INTERSPEECH 2019
September 15–19, 2019, Graz, Austria

Framewise Supervised Training towards End-to-End Speech Recognition Models: First Results

Mohan Li, Yuanjiang Cao, Weicong Zhou, Min Liu

Toshiba (China) R&D Center, China

limohan@toshiba.com.cn, caoyuanjiang.learner@gmail.com, weicongzhou@outlook.com, liumin@toshiba.com.cn

Abstract

Recurrent neural networks (RNNs) trained with connectionist temporal classification (CTC) technique have delivered promising results in many speech recognition tasks. However, the forward-backward algorithm that CTC takes for model optimization requires a huge amount of computation. This paper introduces a new training method towards RNN-based end-to-end models, which significantly saves computing power without losing accuracy. Unlike CTC, the label sequence is aligned to the labelling hypothesis and then to the input sequence by the Weighted Minimum Edit-Distance Aligning (WMEDA) algorithm. Based on the alignment, the framewise supervised training is conducted. Moreover, Pronunciation Embedding (PE), the acoustic representation towards a linguistic target, is proposed in order to calculate the weights in WMEDA algorithm. The model is evaluated on TIMIT and AIShell-1 datasets for English phoneme and Chinese character recognitions. For TIMIT, the model achieves a comparable 18.57% PER to the 18.4% PER of the CTC baseline. As for AIShell-1, a joint Pinyin-character model is trained, giving a 19.38% CER, which is slightly better than the 19.43% CER obtained by the CTC character model, and the training time of this model is only 54.3% of the CTC model's.

Index Terms: connectionist temporal classification, weighted edit-distance aligning, pronunciation embedding

1. Introduction

The advantages in dealing with sequential data make recurrent neural networks (RNNs) the ideal base models in a variety of deep learning tasks, such as speech recognition, machine translation and image caption generation. RNNs are firstly introduced to automatic speech recognition (ASR) systems as the replacement to the Feedforward Neural Networks (FNN) in the Hidden Markov Models (HMM)-hybrid acoustic models [1]. Recently, an increasing amount of works focus on building an independent RNN system that can be trained end-to-end, which is epitomized by connectionist temporal classification (CTC) technique [2] and attention-based encoder-decoder framework [3]. Compared to traditional ASR systems, minimal information is required by the end-to-end training methods. For example, the pronunciation lexicon and external language model which used to be indispensable components of the traditional system are no longer needed, as the end-to-end models could integrate their functions into a single neural structure.

CTC technique has been regarded as one of the most effective solutions to problems that the mapping between input and target sequences is unknown [2]. Instead of training the models to do framewise classification as the HMM-RNN does, CTC maximizes the overall likelihood that the model predicts the correct label sequence, without concerning about at which time steps the sequence is generated. The forward-backward algorithm is employed to calculate the probabilities of all possible alignments as well as the loss gradients. Although this algorithm is obviously more efficient than the labelling path enumeration, the cost of computation turns out to be considerable when the input length increases, which would cause huge burdens to the model optimization process.

This paper will present a new end-to-end training method towards RNN-based speech recognition models. From CTC, we borrow the concepts of the "blank" label and the many-to-one function $\mathcal{F}$ [2] which maps the raw output sequence into the labelling hypothesis. Unlike CTC, which aligns the label sequence to the whole input time steps, our method aligns the label sequence to the labelling hypothesis using the weighted minimum edit-distance aligning (WMEDA) algorithm. By recording the frame location of each hypothesis output, we could further align the label sequence to the input sequence, so as to conduct the framewise supervised training. To calculate the weights used in WMEDA algorithm, we propose the pronunciation embedding (PE), a vector that represents the acoustic features of a linguistic output (e.g. phoneme and grapheme), which could be easily obtained from any other well-trained end-to-end ASR models.

