for the first 25 epochs, then drops by a factor of 0.8 after every subsequent epoch. This NNLM will be the basis for our shallow fusion and cold fusion experiments.

Cold Fusion: The LM projection network f^{LM} contains a 256-dim bottleneck layer, followed by linear projection into 1024 dimensions. The combined projection network f^{CF} consists of a single 1024-dim linear projection layer. These cold fusion-specific components add 7.8M trainable parameters in total. We freeze the NNLM parameters, bootstrap the RNN-T from baseline models, and finetune the network for 10 epochs with a 0.0005 learning rate for the first 3 epochs, then decays by a factor of 0.6 after every epoch. For fair comparison, we also tried finetuning the baseline models for 10 more epochs, but did not obtain better results.

Decoding: We use a beam size of 15 for all experiments. The shallow fusion interpolation weight λ ranges between 0.2 and 0.5 based on tuning results on development sets. The optimal λ is smaller when cold fusion is combined with shallow fusion, likely because the NNLM scores are already implicit within the cold fusion RNN-T scores. Furthermore, we can reuse the NNLM logits for both fusion methods, thus incurring minimal computational overhead when combining them.

4. RESULTS AND DISCUSSION

4.1. WER Overview

Table 1 shows the WER results of our proposed approaches, together with some relevant baselines from published works that also utilized sequence transducers. The main motivation for including these published results is to demonstrate that our baseline model's numbers are very competitive. We are not aiming to outperform the Librispeech state-of-the-art in this paper, which typically entails using non-streamable encoder architectures (e.g., full-context transformer) as well as second pass rescoring on full utterances, whereas the focus of our work is on streaming first-pass decoding.

Both shallow fusion (SF) and cold fusion (CF) significantly improve over the vanilla baselines, with SF giving better results than CF. It is unclear why CF underperforms compared to SF, even though the NNLM is incorporated directly in training. We hypothesize that SF is able to distribute the probability mass more evenly to different WordPieces via direct interpolation, whereas the spiky nature of RNN-T scores limits the impact of CF. We will verify this hypothesis in future work. Combining SF and CF results in further improvement on the more challenging test-other split, producing an overall WER reduction of **13-18%** over the baselines.

4.2. Importance of Iterative Training

Table 2 shows that training cold fusion RNN-T from scratch fails to yield WER improvement; we also observed that the model has difficulties converging, especially with an LSTM encoder. It is therefore crucial to adopt the iterative training approach, i.e., bootstrapping from a well-trained RNN-T model. We hypothesize that the strong signal from the external NNLM makes the model rely less on RNN-T, thus the latter becomes under-trained. Conversely, most parts of the net-

Model	test-clean	test-other
LSTM CF (from scratch)	Did Not Converge	
LSTM CF (iterative)	3.8	9.4
LC-BLSTM CF (from scratch)	3.2	8.4
LC-BLSTM CF (iterative)	3.0	7.6

Table 2: Effect of iterative training on cold fusion (CF).

Model	test-clean	test-other
LSTM CF (15M LM)	4.0	9.7
LSTM CF (53M LM)	3.8	9.4
+ swap 15M LM	3.9	9.6
+ swap 15M oracle LM	2.2	5.8
LC-BLSTM CF (15M LM)	3.1	7.8
LC-BLSTM CF (53M LM)	3.0	7.6
+ swap 15M LM	3.1	7.8
+ swap 15M oracle LM	1.9	4.9

Table 3: Swapping NNLM in cold fusion without re-training.

work are already well-trained in the iterative scenario (only a few linear layers for cold fusion need to be trained from scratch), and the finetuning process mainly teaches the model how to integrate the available signals.

4.3. Flexible NNLM Swapping

The use of NNLM's logits z_u^{LM} allows for flexible plug-and-play of different NNLMs during inference without re-training the whole network. This flexibility can be useful in many scenarios. For example, we only have to train the model once and simply plug in different NNLMs for different devices or surfaces depending on their resource constraints. Table 3 shows an example where we swap out the original NNLM (53M parameters) with a lightweight version (15M parameters). The swapped-in LM works out of the box and gives similar results as using it directly in cold fusion training.

