RNN-TRANSDUCER WITH STATELESS PREDICTION NETWORK

Mohammadreza Ghodsi, Xiaofeng Liu, James Apfel, Rodrigo Cabrera, and Eugene Weinstein

{ghodsi,xiaofengliu,japfel,rodrigocabrera,weinstein}@google.com
Google

ABSTRACT

The RNN-Transducer (RNNT) outperforms classic Automatic Speech Recognition (ASR) systems when a large amount of supervised training data is available. For low-resource languages, the RNNT models overfit, and can not directly take advantage of additional large text corpora as in classic ASR systems.

We focus on the prediction network of the RNNT, since it is believed to be analogous to the Language Model (LM) in the classic ASR systems. We pre-train the prediction network with text-only data, which is not helpful. Moreover, removing the recurrent layers from the prediction network, which makes the prediction network stateless, performs virtually as well as the original RNNT model, when using wordpieces. The stateless prediction network does not depend on the previous output symbols, except the last one. Therefore it simplifies the RNNT architectures and the inference.

Our results suggest that the RNNT prediction network does not function as the LM in classical ASR. Instead, it merely helps the model align to the input audio, while the RNNT encoder and joint networks capture both the acoustic and the linguistic information.

***Index Terms*—** ASR, RNN-Transducer, RNNT, Prediction Network, Stateless

1. INTRODUCTION

Traditionally, Automatic Speech Recognition (ASR) systems were constructed by joining multiple models, including the Acoustic Model (AM), the pronunciation model (lexicon), and the Language Model (LM), which are trained independently. The AM is trained with speech-text (paired) data, and the LM is trained with text-only data.

Recently End-to-End (E2E) ASR models such as the RNN-Transducer (RNNT) [1, 2, 3] have become very popular. Compared to classic ASR systems, they directly predict word sequences (using graphemes or wordpieces) from audio features. Therefore, they simplify the ASR system, and have shown better performance in many cases. However, classic ASR systems benefit from the linguistic information of text-only data, in the form of an LM, whereas E2E models are usually trained only with speech-text data without taking advantage of text-only data (which is typically available in much larger quantities). Therefore, E2E models may not perform equally well on long-tail examples (e.g., uncommon words or proper nouns), or on low-resource languages.

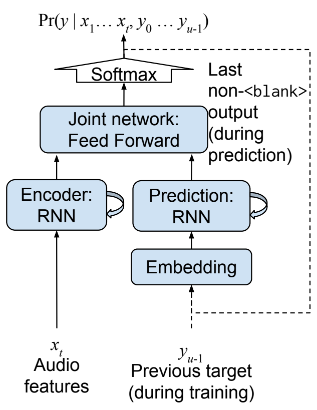
Many recent works have investigated effective ways of using text-only data to improve E2E models, including fusion with an external LM [4, 5, 6, 7, 8], knowledge distillation from an LM [9], augmenting paired data with TTS data [10, 11], etc. Additionally, the encoder-prediction structure of RNNT models has been understood to resemble the AM-LM structure in the classic ASR systems [2]. That is, the encoder network models the acoustic features of speech, similar to an AM, while the recurrent prediction networks memorizes the long term label dependencies, and models the linguistic features, similar to an LM. If this is the case, it may be possible to enhance the recurrent prediction network, by training it with text-only data, and thus improve the overall performance of the RNNT model. Such approaches include pre-training the prediction network with text-only data [12], and multi-task training, where the prediction network is trained with text-only data in addition to paired speech-text data. However, while pre-training the *encoder* with extra data usually helps, pre-training the prediction network with text data has not been helpful in our experiments.

In this paper, we show evidence that contradicts the widely accepted understanding that the prediction network functions like an LM. Our experiments on multiple languages show that a stateless (i.e. non-recurrent) prediction network, that depends only on the last output symbol, works virtually as well as the original RNNT architecture, using wordpieces. This simplifies the RNNT architectures and inference also. These results suggest that the RNNT encoder and joint networks can capture both acoustic and linguistic aspects, while the prediction network merely helps the model choose between outputting an actual symbol or blank.