2. Related Work

Since the emergence of CTC, there have been intensive researches to apply this technique to English phoneme recognition tasks. Compared to HMM-hybrid models, better performances on TIMIT dataset were reported in [2, 4, 5]. With the rising of end-to-end training criteria, CTC was proved to be capable of directly transcribing acoustics to readable linguistic outputs, such as English characters [6, 7, 8]. Extensions of CTC-based models are represented by RNN Transducer [9] and Recurrent Neural Aligner [10], which augment the acoustic-only model with another RNN decoder to capture the interdependencies between outputs. Similar to pure CTC models, these variants also use the forward-backward algorithm to compute the labelling probability given by the RNN encoder.

The attention-based encoder-decoder framework was recently introduced to speech recognition, which could produce soft alignments between targets and the corresponding speech segments. Such sequence-to-sequence architecture is by design free of the conditional independent assumption to which CTC is restricted, making the model more expressive to generate coherent outputs. Comparable results were also obtained by attention-based models with various output levels, such as phoneme [11, 12] and grapheme [13, 14]. Similar techniques include monotonic attention [15] and ACS [16] algorithms. Some other works applied both attention mechanism and CTC to conduct multi-task training [17], and it is claimed that the model performance was benefitted from the joint decoding.

3. Framewise Supervised Training

The model architecture adopted in this paper consists of two bidirectional RNNs that can be trained end-to-end to predict linguistic outputs given the input utterance. Like CTC-based models, we extend the vocabulary with a "blank" symbol $\emptyset$ to label the frames where no output should be emitted. Let $\boldsymbol{x} = (\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T)$ be a sequence of feature vectors (e.g. log-mel filterbank spectra) with the length T, and $\boldsymbol{l} = (l_1, l_2, \dots, l_N)$ denote the label sequence with length N. The first RNN, followed by a single softmax-output layer, processes $\boldsymbol{x}$ into the target distributions $\boldsymbol{p} = (\boldsymbol{p}_1, \boldsymbol{p}_2, \dots, \boldsymbol{p}_T)$. Then we take the element with the largest probability at each time step to produce the raw hypothesis sequence $\widetilde{\boldsymbol{h}} = (h_1, h_2, \dots, h_T)$. With the help of many-to-one function $\mathcal{F}$, all $\emptyset$ labels and consecutively repeated outputs are removed (here we keep the last output in the repetition, for example, if $h_{t-3}, h_{t-2}, h_{t-1}, h_t$ were identical, only $\widetilde{h}_t$ together with its index *t* would remain), thereby we obtain the final hypothesis sequence $\boldsymbol{h} = (h_{t_1}, h_{t_2}, \dots, h_{t_H})$, where *t* is the time step at which the hypothesis *h* is emitted, and H is the length of $\boldsymbol{h}$ (H ≤ T).

We apply WMEDA algorithm to the hypothesis sequence $\boldsymbol{h}$ and the label sequence $\boldsymbol{l}$ to precisely map the hypothesis elements to the ground-truth labels. Fortunately, as the WMEDA algorithm can also provide the error types (correct, substitution, insertion and deletion) of the aligned hypothesis, we could easily acquire all the information needed to conduct supervised training for each frame. The information is given in a table like:

$$\mathbf{I} = \left((l_1, h_{t_1}, e_1), (l_2, h_{t_2}, e_2), \dots, (l_N, h_{t_H}, e_H)\right)$$

where *e* serves as the error type of *h*. Note that N may not equal to H, as the result of insertion and deletion errors, so items like $('-', h_{t_i}, \text{"}insertion\text{"})$ or $(l_i, '-', \text{"}deletion\text{"})$ are very likely to appear in the table, where $'-'$ means no objects are aligned to. The items with insertion errors are simply erased from the table. As for the deletion error cases, within the interval where the deletion occurs, we select the time step with the largest label probability as the time index for l_i, which is calculated as:

$$\text{t} = \overset{argmax}{t} \boldsymbol{p}_t(l_i), \ \text{t} \in (t_{i-1}, t_{i+1}), \tag{1}$$

where t_{i-1} and t_{i+1} are the time indices of the last and next non-deletion labels to the current deleted label l_i. Now, since each label element has been aligned to a particular time index, we have the revised information table $\boldsymbol{I}'$:

$$\boldsymbol{I}' = \left((l_1, t_1), (l_2, t_2), \dots, (l_N, t_N)\right).$$

Finally, by labelling all the non-emitting time steps with $\emptyset$ labels (including the time steps of the erased repeated outputs and insertion errors), a complete label sequence $\boldsymbol{l}' = (l'_1, l'_2, \dots, l'_{T},)$ covering all frames in the utterance is generated, based on which the cross-entropy between $\boldsymbol{p}$ and $\boldsymbol{l}'$ could be used as the loss function to train the first RNN.

As the above aligning algorithm implies, each element in the label sequence is only encouraged to emit once throughout the whole utterance, which makes it even harder for this model

Algorithm 1. Framewise Supervised Training

```
Initialize: 1st-RNN, 2nd-RNN
while stopping criterion not satisfied do
    Extract a training batch $(\boldsymbol{x}, \boldsymbol{l})$
    $\boldsymbol{s}, \boldsymbol{p}$ = 1st-RNN($x$)
    $\boldsymbol{l}' = [], \boldsymbol{s}' = []$
    for $i$ in range(batch_size) do
        $\widetilde{\boldsymbol{h}}^i = \mathbf{argmax}(\boldsymbol{p}^i)$
        $\boldsymbol{h}^i = \mathcal{F}(\widetilde{\boldsymbol{h}}^i)$
        $\boldsymbol{I}^i = \mathbf{WMEDA}(\boldsymbol{h}^i, \boldsymbol{l}^i)$
        $\boldsymbol{I}'^i = \boldsymbol{Rewise}_{insertion\&deletion}(\boldsymbol{I}^i)$
        $\boldsymbol{l}'^i = \mathbf{Supplement}_{\emptyset}(\boldsymbol{I}'^i)$
        $\boldsymbol{s}'^i = \mathbf{SelectTime}(\boldsymbol{s}^i, \boldsymbol{I}'^i)$
        $\boldsymbol{l}' = \mathbf{Stack}(\boldsymbol{l}', \boldsymbol{l}'^i), \boldsymbol{s}' = \mathbf{Stack}(\boldsymbol{s}', \boldsymbol{s}'^i)$
    end for
    $\boldsymbol{p}'$ = 2nd-RNN($\boldsymbol{s}'$)
    $\mathbf{loss}_{\mathbf{1}^{st}-\mathbf{RNN}} = \boldsymbol{CrossEntropy}(\boldsymbol{p}, \boldsymbol{l}')$
    $\mathbf{loss}_{\mathbf{2}^{nd}-\mathbf{RNN}} = \boldsymbol{CrossEntropy}(\boldsymbol{p}', \boldsymbol{l})$
    $\mathbf{loss} = \mathbf{loss}_{\mathbf{1}^{st}-\mathbf{RNN}} + \mathbf{loss}_{\mathbf{2}^{nd}-\mathbf{RNN}}$
    Jointly update 1st-RNN, 2nd-RNN by BP algorithm
end while
```

to capture the inter-label dependencies than CTC-based models. Thus, we incorporate another single layer bidirectional RNN for a second-pass correction. Given the first RNN's hidden states $\boldsymbol{s} = (\boldsymbol{s}_1, \boldsymbol{s}_2, \dots, \boldsymbol{s}_T)$, we select entries of $\boldsymbol{s}$ at the time steps that are aligned to the label sequence $\boldsymbol{l}$, which is $\boldsymbol{s}' = (\boldsymbol{s}_{t_1}, \boldsymbol{s}_{t_2}, \dots, \boldsymbol{s}_{t_N})$, as the inputs to the second RNN. Then the outputs of it will also be projected into the distribution $\boldsymbol{p}'$ of the vocabulary, and again the cross-entropy between $\boldsymbol{p}'$ and $\boldsymbol{l}$ is used as the loss for model optimization. The two RNNs are jointly trained as we hope the loss gradients of the second RNN could be back-propagated into the first one to benefit the accuracy of the hypothesis sequence $\boldsymbol{h}$. See Algorithm 1 for the pseudo codes of the proposed training method.