In cases where we have strong apriori knowledge about the input audio, such as domain information, we could plug in a domain-specific NNLM to obtain better recognition results. Table 3 illustrates this ability where we swap out the original 53M NNLM with an oracle 15M NNLM trained on the combined `test-clean` and `test-other` splits, resulting in massive WER reduction.

4.4. When Does NNLM Fusion Help?

We first analyze WER improvement as a function of utterance length. We split utterances in the evaluation set into three chunks: Short, Medium, and Long. Each chunk represents a third of the utterances in the evaluation set, containing around 1.85K utterances. As shown in Table 4 (first section), the improvement provided by all fusion methods increases as the utterance becomes longer. This implies that NNLMs can better model long-range linguistic information and compensate for the known weakness of RNN-T's prediction network [11].

	SF	CF	SF+CF
Average Length			
Short (8 words)	6.7%	3.5%	9.5%
Medium (16 words)	14.3%	6.7%	17.2%
Long (34 words)	18.7%	7.2%	20.7%
Word Type (# Utts)			
Common (3.8K)	17.4%	7.9%	20.2%
Fixed by LM (1.5K)	14.3%	5.4%	16.5%
Rare/OOV (332)	9.1%	3.8%	11.0%

Table 4: Breakdown of relative WER reduction of shallow fusion (SF), cold fusion (CF), and combined fusion (SF+CF) compared to vanilla baselines. Analysis is done on LSTM's `test-clean` and `test-other` results.

Next we analyze the relation between the rare/out-of-vocabulary (OOV) word issue and NNLM fusion improvement. We define rare/OOV word as a word that appears less than 10 times in the acoustic training transcription. We split utterances in the evaluation set into three chunks: (1) `Common` - utterances that have no rare/OOV word, (2) `Fixed by LM` - utterances that had rare/OOV word, but the issue was fixed if we include LM unpaired text data, and (3) `Rare/OOV` - utterances that still have at least one rare/OOV word after including LM unpaired text data. As shown in Table 4 (second section), the improvement provided by NNLM fusion increases when the unpaired text data are able to mitigate the rare/OOV word issue. This suggests that the increased coverage of the unpaired text corpus plays a crucial role in NNLM fusion's effectiveness.

5. CONCLUSION AND FUTURE WORK

In this work, we proposed external NNLM fusion methods for RNN-T models capable of leveraging unpaired text data in both training and decoding. Our methods are applied on-the-fly during first pass decoding, thus do not adversely impact algorithmic latency and the models remain streamable. Moreover, we showed that iterative training is crucial for getting cold fusion to work and we can obtain complementary benefits from combining both shallow and cold fusion.

For future work, we plan to investigate ways to close the gap between cold fusion and shallow fusion, compare our first pass fusion methods with second pass LM rescoring, and further leverage the flexibility of NNLM swapping in cold fusion and apply it to on-the-fly domain adaptation.

6. REFERENCES

[1] A. Graves, "Sequence transduction with recurrent neural networks," in *ICML Representation Learning Workshop*, 2012.

[2] R. Prabhavalkar, K. Rao, T. Sainath, B. Li, L. Johnson, and N. Jaitly, "A Comparison of Sequence-to-Sequence Models for Speech Recognition," in *Interspeech*, 2017, pp. 939–943.

[3] 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*. IEEE, 2017, pp. 206–213.

[4] Yanzhang He, Tara N Sainath, Rohit Prabhavalkar, Ian McGraw, Raziel Alvarez, Ding Zhao, David Rybach, Anjuli Kannan, Yonghui Wu, Ruoming Pang, et al., "Streaming end-to-end speech recognition for mobile devices," in *ICASSP*. IEEE, 2019, pp. 6381–6385.

