SEQUENCE TRAINING OF ENCODER-DECODER MODEL USING POLICY GRADIENT FOR END-TO-END SPEECH RECOGNITION

Shigeki Karita, Atsunori Ogawa, Marc Delcroix, and Tomohiro Nakatani

NTT Communication Science Laboratories, NTT Corporation, Kyoto, Japan

ABSTRACT

The standard evaluation metric of automatic speech recognition (ASR) is the word error rate (WER), which measures the dissimilarity between recognized word sequences and their ground truth. Many training algorithms designed to reduce sequence-level errors such as WER have been proposed for hidden Markov model (HMM)-based ASR, e.g., state-level minimum Bayes risk (sMBR). However, these approaches cannot be used directly for encoder-decoder model based end-to-end ASR, because the encoder-decoder model employs very different mechanisms from HMM-based approaches. In this paper, we propose a new method for optimizing the encoder-decoder model based on a sequence-level evaluation metric. Since the WER is not directly differentiable, we adopt a policy gradient objective function to train the encoder-decoder model, which enables us to minimize the expected WER of the model predictions. This training method employs the scoring of multiple hypotheses as in the decoding stage while usual cross entropy training uses only the ground truth. Therefore, we can expect it to improve the decoding results of the encoder-decoder model. We perform experiments using the Tedlium corpus to demonstrate the potential of our proposed method for improving the recognition performance of the encoder-decoder model.

Index Terms— end-to-end speech recognition, encoder-decoder model, policy gradient, sequence training, Tedlium

1. INTRODUCTION

An encoder-decoder model [1] is a widely used framework for transcribing sequences of input signals into sequences of output signals using neural networks (NN). In automatic speech recognition (ASR) tasks, the input is a sequence of speech features and the output is a sequence of text transcriptions [2]. Unlike a deep NN (DNN)-hidden Markov model (HMM) hybrid approach [3], the encoder-decoder model requires neither lexicons (e.g., a pronunciation dictionary) nor predefined alignments between acoustic frames and the transcription to be trained. The advantage of the encoder-decoder model is that it can directly learn mapping from speech to text. This property enables the end-to-end joint optimization of acoustic and language modeling [4]. However, there are still some limitations in the training scheme that cause discrepancies between the training and the evaluation stages.

There are two problems in the conventional cross-entropy training of an encoder-decoder model. The first is the lack of a common method for optimizing the model based on an actual sequence-level evaluation metric such as character or word error rate (CER/WER) explicitly. In the DNN-HMM based and connectionist temporal classification (CTC) based approaches [5], sequence discriminative training methods have been proposed for optimizing such a metric [6]. For example, the aim of state-level minimum Bayes risk (sMBR) training [7] is to explicitly maximize the expectation of word accuracy. However, it is difficult to apply sMBR to encoder-decoder models because the size of lattices composed with these models grows exponentially with the lengths of the hypotheses. This is the same problem faced by decoding based on recurrent NN language models (RNNLM) [8].

The second problem with encoder-decoder models is the gap between the prefix tokens used in the conventional training and evaluation stages. Here token are characters, numbers, corpus specific tags (e.g., <NOISE>) and decoder's control tags (e.g., start-of-sequence <SOS> and end-of-sequence <EOS>). The main difference between the training and evaluation stages is a conditional probability provided by the encoder-decoder model [9]. During training, the decoder is conditioned on ground-truth prefix tokens while hypothesized ones are used during evaluation because the ground-truth tokens are unavailable. We anticipate that, by minimizing this gap in the training state, we will be able to reduce the CER/WER in the evaluation stage.

To address the two problems described above, we propose a new training approach that uses a policy gradient [10] to directly optimize expected the CER and WER. Moreover, during this training, the prediction is concatenated to the past predictions as a prefix, which is relevant to the decoding algorithm. The policy gradient method is widely used in reinforcement learning. Recently, state-of-the-art machine translation systems use policy gradient method to optimize the BLEU evaluation metric [11]. In this paper, we explore new definitions of the policy gradient objective function that consists of a partial CER/WER, which is more informative than the constant CER/WER when training an encoder-decoder for ASR.

