Trainable Dynamic Subsampling for End-to-End Speech Recognition

Citation for published version:
Zhang, S, Loweimi, E, Xu, Y, Bell, P & Renals, S 2019, Trainable Dynamic Subsampling for End-to-End Speech Recognition. in *Proceedings Interspeech 2019.* International Speech Communication Association, pp. 1413-1417, Interspeech 2019, Graz, Austria, 15/09/19. https://doi.org/10.21437/Interspeech.2019-2778

Digital Object Identifier (DOI):
10.21437/Interspeech.2019-2778

Link:
Link to publication record in Edinburgh Research Explorer

Document Version:
Peer reviewed version

Published In:
Proceedings Interspeech 2019

Trainable Dynamic Subsampling for End-to-End Speech Recognition

Shucong Zhang [1], *Erfan Loweimi* [1], *Yumo Xu* [2], *Peter Bell* [1], *Steve Renals* [1]

[1] Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK
[2] Institute for Language, Cognition and Computation, University of Edinburgh, Edinburgh, UK
s1603602@sms.ed.ac.uk, {e.loweimi, yumo.xu, peter.bell, s.renals}@ed.ac.uk

Abstract

Jointly optimised attention-based encoder-decoder models have yielded impressive speech recognition results. The recurrent neural network (RNN) encoder is a key component in such models – it learns the hidden representations of the inputs. However, it is difficult for RNNs to model the long sequences characteristic of speech recognition. To address this, subsampling between stacked recurrent layers of the encoder is commonly employed. This method reduces the length of the input sequence and leads to gains in accuracy. However, static subsampling may both include redundant information and miss relevant information.

We propose using a dynamic subsampling RNN (dsRNN) encoder. Unlike a statically subsampled RNN encoder, the dsRNN encoder can learn to skip redundant frames. Furthermore, the skip ratio may vary at different stages of training, thus allowing the encoder to learn the most relevant information for each epoch. Although the dsRNN is unidirectional, it yields lower phone error rates (PERs) than a bidirectional RNN on TIMIT. The dsRNN encoder has a 16.8% PER on the TIMIT test set, a considerable improvement over static subsampling methods used with unidirectional and bidirectional RNN encoders (23.5% and 20.4% PER respectively).

Index Terms: speech recognition, sequence-to-sequence, attentional encoder-decoder, recurrent neural network

1. Introduction

Recently, sequence-to-sequence models, especially attentional encoder-decoder models, have offered impressive performance in speech recognition [1, 2, 3]. Unlike traditional pipeline models which train each component separately, sequence-to-sequence models optimise all their components jointly and thereby can be trained in an end-to-end fashion.

The recurrent neural network (RNN) encoder is an essential component in attentional encoder-decoder models, and learns the hidden representations of the inputs. However, despite their outstanding performance in modelling sequential data, RNNs have some weaknesses including the vanishing gradient problem, and the difficulty in capturing long-range dependency [4]. Gated architectures, such as the Long Short-Term Memory (LSTM) [5] and the Gated Recurrent Unit (GRU) [6] are designed to alleviate these problems.

Nevertheless, for speech recognition tasks, encoder-decoder models with vanilla gated RNN encoders often yield poor results. This phenomenon is due to the long length of the input sequence [7, 8]. Acoustic input sequences typically consist of hundreds or even thousands of acoustic frames. During training, it is difficult for an RNN encoder to learn to extract the most important information from sequences with such long durations. Chan et al [8] demonstrated that with a vanilla multi-layer LSTM encoder, the model gives inferior results even after an extremely long training time.

A fixed, static subsampling in the RNN encoder has been proposed to alleviate this problem and has led to accuracy improvements [7, 8]. This method is commonly used in encoder-decoder models for speech recognition [1, 2, 3, 7, 8, 9, 10, 11, 12]. In this method, an encoder layer reads either one of every two hidden states of its previous layer (or their concatenation), reducing the sequence length by a factor of 2. However, this method subsamples statically, and does not consider different acoustic situations.

In this work, inspired by the Skip RNN [13], we propose a novel dynamic subsampling RNN (dsRNN) architecture. We use the dsRNN to build the encoder for the encoder-decoder models.