3.1. Weighted minimum edit-distance aligning algorithm

Weighted minimum edit-distance (WMED) algorithm is a typical implementation of dynamic-programming, which works out the minimum times of edit operations needed to transform one string to the other, given the cost of transformation. WMED Aligning algorithm, namely WMEDA, is to backtrack the path along which the minimum edit-distance happens, provided the distance table generated at the WMED step.

According to the core idea of dynamic-programming, the distance at each aligning point could be iteratively computed based on the previous values from three directions:

$$d(i,j) = \text{cost}(i,j) + \min \begin{cases} d(i-1, j-1) \\ d(i-1, j) \\ d(i, j-1) \end{cases}, \tag{2}$$

where *i* and *j* is respectively the index of label and hypothesis sequences prepended with a ε symbol (to allow the first hypothesis element to be an insertion error). The weight (cost) matrix plays a vital role in our training method, as they help to determine the most reliable alignment path when some elements could be aligned to multiple targets. This situation frequently happens at the early stage of training, when a large amount of substitution and insertion errors occur. For example, given the label sequence "abc" and a hypothesis sequence "$adefc$", in the backtracking step, MEDA will give a random alignment result out of three options:

a-a (cor), b-d (sub), ∅-e (ins), ∅-f (ins), c-c (cor),
a-a (cor), ∅-d (ins), b-e (sub), ∅-f (ins), c-c (cor),
a-a (cor), ∅-d (ins), ∅-e (ins), b-f (sub), c-c (cor).

We could observe that element "b" could be aligned to anyone among elements "d", "e" and "f", resulting in one substitution error and two insertion errors. This could cause serious confusions to the supervised learning process, as unreliable labels would be given to a lot of frames, making the model hard to converge. Thus, we must find a way to produce the weights used in WMEDA algorithm.

3.2. Pronunciation embedding

Similar to the word embedding in the field of natural language processing (NLP), we expect linguistic targets that are acoustically alike could also be close to each other in some high-dimensional space. Given this property, the acoustic similarity between the targets can simply be calculated as the cosine distance of the corresponding vectors, and we name such vectors the pronunciation embedding (PE). For end-to-end ASR models, as the output layer directly projects the hidden states to the target distribution, we could take straight the rows of its weights $W_{ho} \in \Re^{|V|\times H}$ as the pronunciation embedding of the targets, where $|V|$ and H are the size of the target vocabulary and the number of hidden units, respectively.

The PE matrix may be easily obtained from any well-trained CTC or sequence-to-sequence models. Even if there isn't any of them on hand, we can still train an under-converged one using a small amount of data, as the loss gradients are immediately propagated to the weights of the output layer during optimization, making them quickly enter a stable state.

Therefore, given linguistic targets A and B, along with their corresponding PE, we define the cost of aligning in the WMEDA algorithm as:

$$\begin{aligned}\mathrm{cost}(\mathrm{A},\mathrm{B}) &= 1-\mathrm{Similarity}(\boldsymbol{PE}_A,\boldsymbol{PE}_B)\\ &= 1-\frac{1+\dfrac{\boldsymbol{PE}_A^T\cdot\boldsymbol{PE}_B}{\|\boldsymbol{PE}_A\|\cdot\|\boldsymbol{PE}_B\|}}{2}\\ &= \frac{1}{2}-\frac{\boldsymbol{PE}_A^T\cdot\boldsymbol{PE}_B}{2\cdot\|\boldsymbol{PE}_A\|\cdot\|\boldsymbol{PE}_B\|}\end{aligned} \qquad (3)$$

Note before training our model, a cost table should be generated to cover every two targets in the vocabulary, so that the acquisition of the WMEDA weights would simply require a lookup operation, which largely saves the training time.