The contribution of this paper is summarized as follows: 1) We introduce the policy gradient method to optimize the sequence-level metric CER/WER for encoder-decoder model based ASR. 2) We investigate new fine-grained errors using the dynamic programming matrix of WER computation. 3) We demonstrate the improvement obtained by these two ideas using the Tedlium ASR task.

2. ENCODER-DECODER MODEL

In this section, we describe the basic framework of the encoder-decoder model based ASR.

2.1. Architecture for automatic speech recognition

An encoder-decoder model consists of two parts called "encoder" and "decoder" networks [1]. The encoder receives an utterance consisting of a sequence of features $\mathbf{x} \in \mathbb{R}^{F \times U}$, where F is a feature dimension and U is the number of frames, and transforms it to an intermediate representation $\mathbf{e} \in \mathbb{R}^{2 \times Z \times U'}$ that consists of bidirectional Z units of U' subsampled frames. Then, the decoder network predicts a current token y_t from a vocabulary set $\mathbb{Y} = \{\text{'a'}, \text{'b'}, \ldots, \text{<EOS>}\}$ from the encoder's output $\mathbf{e}$, the decoder's state $\mathbf{s}_t$ and the embedded vector of the previous token y_{t-1}. We describe this processing pipeline as follows:

$$\mathbf{e} = \text{Encoder}(\mathbf{x}), \tag{1}$$

$$y_0 = \text{<SOS>}, \tag{2}$$

$$\mathbf{s}_0 = \mathbf{0}, \tag{3}$$

$$[\Pr(y_t|\mathbf{x}, \mathbf{y}_{1:t-1}), \mathbf{s}_t] = \text{Decoder}(y_{t-1}, \mathbf{s}_{t-1}, \mathbf{e}), \tag{4}$$

where <SOS> is the start of a sequence token. The architectures of the encoder and decoder are similar to the model proposed in [4] (i.e., an encoder consists of three bidirectional LSTM layers and a decoder consists of one LSTM layer and location based attention mechanism). Note that, the decoder emits an <EOS> token when it predicts the end of a sequence.

2.2. Gaps between training and decoding

There are two gaps between the training and decoding stages. First, the models are trained to optimize the "frame-level" cross-entropy between ground-truth and predicted tokens, whereas decoding results are usually evaluated in terms of a sequence level metric such as CER or WER. Second, the prediction of the current token shown in Eq. (4) is accomplished with ground-truth tokens during training while it is realized with a hypothesis during decoding.

We illustrate the training and decoding procedures for encoder-decoder models in Figs. 1 and 2, respectively. During training, we feed prefix ground-truth tokens $\mathbf{y}_{1:t-1}$ to the decoder network since the cross-entropy is the negative log-likelihoods of the ground-truth tokens given the input utterance $\mathbf{x}$ as follows:

$$\text{J}_{\text{CE}} = -\log \Pr(\mathbf{y}_{1:K}|\mathbf{x}) = -\sum_{t=1}^{K} \log \Pr(y_t|\mathbf{x}, \mathbf{y}_{1:t-1}) \tag{5}$$

where $\mathbf{y}_{1:K}$ is a ground-truth token sequence.

During decoding, the encoder-decoder model requires a prefix search [1], [4], [12] to find the best hypothesis transcription for the input utterance $\mathbf{x}$ because the ground-truth tokens are unavailable in the evaluation set.

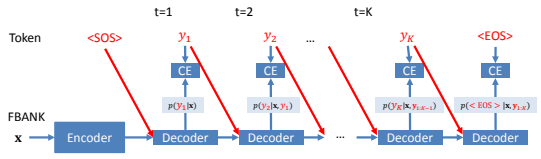

Fig. 1. Training stage. A ground-truth sequence is used as the prefix tokens $\mathbf{y}_{1:t-1}$ to compute y_t.