This learned subsampling strategy is helpful in two aspects:

- First the network learns *what to skip* – it learns to skip redundant frames but keeps the important frames. Thus, the skip policy forces the RNN encoder only to consider the most relevant inputs during training.
- Second, it learns *how frequently to skip input frames* – in practice this may be adaptively modified during training. At the start of training, we observe the skip rates to be high, enabling the encoder to learn the hidden representations for the most essential frames at the early training stage. As training progresses, the skip rate decays, allowing the encoder to retain more input information.

Because of these two abilities, the dsRNN encoder learns a better representation of the input sequence and obtains a significant error rate reduction compared to the statically subsampled RNN encoder. Although our dsRNN encoder is unidirectional, we observe increased accuracy compared to a bidirectional statically subsampled RNN encoder.

2. Related Work

RNNs are powerful in processing sequential data, but they have several limitations, including high computation costs, vanishing gradients, and difficulties in capturing long-range dependencies. These problems are severe for speech recognition since the input acoustic frame sequences are long.

For encoder-decoder models for speech recognition, subsampling between stacked encoder RNN layers has been proposed to reduce the length of the input sequence [7, 8]. An encoder layer reads either one of every two hidden states of the previous layer (or their concatenation), reducing the input length by a factor of 2. Subsampling is often performed between two or three layers, which leads to a reduction rate of 4 or 8. High reduction rates (e.g. 32) are possible, but it requires careful pre-training [3].

SZ was supported by a PhD Studenship funded by Toshiba Research Europe Limited. EL, PB, and SR were supported by EPSRC Project EP/R012180/1 (SpeechWave).

Although these length reduction methods yield excellent results, the subsampling is fixed. It does not consider different acoustic events: for example, encoder-decoder models have notable performance gains if the beginning silence part is removed entirely [14]. Furthermore, when the input acoustic features change rapidly, the skip rate (amount of subsampling) should be low while when the input acoustic features are similar, the skip rate should be high. However, static subsampling fails to recognise these situations. This strategy may retain redundant information, while dropping essential frames.

Recently, Skip RNN was proposed to allow an RNN to skip inputs dynamically [13]. The Skip RNN demonstrated promising results in several sequential modelling tasks. However, the Skip RNN uses only the previous RNN hidden state in order to predict if the current input should be skipped or not. This approach is risky, since relevant inputs may be overlooked while unnecessary inputs may be kept. For speech recognition, a similar Skip RNN model has been proposed, which also skips without considering the current input [15]. Compared with regular RNNs it has no gain in accuracy; neither has it been applied to sequence-to-sequence models.

The Skip RNN and our proposed dsRNN are related to Highway Networks [16]. Highway Networks are feedforward networks that use a transform gate and a carry gate to decide how much information from the current layer should be copied to the next layer. Similarly, Skip RNN and dsRNN determine if the hidden state of the current time step should be copied to the next time step. However, there are two main differences between Highway Networks and the dsRNN. First, Highway Networks are feedforward networks which may skip some intermediate layers. The proposed dsRNN is a recurrent network. Although from the viewpoint of unfolding computational graphs, it also omits some intermediate layers, the main characteristic of the dsRNN is the dropping of inputs. Second, Highway Networks use gates to drop intermediate layers in a soft way, while the dsRNN skips inputs in a hard fashion – the inputs are either retained or discarded.

In the perspective of a single time step, our method has some similarity to dropout [17]. However, dropout randomly drops some weights of the network while our model learns to skip the input. Thus, dropout puts a random mask on the network weights while our model puts a learned mask on the input.