3.3. Comparison of computational complexity

The differences between the computational complexity of CTC and our proposed training method occur in two stages, which are the loss calculation and the back-propagation of the loss gradients. First, CTC takes the forward-backward algorithm to compute loss, where at each time step t, the forward and backward variables are computed for ∅ labels and all other elements in the ∅-augmented label sequence z. Taking the forward variables for example:

$$\alpha(t,u)=\sum_{k=f}^{u}\alpha(t-1,k)\cdot y_k^t,$$

$$f=\begin{cases}u-1, & if\ l'_k=\emptyset\\ u-2, & otherwise\end{cases}, \qquad (4)$$

where u is the index of the label element being focused on, and y is the corresponding probability given by the RNN model. From equation (4), we can figure out that it requires N additions and (N+1) multiplications to compute the forward variables for ∅ labels, as well as (2N-1) additions and N multiplications for that of the emitting labels, resulting in an amount of computation as approximate 3N additions plus 2N multiplications. Together with the backward variables, the total amount of computation of CTC's loss calculation sums up to 6NT additions plus 4NT multiplications, given T as the length of the inputs. By contrast, as equation (2) suggests, the calculation towards the edit-distances in our WMEDA step only requires 3NH times of computation, including NH additions and 2NH comparison operations, where H is the length of the hypothesis sequence. At the early stage of training, when very few ∅ labels are emitted, H is numerically close to T. However, as the model converges, H will quickly be reduced to around the length of the label sequence, N, which further helps to speed up the training.

On the other hand, for CTC, the loss gradient that is propagated to the output a_j^t of the RNN model can be calculated as:

$$\frac{\partial\mathcal{L}(x,z)}{\partial a_j^t}=y_j^t-\frac{1}{p(l|z)}\sum_{u\in B(z,j)}\alpha(t,u)\beta(t,u), \qquad (5)$$

where y_j^t is the softmax of a_j^t, and $B(\boldsymbol{z},j)$ is the set of positions where j appears in z. This totally requires $|V|$T additions and (2N+2)T multiplications for all the RNN's output units and for all time steps. As for the loss gradient in our cross-entropy function,

$$\frac{\partial\mathcal{L}(x,l\prime)}{\partial a_j^t}=y_j^t-l'^t_j, \qquad (6)$$

only $|V|$T additions are needed during the back-propagation.

4. Experiments

We verified the proposed training method on two open-source datasets TIMIT [18] and AIShell-1 [19] for English phoneme and Chinese grapheme recognitions. In both experiments, we initialized the first RNN by letting it predict the HMM states obtained from the tradition ASR system, and the PE was collected from the attention models in our previous work [16]. Adam algorithm [20] was used in the first stage of training to achieve fast convergence. The second RNN was not involved until the first RNN reached a good recognition performance, and meanwhile, we switched to momentum stochastic gradient descent (SGD) to jointly train the two RNNs. Besides, we didn't rule out the insertion errors from the information table in the first several epochs, so as to encourage the first RNN to produce non-blank outputs. Otherwise, the model would quickly over-fit to the blank label, as it occupies most of the frames in the utterance.

Previous works [21] have proved that the acoustic part of end-to-end models are difficult to converge with Chinese data when the models are designed to directly predict the Chinese characters. This is because Chinese characters provide minimal information towards the pronunciations of the spoken language. Therefore, in the AIShell-1 task, we would rather let the first RNN output the phonetic representation of the Chinese character, Pinyin, as an intermediate product of the model, and then use the second RNN to transcribe the corresponding hidden states to the final character targets.