3. POLICY GRADIENT BASED TRAINING

In this section, we define a new objective function to mitigate the gap between training and decoding as discussed in the previous section. To implement the objective function, we introduce a sampling method for training that is similar to the prefix search used during decoding and explore new error definitions.

3.1. Expected CER/WER objective function

As regards the search problem, a policy gradient [10] is a widely used method for directly minimizing the expectation of an undifferentiable error L. For example, a simple definition of $\text{L}(\tilde{\mathbf{y}}, \mathbf{y})$ for ASR is a CER/WER between a hypothesis $\tilde{\mathbf{y}}$ and ground-truth transcription $\mathbf{y}$ corresponding to an input utterance $\mathbf{x}$. The policy gradient based objective function and its gradient are approximated as follows [13]:

$$\text{J}_{\text{PG}}(\Theta) = \text{E}_{\tilde{\mathbf{y}} \sim \Pr_\Theta(\tilde{\mathbf{y}}|\mathbf{x})}[\text{L}(\tilde{\mathbf{y}}, \mathbf{y})] \approx \frac{1}{M} \sum_{m=1}^{M} \text{L}(\tilde{\mathbf{y}}^{(m)}, \mathbf{y}), \tag{6}$$

$$\frac{\partial}{\partial \Theta} \text{J}_{\text{PG}}(\Theta) \approx \frac{1}{M} \sum_{m=1}^{M} \text{L}(\tilde{\mathbf{y}}^{(m)}, \mathbf{y}) \frac{\partial}{\partial \Theta} \log \Pr_\Theta(\mathbf{y} = \tilde{\mathbf{y}}^{(m)}|\mathbf{x}), \tag{7}$$

where M is the number of samples, $\Pr_\Theta(\tilde{\mathbf{y}}|\mathbf{x})$ is the likelihood of the hypothesis $\tilde{\mathbf{y}}$ that the ASR model with its parameter Θ predicts from the observed input utterance $\mathbf{x}$, $\mathbf{y}$ and $\tilde{\mathbf{y}}^m$ are ground truth and m th sampled hypothesis transcriptions from $\Pr_\Theta(\tilde{\mathbf{y}}|\mathbf{x})$, respectively.

Except for the details of the error L and the sampling method used on $\Pr_\Theta(\tilde{\mathbf{y}}|\mathbf{x})$ for encoder-decoder model based ASR, our framework is very similar to edit-distance minimum Bayes risk training for CTC based ASR [14].

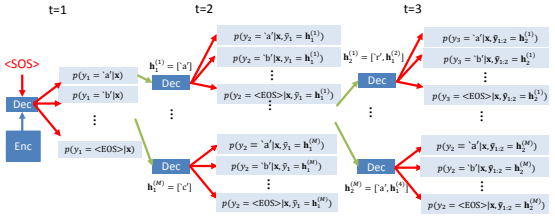

Fig. 2. Decoding stage. The best sequences of prefix tokens $\tilde{\mathbf{y}}_{1:t-1}$ are beam-searched to compute $\tilde{y}_t$.

3.2. Sampling method for encoder-decoder models

Here we describe how to sample hypothesis tokens $\tilde{\mathbf{y}}$ from an encoder-decoder's output distribution $\Pr_\Theta(\tilde{\mathbf{y}}|\mathbf{x})$ in Eq. (6). However, it is difficult to apply the existing algorithm used on the CTC system [14] to the encoder-decoder because the size of the lattices composed with these models grows exponentially with the length of the hypotheses [8]. Hence we adopt a similar approach to prefix-search decoding as seen in Fig. 2 for the sampling. They resemble each other in the context of hypothesis space approximation except that we conduct the sampling during this training, whereas we do sorting during the prefix-search decoding.