3. Dynamic Subsampling RNN

An RNN maps a sequence of input frames $(\boldsymbol{x}_1, \cdots, \boldsymbol{x}_t)$ to a sequence of hidden representations $(\boldsymbol{h}_1, \cdots, \boldsymbol{h}_t)$. The hidden state $\boldsymbol{h}_i$ is computed recurrently by the RNN:

$$\boldsymbol{h}_i = \mathrm{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}) \tag{1}$$

We add an update gate to the RNN, which takes the previous hidden state $\boldsymbol{h}_{i-1}$ and the current input $\boldsymbol{x}_i$ to output an indicator $\boldsymbol{u}_i \in \{0, 1\}$, where 0 stands for skip $\boldsymbol{x}_i$ and 1 indicates using $\boldsymbol{x}_i$ to update the hidden state. The process is described below. First, the RNN computes a temporary current hidden state $\tilde{\boldsymbol{h}}_i$:

$$\tilde{\boldsymbol{h}}_i = \mathrm{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}) \,. \tag{2}$$

The update gate takes the concatenation of the temporary current RNN hidden state and the previous RNN hidden state as its input and uses a multilayer perceptron (MLP) to compute the increment of the skip probability $\Delta\boldsymbol{p}_i$:

$$\Delta\boldsymbol{p}_i = \mathrm{sigmoid}(\mathrm{MLP}([\boldsymbol{h}_{i-1}, \tilde{\boldsymbol{h}}_i])) \tag{3}$$

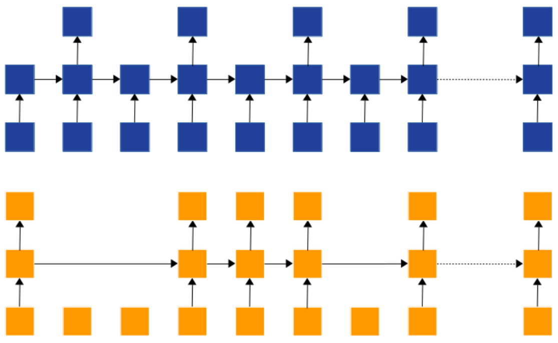

Figure 1: *Subsampling: static (above) vs. dynamic (below).*

When the RNN is composed of multiple layers, $\boldsymbol{h}_{i-1}$ and $\tilde{\boldsymbol{h}}_i$ can be the hidden states of one specific layer or the concatenation of the hidden states of all the layers. The advantage of computing an increment skip probability over estimating a skip probability directly is it distributes the probability over a time span, thus the network depends more on the context to make skips. It also prevents the network from making too few skips.

The current skip probability $\boldsymbol{p}$ is the sum of the previous accumulation skip probability $\boldsymbol{c}_{i-1}$ and the increment of the skip probability $\Delta\boldsymbol{p}_i$:

$$\boldsymbol{p}_i = \boldsymbol{c}_{i-1} + \min(\Delta\boldsymbol{p}_i, 1 - \boldsymbol{c}_{i-1}) \,. \tag{4}$$

The $\min$ operator is to ensure a valid probability.

The network can choose to skip the current input according to the skip probability $\boldsymbol{p}_i$. However, it may lead to unstable behaviors of the network. Thus, the probability is fed to a binarizer to produce a binary value $\boldsymbol{u}_i \in \{0, 1\}$:

$$\boldsymbol{u}_i = \mathrm{Binarizer}(\boldsymbol{p}_i) \,. \tag{5}$$

If $\boldsymbol{p}_i$ is above a threshold, then $\boldsymbol{u}_i$ is set to 1. Otherwise, $\boldsymbol{u}_i$ is set to 0. The threshold for the binarization function is set to 0.5 for the original Skip RNN [13]. In this work, considering the duration of different acoustic events varies, we use an MLP which takes the previous RNN hidden state $\boldsymbol{h}_{i-1}$ as the input to adjust the threshold. The binarization function is not differentiable. We set the binarization function to behave as a linear function with slope 1 during backpropagation [18].

If $\boldsymbol{u}_i$ is 1, the current input is used to update the RNN hidden state and the accumulation skip probability is reset to 0. Otherwise, the current input is skipped. The current RNN hidden state $\boldsymbol{h}_i$ and the accumulation skip probability $\boldsymbol{c}_i$ are:

$$\boldsymbol{h}_i = \boldsymbol{u}_i \cdot \tilde{\boldsymbol{h}}_i + (1 - \boldsymbol{u}_i) \cdot \boldsymbol{h}_{i-1} \tag{6}$$

$$\boldsymbol{c}_i = 0 + (1 - \boldsymbol{u}_i) \cdot \boldsymbol{p}_i \,. \tag{7}$$

Therefore, the proposed network learns to skip input frames dynamically by examining both history and current information.