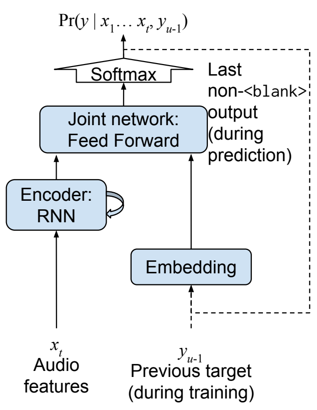
2. METHODS

2.1. The RNN Transducer

The RNNT is a streaming [13] E2E model, as illustrated in Fig. 1a. It typically consists of the following components: the encoder, the prediction network, and the joint network. Its

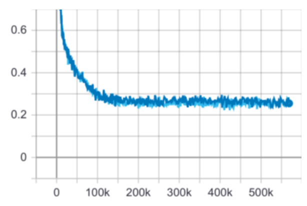

$\Pr(y \mid x_1 \ldots x_t, y_0 \ldots y_{u-1})$
Softmax
Last non-<blank> output (during prediction)
Joint network: Feed Forward
Encoder: RNN
Prediction: RNN
Embedding
x_t Audio features
y_{u-1} Previous target (during training)

(a) Baseline RNNT

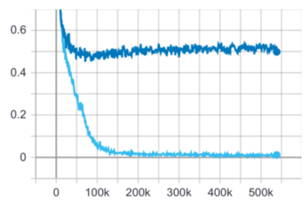

$\Pr(y \mid x_1 \ldots x_t, y_{u-1})$
Softmax
Last non-<blank> output (during prediction)
Joint network: Feed Forward
Encoder: RNN
Embedding
x_t Audio features
y_{u-1} Previous target (during training)

(b) RNNT-SLP

Fig. 1: Representation of the baseline RNNT model and the RNNT with stateless prediction (SLP) network.



(a) 39.3 million utterances



(b) 393 thousand utterances

Fig. 2: RNNT training performance on En-US with (a) the full US English training set, and (b) 1% random sample of the full set. The x-axis is the number of training steps, and the y-axis is WER. The performance on train set and dev set are plotted in cyan and blue respectively.

input is a sequence of audio feature vectors that correspond to segments of the audio (frames), represented by x_t. Its target is a sequence of sub-word symbols, represented by y_u, which can be graphemes (characters) or wordpieces [14].

The length of the output sequence is usually shorter than the input, so the model is also allowed to predict a special blank symbol (<b>), which skips this input frame without changing the state of the prediction network. At a high level, the model predicts the probability of the next symbol or blank $\hat{y} \in \mathcal{Y} \cup \{\text{<b>}\}$ at any given time t, given the audio features up to x_t, and previous *non-blank* symbols $y_0 \ldots y_{u-1}$:

$$\Pr\left(\hat{y} \mid x_1 \ldots x_t, y_0 \ldots y_{u-1}\right) \quad (1)$$

We refer readers to [1, 15] for more detail.

2.2. Pre-training the prediction network

The RNNT's recurrent prediction network is believed to act like an LM, capturing the linguistic dependencies between symbols. Therefore, we try pre-training the prediction network using text only data. That is, we initialize the parameters of the RNNT prediction network (the embedding and the recurrent layers), from an RNN-LM trained on the text-only data, then we continue training the RNNT model on paired data.

2.3. Studying the RNNT behavior

To simulate a low-resource language and better understand the RNNT behavior, we trained two identical RNNT models: the first is trained on all available En-US data, and the second is trained on a 1% random sample of the same data. Fig. 2 shows the Word Error Rate (WER) on train and dev sets for these models. Note that in the 1% experiment, the model exhibits extreme overfitting with high WER. Given this issue, we investigate how much data is required to train each part of the RNNT, and their relative contributions to the overall accuracy. We do this by building on the small data simulation experiment: we continue training different parts of the overfit model using all of the data, and measure the improvements.