First, we draw initial M tokens as hypotheses $\tilde{y}_1^{(m)}, (m = 1, \ldots, M)$ from the encoder-decoder's posterior as follows:

$$\left[\Pr(y_1|\mathbf{x}), \mathbf{s}_1^{(m)}\right] = \text{Decoder}(\text{<SOS>}, \mathbf{0}, \mathbf{e}), \tag{8}$$

$$\tilde{\mathbf{h}}_1^{(m)} = \left[\tilde{y}_1^{(m)}\right] \sim \text{Multi}_M \Pr(y_1|\mathbf{x}), \tag{9}$$

$$\Pr(\tilde{\mathbf{h}}_1^{(m)}|\mathbf{x}) = \Pr(y_1|\mathbf{x}). \tag{10}$$

where $\tilde{\mathbf{h}}_{t-1}^{(m)}$ is a sequence of $t-1$ prefix tokens in the m th sampled hypothesis at time t. Note that $\tilde{\mathbf{h}}_t^{(m)}$ would be a pair consisting of the next token $\tilde{y}_t^{(m)}$ and the sampled predecessor tokens $\tilde{\mathbf{h}}_{t-1}^{(l)}$, where the coupled sample-ids m and l are not always theA same because all the samples indexed with $m \in \{1, 2, \ldots, M\}$ are sampled together as discussed below.

At succeeding time steps $t \geq 2$, we compute all the probabilities of the current token $y_t \in \mathbb{Y} = \{\text{'a'}, \text{'b'}, \ldots, \text{<EOS>}\}$ from the previous sampled token $\tilde{y}_{t-1}^{(m)}$ and the decoder's state $\mathbf{s}_t^{(m)}$ corresponding to the sampled prefix $\tilde{\mathbf{h}}_t^{(m)}$ as follows:

$$\left[\Pr(y_t|\mathbf{x}, \tilde{\mathbf{h}}_t^{(m)}), \mathbf{s}_t^{(m)}\right] = \text{Decoder}(y_{t-1}^{(m)}, \mathbf{s}_{t-1}^{(m)}, \mathbf{e}), \tag{11}$$

$$\Pr(y_t, \tilde{\mathbf{h}}_t^{(m)}|\mathbf{x}) = \Pr(y_t|\mathbf{x}, \tilde{\mathbf{h}}_t^{(m)}) \Pr(\tilde{\mathbf{h}}_t^{(m)}|\mathbf{x}), \tag{12}$$

$$\pi_{m,y_t} = \frac{\Pr(y_t, \tilde{\mathbf{h}}_t^{(m)}|\mathbf{x})}{\sum_{m=1}^{M} \sum_{y_t \in \mathbb{Y}} \Pr(y_t, \tilde{\mathbf{h}}_t^{(m)}|\mathbf{x})}. \tag{13}$$

From this multinomial distribution parameter π with $M \times |\mathbb{Y}|$ bins, we draw the next M samples

$$\tilde{\mathbf{h}}_{t+1}^{(m)} = \left[\tilde{y}_t^{(m)}, \tilde{\mathbf{h}}_t^{(l)}\right] \sim \text{Multi}_M \left\{\pi_{m,y_t}\right\}, \tag{14}$$

$$\mathbf{s}_t^{(m)} = \mathbf{s}_t^{(l)}, (l, m \in \{1, 2, \ldots, M\}), \tag{15}$$

$$\Pr(\tilde{\mathbf{h}}_{t+1}^{(m)}|\mathbf{x}) = \Pr(y_t = \tilde{y}_t^{(m)}|\mathbf{x}, \tilde{\mathbf{h}}_t^{(k)}) \Pr(\tilde{\mathbf{h}}_t^{(k)}|\mathbf{x}). \tag{16}$$

We repeat this procedure in Eqs. (12) – (16) until all the hypotheses $\tilde{\mathbf{h}}^{(m)}$ end with <EOS>. Finally, we obtain the normalized likelihoods of the <EOS> ended samples in Eq. (13) as probabilities in Eq. 7.

3.3. Edit-distance based error

In this paper, we propose two different definitions of the error L in Eq. 6. One simply uses the CER/WER between the hypothesis and ground-truth transcriptions as a "constant" error L. The other exploits the dynamic programming matrix used in a CER/WER computation to obtain fine-grained "partial" errors $L_t, (t = 1, 2, \ldots, T)$, where t is a time step of the hypothesis.