We use our dsRNN as the encoder in the encoder-decoder model. The dsRNN is unidirectional, but will be compared with unidirectional and bidirectional RNNs in our experiments.

4. Experimental Setup

We test our model on the TIMIT dataset [19]. The dataset is divided into training, validation and test sets following the Kaldi s5 recipe [20]. The input features are 80-dimensional filter-bank features with energy extracted by the Kaldi toolkit [20]. The input features were rescaled by subtracting the mean and divided by the standard deviation from the training set. The outputs are 39 phonemes with start/end sentence (*sos*/*eos*) and the space token, which make 42 labels in total.

For the encoder-decoder architecture, we use three layers of dynamic subsampling unidirectional LSTM (dsULSTM) as the encoder. The baseline models use three layers of unidirectional LSTM (ULSTM) or bidirectional LSTM (BLSTM) as the encoder. For the baseline encoders, the top two layers read every second hidden state from their previous layers, resulting in the input sequence length being reduced by a factor 4. For all models the decoder is a one layer ULSTM and content-based attention is used [21, 22]. We test two settings of the number of hidden units. The first setting is to optimise the dsULSTM encoder. The number of hidden units is 300 for ULSTMs and 150 in each direction for BLSTM. The second setting is to optimise the baseline. The number of hidden units is 512 for ULSTMs and 256 in each direction for BLSTM. The MLP for estimating the increment of the skip probability uses Leaky ReLU [23] as its activation function. The number of hidden units of the MLP is 150 for the dsULSTM encoder with 300 hidden units and 100 for the dsULSTM encoder with 512 hidden units respectively.

The Adam algorithm [24] is used as the optimisation method. All the models are trained for 25 epochs. Beam Search with a beam size 20 is used for decoding. All the models are implemented through the ESPnet toolkit [25].

5. Results and Discussion

5.1. dsULSTM vs. baseline models

First, we compare the dsULSTM with the original Skip LSTM as well as the baseline ULSTM models. Since the LSTM encoders have three layers, we test using the hidden states in the bottom layer, in the middle layer, in the top layer, and the concatenation of the hidden states of all the three layers to make the skip decisions respectively.

Table 1: *The phone error rates (PER) on TIMIT test set of using hidden states in different layers to make the skip decisions.*

Layer used for skip decision	PER
Bottom	23.8%
Middle	20.5%
Top	**19.4%**
Concatenation of all layers	22.5%
Skip LSTM (use the top layer)	26.4%
ULSTM	29.4%
ULSTM + static subsampling	24.7%

The results in Table 1 show subsampling is essential for good accuracy. Our dsULSTM encoder is notably better than LSTM encoder with static subsampling and it indicates that the static subsampling method omits some relevant information. The Skip LSTM does not yield a competitive PER, which implies for speech recognition, skipping input frames blindly is harmful. To better understand the dsULSTM encoder, we make a statistic of the average ratio between the number of skipped frames and the number of all frames for each sequence on the TIMIT development set for each training epoch.

The average skip ratios are shown in Figure 2. The skip ratios go to zero after the first few epochs, except when using the bottom layer to decide the skips. This is unsurprising. We argue that when using the bottom layer, the hidden states are similar to each other when the input frames are similar, and leads to high