2.4. RNNT with stateless prediction network (RNNT-SLP)

Our previous experiments suggest that the RNNT encoder is more significant than the prediction network. This prompts a closer look at the prediction network, and the role it plays.

To investigate how much context the prediction network requires, we remove the recurrent layers from the prediction network. The resulting RNNT has no hidden state in the prediction network (i.e. it is "stateless"). We use the term RNNT with StateLess Prediction network (RNNT-SLP) to refer to this model. Its prediction is conditioned only on the *last* (output or target) symbol, acting effectively as a 2-gram LM on the output subword set:

$$\Pr\left(\hat{y} \mid x_1 \ldots x_t, y_{u-1}\right) \quad (2)$$

We compare RNNT-SLP with the original RNNT model (Equation 1) and an end-to-end Connectionist Temporal Classification (CTC) model that has no dependence on previous symbols:

$$\Pr\left(\hat{y} \mid x_1 \ldots x_t\right) \quad (3)$$

3. RESULTS

3.1. Data sets and model architecture details

Our data preparation, and our baseline RNNT model architecture are generally the same as [13]. Here we recount only the most relevant information.

We have chosen 3 languages that represent high, medium, and low amounts of training data: English (US) has 39.3 million utterances (∼ 30,000 hours) of training data, Spanish (Spain) has 13.4 million utterances (∼ 11,000 hours), and Norwegian has 3.1 million utterances (∼ 2,000 hours). The

Table 1: The RNNT prediction network pre-training results.

Experiment	Test WER
Baseline	23.5
Freeze nothing	23.9 (+1.7%)
Freeze prediction network	55.1 (+134%)
Add two layers, freeze two	26.0 (+10.6%)

utterances are anonymized and transcribed, and are representative of Google's voice-search traffic. The training set is created by artificially corrupting clean utterances using a room simulator. The dev set is a small fraction of the training set held out for validation. The test set also contains anonymous transcribed utterances from the voice-search task.

Our baseline RNNT architecture has an encoder network that consists of 8 LSTM layers, where each layer has 2,048 hidden units followed by a 640-dimensional projection layer. The prediction network consists of an embedding layer followed by 2 LSTM/projection layers, and the joint network is a feed-forward layer with 640 hidden units. The total size of RNNT model is 120M parameters for wordpiece models. For output symbols, we use either 4096 wordpieces, or less than 100 graphemes.

3.2. Pre-training the prediction network

In pretraining, a RNN-LM with identical architecture to the RNNT prediction network was trained using a mix of: the transcripts of the paired training data, public web pages, and typed search query data. We tried three variations of pre-training. In the first experiment, the RNNT is trained without freezing any parameters. In the second experiment we freeze the prediction network and train only the encoder and the joint network. The third experiment is an intermediate approach: we add two extra recurrent layers to the prediction network, initialize the bottom two from the RNN-LM and freeze just the bottom two layers. The results of prediction network pre-training are shown in Table 1. Pre-training was not found to help in any of these experiments. The best result is when the prediction network is not frozen. This suggests that what the RNN-LM learns from the text-only data is not useful for the RNNT model, and is best "forgotten".

3.3. The relative importance of the components of the the RNNT model

To measure how much data is needed to train each component of the the RNNT model, we set up the following experiment. We first train the baseline RNNT architecture on a 1% sample of English data. The WER on for this model is shown in the first row of Table 2. In the following experiments, we initialize all parameters of the RNNT model from the 1% experiment, then freeze some components of the RNNT model, and continue training the rest on 100% of En-US data. This means that the frozen parameters have been trained on a very small amount of the data, while the rest are trained on all of the data.