3.3.1. Constant error

We call $C_{T,K}^{(\tilde{\mathbf{y}},\mathbf{y})}$ the edit distance between a hypothesis $\tilde{\mathbf{y}}_{1:T}$ and a ground truth. $\mathbf{y}_{1:K}$ to $C_{T,K}^{(\tilde{\mathbf{y}},\mathbf{y})}$ can be obtained from a dynamic programming matrix $\mathbf{C}^{(\tilde{\mathbf{y}},\mathbf{y})} \in \mathbb{R}^{(T+1)\times(K+1)}$ as follows:

$$C_{t,0}^{(\tilde{\mathbf{y}},\mathbf{y})} = t, (t = 0, 1, \ldots, T), \tag{17}$$

$$C_{0,k}^{(\tilde{\mathbf{y}},\mathbf{y})} = j, (k = 0, 1, \ldots, K), \tag{18}$$

$$C_{t,k}^{(\tilde{\mathbf{y}},\mathbf{y})} = \min(C_{t-1,k}^{(\tilde{\mathbf{y}},\mathbf{y})} + 1, C_{t,k-1}^{(\tilde{\mathbf{y}},\mathbf{y})} + 1, C_{t-1,k-1}^{(\tilde{\mathbf{y}},\mathbf{y})} + \delta_{t,k}) \quad (t = 1, 2, \ldots, T, k = 1, 2, \ldots, K), \tag{19}$$

$$\delta_{t,k} = \begin{cases} 0 & \text{if } \tilde{y}_t = y_k \\ 2 & \text{otherwise.} \end{cases} \tag{20}$$

The constant error simply uses this edit-distance-based error rate $\mathrm{L}(\tilde{\mathbf{y}}, \mathbf{y}) = C_{T,K}^{(\tilde{\mathbf{y}},\mathbf{y})}/K$ for every step of the encoder-decoder's output $\tilde{y}_t$ uniformly as seen in related work [14]. Note that, we implement this function not only for character tokens but also for word tokens by grouping a sequence of characters by the space tokens. We refer to these errors as "constant" CER and WER, respectively.

3.3.2. Fine-grained partial error

Using the constant error only provides a constant loss and its gradient w.r.t. the encoder-decoder's parameters at each time step t. We expect that a fine-grained error L_t at each token in the hypothesis $\tilde{y}_t^{(m)} (t = 1, \ldots, T)$ and objective function would be better for training the encoder-decoder model because they contain more informative training signals at each step as discussed in [9]. With such a fine-grained error, the objective function becomes

$$\mathrm{J}_{\mathrm{FPG}}(\Theta) \approx \frac{1}{M} \sum_{m=1}^{M} \sum_{t=1}^{T} \mathrm{L}_t(\tilde{\mathbf{y}}^{(m)}, \mathbf{y}). \tag{21}$$

Considering the minimum path $\hat{\mathbb{C}}^{(\tilde{\mathbf{y}},\mathbf{y})} = \{C_{T,K}^{(\tilde{\mathbf{y}},\mathbf{y})}, \ldots, C_{0,0}^{(\tilde{\mathbf{y}},\mathbf{y})}\}$ from $C_{T,K}^{(\tilde{\mathbf{y}},\mathbf{y})}$ to $C_{0,0}^{(\tilde{\mathbf{y}},\mathbf{y})}$ that is obtained by backtracking the minimum selection in Eq. (19) on $\mathbf{C}^{(\tilde{\mathbf{y}},\mathbf{y})}$, we define our fine-grained error L_t as follows:

$$\hat{k}(t) = \text{argmin}_k \left\{ C_{t,k}^{(\tilde{\mathbf{y}},\mathbf{y})} \middle| C_{t,k}^{\tilde{\mathbf{y}},\mathbf{y}} \in \hat{\mathbb{C}}^{(\tilde{\mathbf{y}},\mathbf{y})} \right\}, \tag{22}$$

$$\mathrm{L}_t(\tilde{\mathbf{y}}, \mathbf{y}) = C_{t,\hat{k}(t)}{}^{(\tilde{\mathbf{y}},\mathbf{y})}/\hat{k}(t), \tag{23}$$

where $\hat{k}(t)$ is the length of the ground truth at the minimum partial error of $C_{t,k}^{\tilde{\mathbf{y}},\mathbf{y}}$. We call this error L_t "partial" CER/WER.