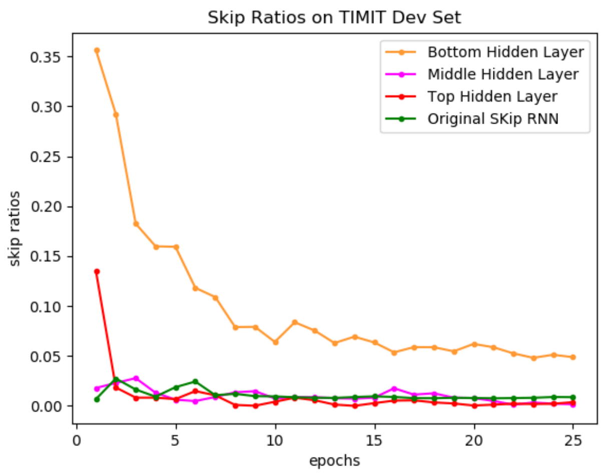

Figure 2: *The average skip ratios on TIMIT development set for each epoch.*

skip rates. When using the top layer, the hidden states are more abstract and well separated and resulting in low skip rates. We find that high skip rates in the first epochs is important for low PER – it forces the encoder to learn the abstract information of the most relevant frames at the beginning of the training. Then, the low skip rates in the remaining epochs retain more input information. Both Skip LSTM and dsULSTM using middle layer to decide skips have overall low skip ratios but Skip LSTM has a much higher PER. Furthermore, Skip LSTM also has a higher PER than the static subsampling method. Thus, we conclude Skip LSTM often makes unsuitable skips.

5.2. dsULSTM vs. random subsampling

We observe that when using the top layer or the bottom layer to decide the skips, even in the first epoch, the network already learns to make meaningful skips (e.g. it always skips the silence part). To further show our model makes meaningful skips rather than skipping randomly, we compare dsULSTM with random skips. In our best model (using the top layer), the skip rates after the first epoch and the second epoch are 0.14 and 0.02. Thus, we test randomly skipping inputs during training with a fixed probability 0.14 and with a fixed probability 0.02, respectively.

To better mimic the behaviour of dsULSTM, we test randomly skipping input with skip probability decay. Since the weights of dsULSTM are initialized around 0, the initial skip rate is close to 0.5 because of the sigmoid function. Thus, the initial skip probability is set to 0.5 for skip probability decay with random skips. The skip probability is 0.14 for epoch 2 and 0.02 for epoch 3. After the third epoch, the skip probability is 0.004, which is the average skip rates of dsULSTM for the remaining epochs. We also compare the learned skip probability decay policy with two other skip probability decay strategies. The first method is decaying exponentially: the ratio between the current skip probability and the previous skip probability is $(\frac{1}{2})^n$, where n is the number of epochs. The skip probability stops decaying after the third epoch, since the probability is already close to 0. The second approach is to lower the initial skip probabilities.

Surprisingly, Table 2 indicates that a random skip probability 0.14 has a better PER compared with ULSTM with static

Table 2: *PER on TIMIT test set of randomly skipping inputs during training.*

Fixed random skip probability	PER
0.2	26.7%
0.14	21.6%
0.1	22.7%
0.02	29.0%
Skip probability decay	
$0.5 \rightarrow 0.14 \rightarrow 0.02 \rightarrow 0.004$	21.1%
$0.5 \rightarrow 0.25 \rightarrow 0.0625 \rightarrow 0.0078$	23.1%
$0.35 \rightarrow 0.1 \rightarrow 0.02 \rightarrow 0.004$	23.7%
dsULSTM	**19.4%**
ULSTM + static subsampling	24.7%

subsampling. We also notice that the PER increases when the random skip probability diverges from 0.14. Thus, we conclude that for this architecture, the average redundant frame rate is around 0.14. A random skip probability of 0.02 does not filter enough irrelevant frames nor sufficiently reduce the length of the input sequence and leads to a poor result.

Although a fixed skip probability of 0.14 is beneficial and a fixed skip probability of 0.02 is harmful, when combined the PER is lower than using the skip probability 0.14 alone. Moreover, all the tested skip probability decay policies yield better results than ULSTM with static subsampling. This further supports the hypothesis that high skip rates at the beginning of training are helpful, even when the skips are randomly made. In the early epochs, due to random skips, the inputs are shortened – it is easier for the encoder to learn the hidden representation of shortened sequences. Then, in the remaining epochs, low skip probabilities ensure the encoder taking all inputs. In contrast, static subsampling misses some input information.

Finally, we argue that our model learns the best dynamic skip strategy and makes meaningful skips. Firstly, both the best random skip rate (0.14) and the best skip rate decay method are learned by our model. Second, with a similar skip rate for each epoch during training, our dsULSTM has a lower PER than the best skip rate decay policy with random skips, which indicates that the dsLSTM makes meaningful skips rather than skipping randomly.