Table 1: Phoneme error rate (PER) evaluated on TIMIT

Network	**PER**
HMM-based Models	
Triphone GMM-HMM [22]	25.6%
HMM-DNN [22]	18.5%
End-to-end Models	
Attention Model [12]	18.57%
RNN Transducer [9]	17.7%
CTC [9]	18.4%
Our Model (first RNN)	18.62%
Our Model (second RNN)	18.57%

4.1. TIMIT

The architecture of the first RNN that we used for the TIMIT dataset follows exactly the CTC model presented in [5], which was a 5-layer 250-unit Bi-LSTM. And the second RNN was implemented as a 1-layer 250-unit Bi-LSTM. The set-up of the experiment complied with the definition in Kaldi TIMIT s5 recipe [22], where 3696 utterances from 462 speakers formed the training set, and another set of 50 speakers was used for early-stopping. Our model was evaluated on the core test set which contains 24 speakers' 192 sentences. The feature adopted in our experiment consisted of 40-dimensional mel-filterbank coefficients with the first and second derivatives, resulting in the total input size of 120. We used all 61 phoneme labels for training and decoding, and then mapped the outputs to 39 labels to calculate the final error rate. The evaluation results of our model are produced by the beam search decoding with a width of 16.

As shown in Table 1, our model has delivered promising performances on par with that of the HMM-hybrid model and all other end-to-end models. The final result, as the 18.57% PER given by the second RNN in our model, is slightly better than the first RNN's 18.62% PER without second-pass correction. The best result is achieved by the RNN Transducer, which incorporates a more complicated RNN decoder as in the attention-based model.

4.2. AIShell-1

For AIShell-1 dataset, we used a 5-layer 256-unit Bi-GRU for the first RNN, and a 1-layer 256-unit Bi-GRU for the second one. The same structure was adopted for the CTC baseline. We also followed the Kaldi AIShell-1 s5 recipe as the division scheme towards the dataset, where 119779 utterances presenting 150 hours were used as the training data, and 14236 utterances for about 10 hours were used for the development set. We evaluated the models on the standard test set which contains 7176 sentences. The components of the input feature include 24-dimensional energy augmented mel frequency cepstral coefficients (MFCC) plus 1-dimensional pitch information with their first and second derivatives. The Chinese character vocabulary covers 7065 frequently used characters plus a ∅ symbol, and the corresponding 1836 Pinyin tokens are included in the Pinyin vocabulary.

We trained our model for 30 epochs totally, with the first 10 epochs involving only the first RNN to predict Pinyin, and the rest 20 epochs jointly training the two RNNs for character-level recognition. Table 2 illustrates the results given by the greedy decoding.

Table 2: Character error rate (CER) evaluated on AIShell-1

Network	**CER**	**Training Time**
HMM-based Model		
HMM-DNN [22]	8.5%	--
End-to-end Models		
Attention Model [16]	23.2%	--
ACS Model [16]	21.6%	--
CTC (pre-trained)	19.43%	215 min/epoch
Our Model	19.38%	110 min/epoch (1-10)
		120 min/epoch (11-20)

From Table 2, we could observe that the model trained with our proposed method outperforms all other end-to-end models including the CTC baseline, which was also initialized by the HMM state. The disparity between end-to-end models and the HMM-hybrid model is still obvious, whereas we believe larger datasets would narrow the gap. Although our model has introduced little advantage over the CTC model in terms of CER, the time it saved for the training is considerable. For the total 30 epochs, our model got fully trained with only 54.3% of the time that was used on the CTC model.

Additional experiments were conducted to measure how much harm the second-pass correction does to the decoding speed. On the whole test set of AIShell-1, it took an average 146.32 ms to decode an utterance in our testing environment, and the second-pass correction lasted for 6.76 ms, accounting for only 4.62% of the total decoding time. The figure for the CTC model came to 147.59 ms, as the softmax dimension at the RNN's output layer is much larger than that of the first RNN in our model, which significantly slows down the decoding process.

5. Conclusions

In this paper, we present a new method of training end-to-end speech recognition models in a framewise supervised manner. Like CTC, no pre-segmentation towards the utterance is needed, and the model would automatically decide the time steps to emit the most probable outputs. The Weighted Minimum Edit-Distance Aligning algorithm is used in the method as a replacement to the forward-backward algorithm of CTC, and the Pronunciation Embedding is proposed to calculate the cost of aligning. The models trained with the method achieved similar performances to that of the CTC baselines, on both TIMIT and AIShell-1 tasks. More importantly, the training time can be cut down to a large extent by using the method, which is a 45.7% reduction on AIShell-1compared to CTC.