Table 2: The result of training different components of the RNNT model on additional data. We first initialize the model parameters from a model trained on 1% of supervised English data. Then freeze some parameters, and continue training others on 100% of the data. Here PN refers to the prediction network, and JN refers to the joint network.

Experiment	Dev WER
Base model trained on 1% training data	32
Train PN and JN	24
Train PN, JN and top layer of encoder	20
Train encoder	17
Train PN, JN and top 4 layers of encoder	15
Train encoder and JN	15
Train all parameters	13

The most important insight from Table 2 comes from comparing the WER after tuning the prediction and joint network (24%) to WER after tuning only the encoder (17%). Using additional data to tune the encoder yields more gains. Additionally, tuning both the encoder and joint network is almost as good as tuning all parameters, which suggests that the prediction network has a relatively small role.

3.4. Stateless prediction network

The effect of using a stateless (non-recurrent) prediction network depends on the type of model output symbols (that is, graphemes vs wordpieces). Stateless prediction network causes significant regressions for the grapheme models, but the errors mainly fit a particular patter illustrated in Table 3. Namely, for transcripts that contain repeated graphemes (like 'food'), the model predicts any number of those graphemes with almost the same probability (that is, anything matching the regular expression 'foo*d'). This is because the model sees only one letter into the past, and does not "remember" whether it has output one or two 'o's. So it can continue repeating the same symbol many times. Therefore, an important job of the prediction network is that it stops the model from outputting repetitive symbols many times.

The repetition problem is less significant for wordpiece models, for two reasons. Firstly, because of the way wordpieces are constructed, the common repetitive patterns (for example the ones that have linguistic significance, like 'oo' and 'ee') are represented by distinct wordpieces. Secondly, there are many more wordpieces than graphemes, so the probability of two wordpieces that are the same appearing next to each other is lower.

In fact, we get little or no WER regression from using a stateless prediction network with wordpieces, as shown in Ta-

Table 3: Example errors of a grapheme RNNT model with stateless prediction network. These are most likely hypotheses for a single utterance.

kent island seafood
kent island seafod
kent island seafoood
kent island seafooood
cant island seafood
cant island seafod

ble 4. For small and medium size languages, there is no noticeable difference between the stateless prediction network and the baseline. We did observe a regression on our largest language (English). However, if we remove the same number of recurrent layers from encoder instead of prediction network, we observe the same regression. And, with the stateless prediction network, if we add two more recurrent layers in the *encoder*, we recover some of the above regressions. These experiments suggest that the total number of the parameters is more important than whether the recurrent layers are in the encoder or the prediction network.

We also compared the RNNT-SLP approach to a CTC wordpiece model, to study the effect of auto-regression in the prediction network. Table 5 compares a wordpiece CTC model and a RNNT-SLP model trained on the same (Spanish) training data. The CTC model consists of 8 bidirectional (2×640 unit) LSTM layers, which have the same numbers of hidden units and projection units as the RNNT-SLP model, except that the RNNT encoder uses unidirectional LSTM layers. Thus the CTC model has similar number of parameters as the RNNT-SLP model. The CTC model is decoded using beam-search without an LM.

Table 5 shows that the dev set WER is much worse for the CTC model. This suggests that the independence assumption between all symbols (that the CTC model makes) is too strong. In practical terms, the RNNT-SLP models a 2-gram LM (due to the dependence on the last output), whereas the CTC model lacks any LM information.

4. DISCUSSION

The key insight from this investigation is that the RNNT prediction network is not directly comparable to the LM in the classical ASR system. Its most important function seems to be preventing repeated outputs of the same symbol. To do that, the prediction network needs to drive the blank probability of the network, which a conventionally trained RNN-LM cannot do.