Table 3: *PERs of different encoder models on TIMIT test set.*

Encoder: 3 layers with 300 hidden units	PER
ULSTM	29.4%
ULSTM + static subsampling	24.7%
BLSTM + static subsampling	21.1%
dsULSTM	**19.4%**
Encoder: 3 layers with 512 hidden units	
ULSTM	28.3%
ULSTM + static subsampling	23.5%
BLSTM + static subsampling	20.4%
dsULSTM	**19.9%**

All the experiments above for random skipping and dsULSTM use architectures which are optimised for the dsULSTM (3 hidden layers with 300 hidden states for ULSTM encoders and 3 hidden layers with 150 hidden state in each direction for the BLSTM encoder). We also test our model on an architecture optimised for the baseline (3 hidden layers with 512 hidden states for ULSTM encoders and 3 hidden layers with 256 hidden state in each direction for the BLSTM encoder). Results are in Table 3. The dsULSTM encoders result in lower PER than the BLSTM encoders, indicating that the context information gained from BLSTM is inferior to the information loss from the static subsampling.

5.3. dsULSTM on ULSTM

Static subsampling does not allow LSTM encoders to drop inputs directly – the bottom layer always reads the entire input sequence and subsampling operations are between stacked LSTM layers. Thus, we test to stack two dsULSTM layers on one ULSTM layer. Table 4 shows for the second architecture of the encoders, two dsULSTM layers stacked on one ULSTM layer yields significant better results than all other models. This may be due to the fact that the bottom ULSTM layer prevents the model from losing any input information, since the bottom ULSTM does not drop any input frame. However, further investigation is needed since in the first architecture, though better than static subsampling, stacked dsULSTM on ULSTM is no better than the pure dsULSTM encoder.

Table 4: *PERs of different models on TIMIT test set. The bidirectional GRU encoder-decoder is better than our BLSTM models. However, it requires complicated training and decoding procedures [26].*

Encoder: 3 layers with 300 hidden units	PER
ULSTM + static subsampling	24.7%
BLSTM + static subsampling	21.1%
dsULSTM	**19.4%**
dsULSTM on ULSTM	23.3%
Encoder: 3 layers with 512 hidden units	
ULSTM + static subsampling	23.5%
BLSTM + static subsampling	20.4%
dsULSTM	19.9%
dsULSTM on ULSTM	**16.8%**
Results of previous end-to-end models	
Bidirectional GRU encoder-decoder [26]	18.7%
Segmental RNNs [27]	20.5%
BLSTM CTC [28]	24.6%

6. Conclusion

In this paper, we propose a novel dynamic subsampling RNN and use it as the encoder in encoder-decoder models. Compared with statically subsampled RNN encoders, our dsRNN encoder has significant performance gains. The dsRNN encoder learns a good dynamic subsampling strategy – the skip rate decay. Also, it learns to make meaningful skips rather than skipping randomly. Thereby, at the beginning of training, the skip rates are high and the skips are valid, which makes the encoder learn the hidden representations of the most relevant input frames. As training progresses, the low skip rates retain input information.

Although arguably, the input gate and the forget gate in LSTM should have the ability to drop inputs and copy hidden states dynamically, the dsLSTM results in lower PERs than LSTM for speech recognition experiments on TIMIT. Thus, encouraging RNNs to drop inputs and copy hidden states in a hard fashion may be beneficial for speech recognition tasks.

Our future work will include extending the model to bidirectional and investigating if the model can skip noise frames.

Source code: `https://github.com/qishuxiyou/dsRNN`.

7. References

[1] T. Hori, S. Watanabe, Y. Zhang, and W. Chan, “Advances in joint ctc-attention based end-to-end speech recognition with a deep cnn encoder and rnn-lm,” *arXiv preprint arXiv:1706.02737*, 2017.

[2] C.-C. Chiu, T. N. Sainath, Y. Wu, R. Prabhavalkar, P. Nguyen, Z. Chen, A. Kannan, R. J. Weiss, K. Rao, E. Gonina *et al.*, “State-of-the-art speech recognition with sequence-to-sequence models,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4774–4778.

[3] A. Zeyer, K. Irie, R. Schlüter, and H. Ney, “Improved training of end-to-end attention models for speech recognition,” *arXiv preprint arXiv:1805.03294*, 2018.