[5] Jinyu Li, Rui Zhao, Hu Hu, and Yifan Gong, "Improving rnn transducer modeling for end-to-end speech recognition," in *ASRU*. IEEE, 2019, pp. 114–121.

[6] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NeurIPS*, 2015.

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

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

[9] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint ctc-attention based end-to-end speech recognition using multi-task learning," in *ICASSP*. IEEE, 2017.

[10] C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina, N. Jaitly, B. Li, J. Chorowski, and M. Bacchiani, "State-of-the-Art Speech Recognition with Sequence-to-Sequence Models," in *ICASSP*, 2018.

[11] Eugene Weinstein, James Apfel, Mohammadreza Ghodsi, Rodrigo Cabrera, and Xiaofeng Liu, "Rnn-transducer with stateless prediction network," in *ICASSP*, 2020, pp. 7049–7053.

[12] Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio, "On using monolingual corpora in neural machine translation," *arXiv preprint arXiv:1503.03535*, 2015.

[13] Anjuli Kannan, Yonghui Wu, Patrick Nguyen, Tara N Sainath, Zhijeng Chen, and Rohit Prabhavalkar, "An analysis of incorporating an external language model into a sequence-to-sequence model," in *ICASSP*. IEEE, 2018, pp. 1–5828.

[14] Anuroop Sriram, Heewoo Jun, Sanjeev Satheesh, and Adam Coates, "Cold fusion: Training seq2seq models together with language models," *arXiv preprint arXiv:1708.06426*, 2017.

[15] Changhao Shan, Chao Weng, Guangsen Wang, Dan Su, Min Luo, Dong Yu, and Lei Xie, "Component fusion: Learning replaceable language model component for end-to-end speech recognition system," in *ICASSP*. IEEE, 2019, pp. 5361–5635.

[16] 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 *SLT*. IEEE, 2018, pp. 369–375.

[17] Eric McDermott, Hasim Sak, and Ehsan Variani, "A density ratio approach to language model fusion in end-to-end automatic speech recognition," in *ASRU*. IEEE, 2019.

[18] Ehsan Variani, David Rybach, Cyril Allauzen, and Michael Riley, "Hybrid autoregressive transducer (hat)," in *ICASSP*. IEEE, 2020, pp. 6139–6143.

[19] Suyoun Kim and Michael L Seltzer, "Towards language-universal end-to-end speech recognition," in *ICASSP*. IEEE, 2018.

[20] Chung-Cheng Chiu, Arun Narayanan, Wei Han, Rohit Prabhavalkar, Yu Zhang, Navdeep Jaitly, Ruoming Pang, Tara N. Sainath, Patrick Nguyen, Liangliang Cao, and Yonghui Wu, "Rnn-t models fail to generalize to out-of-domain audio: Causes and solutions," in *Interspeech*, 2020.

[21] Qian Zhang, Han Lu, Hasim Sak, Anshuman Tripathi, Erik McDermott, Stephen Koo, and Shankar Kumar, "Transformer transducer: A streamable speech recognition model with transformer encoders and rnn-t loss," in *ICASSP*. IEEE, 2020, pp. 7829–7833.

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

[23] 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," *arXiv preprint arXiv:1904.08779*, 2019.

[24] Tom Ko, Vijayaditya Peddinti, Daniel Povey, and Sanjeev Khudanpur, "Audio augmentation for speech recognition," in *Interspeech*, 2015.

[25] T. Kudo, "Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates," in *ACL*, 2018.

[26] Taku Kudo and John Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," *arXiv preprint arXiv:1808.06226*, 2018.

[27] Yu Zhang, Guoguo Chen, Dong Yu, Kaisheng Yaco, Sanjeev Khudanpur, and James Glass, "Highway long short-term memory RNNs for distant speech recognition," in *ICASSP*, 2016.

[28] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," in *ICLR*, 2015.