We also compared three sequence-to-sequence architectures with different amounts of dependence on their previous output symbols: Our experiments show that the RNNT-SLP is competitive with the baseline RNNT for wordpieces, and

Table 4: Comparing baseline RNNT model WER, with models with zero recurrent layers in the prediction network. We also include some other variants with by changing the number of of layers in the encoder. These three languages represent relatively small, medium or large amounts of training data. The pair of numbers in the first column denote the number of layers in the (encoder, prediction) network respectively. For example, (8, 0) refers to the stateless prediction network architecture.

(a) Norwegian (Norway), trained on 3.1 million utterances.

Model	Params	Test WER
(8, 2)	120M	22.2
(8, 0)	100M	19.9 (-1.5%)

(b) Spanish (Spain), trained on 13.4 million utterances.

Model	Params	Test WER
(8, 2)	120M	9.6
(6, 2)	96M	9.7 (+1.0%)
(8, 0)	102M	9.7 (+1.0%)

(c) English (US), trained on 39.3 million utterances.

Model	Params	Test WER
(8, 2)	120M	6.8
(6, 2)	96M	7.4 (+8.8%)
(8, 0)	102M	7.2 (+5.9%)
(10, 0)	126M	7.1 (+4.4%)

Table 5: Comparing RNNT-SLP with CTC.

Model	Params	Dev WER
RNNT – stateless PN (8, 0)	102M	13.0
CTC – bidirectional encoder	90M	20.8 (+60.0%)

significantly outperforms CTC. This suggests that it is helpful to depend on the label history, but that at least for low and medium resource languages, the RNNT does not need more than one wordpiece of history to achieve full performance.

The RNNT-SLP has two additional advantages over the regular RNNT: It reduces the total number of model parameters and simplifies the model architecture. And it simplifies the beam-search decoding implementation, since it does not need to keep track of the hidden state of the recurrent layers in the prediction network for each partial hypothesis.

How to leverage text data for RNNT training and how to achieve competitive results in low-resource languages remain topics for further investigation, but direct training of the RNNT model using text data does not seem to be a fruitful approach.

5. REFERENCES

[1] Alex Graves, "Sequence transduction with recurrent neural networks," *arXiv preprint arXiv:1211.3711*, 2012.

[2] Kanishka Rao, Haşim Sak, and Rohit Prabhavalkar, "Exploring architectures, data and units for streaming end-to-end speech recognition with rnn-transducer," in *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. IEEE, 2017, pp. 193–199.

[3] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly, "A comparison of sequence-to-sequence models for speech recognition.," in *Interspeech*, 2017, pp. 939–943.

[4] 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.

[5] 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 *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 1–5828.

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

[7] 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 *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 5361–5635.

[8] 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 *2018 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2018, pp. 369–375.

[9] Ye Bai, Jiangyan Yi, Jianhua Tao, Zhengkun Tian, and Zhengqi Wen, "Learn spelling from teachers: Transferring knowledge from language models to sequence-to-sequence speech recognition," *arXiv preprint arXiv:1907.06017*, 2019.

[10] Takaaki Hori, Ramon Astudillo, Tomoki Hayashi, Yu Zhang, Shinji Watanabe, and Jonathan Le Roux, "Cycle-consistency training for end-to-end speech recognition," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6271–6275.

[11] Shigeki Karita, Shinji Watanabe, Tomoharu Iwata, Marc Delcroix, Atsunori Ogawa, and Tomohiro Nakatani, "Semi-supervised end-to-end speech recognition using text-to-speech and autoencoders," in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6166–6170.

[12] Senmao Wang, Pan Zhou, Wei Chen, Jia Jia, and Lei Xie, "Exploring rnn-transducer for chinese speech recognition," *arXiv preprint arXiv:1811.05097*, 2018.

[13] 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 *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6381–6385.

[14] Mike Schuster and Kaisuke Nakajima, "Japanese and korean voice search," in *2012 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2012, pp. 5149–5152.

[15] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton, "Speech recognition with deep recurrent neural networks," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2013, pp. 6645–6649.