[4] Y. Bengio, P. Simard, P. Frasconi *et al.*, “Learning long-term dependencies with gradient descent is difficult,” *IEEE transactions on neural networks*, vol. 5, no. 2, pp. 157–166, 1994.

[5] S. Hochreiter and J. Schmidhuber, “Long short-term memory,” *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[6] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, “Empirical evaluation of gated recurrent neural networks on sequence modeling,” *arXiv preprint arXiv:1412.3555*, 2014.

[7] L. Lu, X. Zhang, K. Cho, and S. Renals, “A study of the recurrent neural network encoder-decoder for large vocabulary speech recognition,” in *Sixteenth Annual Conference of the International Speech Communication Association*, 2015.

[8] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4960–4964.

[9] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 4945–4949.

[10] T. Ochiai, S. Watanabe, T. Hori, and J. R. Hershey, “Multichannel end-to-end speech recognition,” in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2632–2641.

[11] S. Kim, T. Hori, and S. Watanabe, “Joint ctc-attention based end-to-end speech recognition using multi-task learning,” in *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2017, pp. 4835–4839.

[12] C. Weng, J. Cui, G. Wang, J. Wang, C. Yu, D. Su, and D. Yu, “Improving attention based sequence-tosequence models for end-to-end english conversational speech recognition,” *Proc. Interspeech, Hyderabad, India*, pp. 761–765, 2018.

[13] V. Campos, B. Jou, X. Giró-i Nieto, J. Torres, and S.-F. Chang, “Skip rnn: Learning to skip state updates in recurrent neural networks,” *arXiv preprint arXiv:1708.06834*, 2017.

[14] S. Zhang, E. Loweimi, P. Bell, and S. Renals, “Windowed attention mechanisms for speech recognition,” in *2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019.

[15] I. Song, J. Chung, T. Kim, and Y. Bengio, “Dynamic frame skipping for fast speech recognition in recurrent neural network based acoustic models,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4984–4988.

[16] R. K. Srivastava, K. Greff, and J. Schmidhuber, “Highway networks,” *arXiv preprint arXiv:1505.00387*, 2015.

[17] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *The Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[18] Y. Bengio, N. Léonard, and A. Courville, “Estimating or propagating gradients through stochastic neurons for conditional computation,” *arXiv preprint arXiv:1308.3432*, 2013.

[19] J. S. Garofolo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, and D. S. Pallett, “DARPA TIMIT acoustic-phonetic continuous speech corpus CD-ROM. NIST speech disc 1-1.1,” *NASA STI/Recon technical report n*, vol. 93, 1993.

[20] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel, M. Hannemann, P. Motlicek, Y. Qian, P. Schwarz *et al.*, “The Kaldi speech recognition toolkit,” in *IEEE ASRU*, 2011.

[21] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” *arXiv preprint arXiv:1409.0473*, 2014.

[22] M.-T. Luong, H. Pham, and C. D. Manning, “Effective approaches to attention-based neural machine translation,” *arXiv preprint arXiv:1508.04025*, 2015.

[23] B. Xu, N. Wang, T. Chen, and M. Li, “Empirical evaluation of rectified activations in convolutional network,” *arXiv preprint arXiv:1505.00853*, 2015.

[24] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization,” *arXiv preprint arXiv:1412.6980*, 2014.

[25] S. Watanabe, T. Hori, S. Karita, T. Hayashi, J. Nishitoba, Y. Unno, N. E. Y. Soplin, J. Heymann, M. Wiesner, N. Chen *et al.*, “ESPnet: End-to-end speech processing toolkit,” *arXiv preprint arXiv:1804.00015*, 2018.

[26] J. K. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-based models for speech recognition,” in *Advances in neural information processing systems*, 2015, pp. 577–585.

[27] L. Lu, L. Kong, C. Dyer, N. A. Smith, and S. Renals, “Segmental recurrent neural networks for end-to-end speech recognition,” *arXiv preprint arXiv:1603.00223*, 2016.

[28] S. Fernández, A. Graves, and J. Schmidhuber, “Phoneme recognition in timit with blstm-ctc,” *arXiv preprint arXiv:0804.3269*, 2008.