4. RELATED WORK

The policy gradient has been an effective way to optimize searching problems. For example, the policy gradient is the state-of-the-art technique for the image area selection known as a hard attention mechanism for image caption generation [15], for board evaluation and move selection in go [16] and for direct optimization of the expected BLEU in machine translation [11], [17].

This paper is the first result for the policy gradient based optimization of the expected CER/WER in encoder-decoder model based ASR, but two papers have similar concepts to our proposed methods. In [14], a CTC based acoustic model trained with a similar policy gradient framework to optimize expected WER for ASR described in Sec. 3.1 has shown better results than the one trained with sMBR. For an encoder-decoder model based system, the authors of [9] use edit distance in their objective function, which is based on a frame-wise edit-distance regression, to improve decoding behaviors. The motivation behind their proposed "optimistic loss" is the same as that of our fine-grained partial edit-distance approach as discussed in Sec. 3.3.2.

5. EXPERIMENTS

5.1. Settings

To investigate the effect of our proposed method experimentally, we used Tedlium (release 1) corpus [18], which consists of presentation speeches and transcriptions from TED talks, because an improvement in a sequence level training method has been reported [19]. Note that, as we did not use any lexicon or language models, our results cannot be compared directly with the existing results of the acoustic model based systems.

5.1.1. Input feature and target tokens

Tedlium consists of about 135 hours of speech (train 56803, dev 507, test 1155 utterances). We adopted the 40 dimensional FBANK features extracted by Kaldi [20] for input into the encoder network. After feature extraction, we conducted the global normalization of the FBANK coefficients using the mean and standard deviations computed on the training sets. Then, we concatenated the delta and acceleration features.

The target tokens are mainly alphabet letters, space, punctuation and numbers. Tedlium has 50 vocabulary tokens. In addition, there are some special tags (e.g., <NOISE>). All the target sentences start with <SOS> and end with <EOS>. These tokens are converted to 320-dimensional embedded vectors in the first layer of the decoder network.

5.1.2. Initialization and training

We used three BLSTM layers of 320 units in the encoder and one LSTM layer in the decoder based on the setting in [4]. All the weights were initialized randomly as Uniform(-0.1,0.1). First, we optimized our models with cross entropy and Adam [21] with a mini-batch size corresponding to eight utterances.

Table 1. Character and word error rates (%) on Tedlium (135 hours). These results were obtained without any lexicons or language models.

	CER		WER	
	dev	eval	dev	eval
baseline	21.7	21.1	42.3	41.9
J_{PG}-ConstCER	22.6	22.4	43.4	44.0
J_{PG}-ConstWER	22.2	21.9	42.3	42.1
J_{FPG}-PartialCER	21.1	20.5	40.9	40.4
J_{FPG}-PartialWER	**20.9**	**20.0**	**40.3**	**38.6**

The learning rate started from 1e-3 and was halved until 1e-9 when no reduction of WER on the dev set was obtained. We used this model as our baseline. Next, we conducted policy gradient training with the cross entropy trained model as an initial parameter. The learning rate was set at 1e-6 and halved until 1e-9 by monitoring the WERs on the dev set. Convergence took around 4 days. The number of samples used for the policy gradient method in Eq. (7) is $M = 3$.

5.1.3. Decoding and evaluation

We decoded a character sequence from an utterance with a prefix search as described in Section 2.2. Note that, we did not use any language models or lexicons. We used the dev set to select the best models and decoding parameters (beam size and word insertion).

5.2. Results

Table 1 summarizes the CERs and WERs for Tedlium. We observed clear improvements in both CERs and WERs with the fine-grained policy gradient training of partial CER/WER denoted as J_{FPG}-PartialCER and J_{FPG}-PartialWER, respectively, while the policy gradient training of constant CER/WER denoted as J_{PG}-ConstCER and J_{PG}-ConstWER, respectively, degrade from the cross entropy trained baseline. Specifically, the model trained with the policy gradient of the partial WER achieved the best CER of 20.0% and WER of 38.6%. These results support our expectations in Sec. 3.3.2 that our fine-grained partial error in our sampling method assists optimization and reduces the CER/WER.

6. CONCLUSION

In this work, we presented a sequence training framework for encoder-decoder models designed to reduce the CER/WER using the policy gradient method. The experimental results showed the potential of our proposed model. In our future work, we will focus on the variance reduction of the policy gradient as discussed in [11], [14] because our training method requires a very small learning rate and results in slow convergence.

7. REFERENCES

[1] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," *Advances in Neural Information Processing Systems*, pp. 3104–3112, 2014.

[2] Y. Zhang, W. Chan, and N. Jaitly, "Very deep convolutional networks for end-to-end speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4845–4849, 2017.

[3] H. A. Bourlard and N. Morgan, "Connectionist speech recognition," in *International Journal on Pattern Recognition and Artificial Intelligence*, vol. 247, 1994, pp. 647–668.

[4] S. Kim, T. Hori, and S. Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 4835–4839.

[5] A. Graves, A.-r. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2013, pp. 6645–6649.

[6] H. Sak, A. Senior, K. Rao, O. İrsoy, A. Graves, F. Beaufays, and J. Schalkwyk, "Learning acoustic frame labeling for speech recognition with recurrent neural networks," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4280–4284.

[7] M. Gibson and T. Hain, "Hypothesis spaces for minimum bayes risk training in large vocabulary speech recognition," *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, vol. 5, no. 3, pp. 2406–2409, 2006.

[8] T. Hori, Y. Kubo, and A. Nakamura, "Real-time one-pass decoding with recurrent neural network language model for speech recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2014, pp. 6364–6368.

[9] D. Bahdanau, D. Serdyuk, P. Brakel, N. R. Ke, J. Chorowski, A. Courville, and Y. Bengio, "Task loss estimation for sequence prediction," *International Conference on Learning Representation Workshop*, pp. 1–13, 2016.

[10] R. S. Sutton, D. Mcallester, S. Singh, and Y. Mansour, "Policy gradient methods for reinforcement learning with function approximation," *In Advances in Neural Information Processing Systems 12*, pp. 1057–1063, 1999.

[11] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *International Conference on Learning Representation*, pp. 1–17, 2017.

[12] D. Bahdanau, J. Chorowski, D. Serdyuk, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 4945–4949, 2016.

[13] J. Schulman, N. Heess, T. Weber, and P. Abbeel, "Gradient estimation using stochastic computation graphs," in *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds., 2015, pp. 3528–3536.

[14] M. Shannon, "Optimizing expected word error rate via sampling for speech recognition," *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, pp. 3537–3541, 2017.

[15] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: neural image caption generation with visual attention," in *Proceedings of the 32nd International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 37, 2015, pp. 2048–2057.

[16] D. Silver, A. Huang, C. J. Maddison, *et al.*, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.

[17] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *International Conference on Learning Representation*, pp. 1–16, 2016.

[18] A. Rousseau, P. Deléglise, and Y. Estève, "TED-LIUM: An automatic speech recognition dedicated corpus," in *Proceedings of the Eight International Conference on Language Resources and Evaluation*, 2012.

[19] Y. Wang, V. Peddinti, H. Xu, X. Zhang, D. Povey, and S. Khudanpur, "Backstitch: Counteracting finite-sample bias via negative steps," *Proceedings of International Conference on Spoken Language Processing, INTERSPEECH*, pp. 1631–1635, 2017.

[20] D. Povey, A. Ghoshal, G. Boulianne, *et al.*, "The Kaldi speech recognition toolkit," in *IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[21] D. Kingma and J. Ba, "Adam: a method for stochastic optimization," *International Conference on Learning Representations*, pp. 1–13, 2014.