However, for all the experiments we have done, the proportions of the insertion and deletion errors delivered by our models were found larger than the counterparts of the CTC models. It is assumed that encouraging only one target to emit at each frame is hard to balance the chance to output ∅ and other labels. We plan to solve this problem by revising the cross-entropy function in future work.

6. Acknowledgements

All experiments were conducted using Pytorch libraries [23].

7. References

[1] A. J. Robinson, “An application of recurrent nets to phone probability estimation,” *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 298-305, 1994

[2] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning. ACM*, 2006, pp. 369–376.

[3] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate,” in *International Conference on Learning Representations*, 2015.

[4] S. Fernández, A. Graves, and J. Schmidhuber, “Phoneme recognition in TIMIT with BLSTM-CTC,” IDSIA, Tech. Rep., 2008.

[5] A. Graves, A-R. Mohamed, and G. Hinton. “Speech Recognition with Deep Recurrent Neural Networks,” in *Acoustics, Speech and Signal Processing (ICASSP), 2013 IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

[6] A. Graves and N. Jaitly, “Towards end-to-end speech recognition with recurrent neural networks,” in *International Conference on Machine Learning*, 2014.

[7] A. Hannun, A. Maas, D. Jurafsky, and A. Ng, “First-pass large vocabulary continuous speech recognition using bi-directional recurrent DNNs,” in *arXiv*, 2014.

[8] M. Müller, S. Stüker, A. Waibel, “Phonemic and graphemic multilingual CTC based speech recognition,” in *arXiv*, 2017.

[9] A. Graves, “Sequence transduction with recurrent neural networks,” *Computer Science*, vol. 58, no. 3, pp. 235–242, 2012.

[10] H. Sak, M. Shannon, K. Rao, and F. Beaufays, “Recurrent neural aligner: An encoder-decoder neuralnetwork model for sequence to sequence mapping,” in *Proc. of Interspeech*, 2017.

[11] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, “End-to-end Continuous Speech Recognition using Attention-based Recurrent NN: First Results,” in *Neural Information Processing Systems: Workshop Deep Learning and Representation Learning Workshop*, 2014.

[12] J. Chorowski, D. Bahdanau, D. Serdyuk, K. Cho, and Y. Bengio, “Attention-Based Models for Speech Recognition,” in *Neural Information Processing Systems*, 2015.

[13] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, “End-to-end attention-based large vocabulary speech recognition,” in *arXiv*, 2015.

[14] W. Chan, N. Jaitly, Q. Le, and O. Vinyals, “Listen, attend and spell: A neural network for large vocabulary conversational speech recognition,” in *ICASSP. IEEE*, 2016, pp. 4960–4964.

[15] C. Raffel, T. Luong, P. J. Liu, R. J. Weiss, and D. Eck. “Online and linear-time attention by enforcing monotonic alignments,” in *International Conference on Machine Learning*, 2017.

[16] M. Li, M Liu and H. Masanori, “End-to-end speech recognition with adaptive computation steps,” in *arXiv*, 2018.

[17] S. Kim, T. Hori, and S. Watanabe, “Joint CTC-attention based end-to-end speech recognition using multi-task learning,” in *arXiv*, 2016.

[18] 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.

[19] H. Bu, J. Du, X. Na, B. Wu, and H. Zheng, “AIShell-1: An open-source Mandarin speech corpus and a speech recognition baseline,” in *Proc. Oriental COCOSDA*, 2017.

[20] D. Kingma and J. Ba, “Adam: A method for stochastic optimization,” in *arXiv*, 2014.

[21] W. Chan and I. Lane, “On online attention-based speech recognition and joint mandarin character-pinyin training,” *Interspeech*, pp. 3404–3408, 2016.

[22] D. Povey, A. Ghoshal, G. Boulianne, et al, “The Kaldi speech recognition toolkit,” in *Proc. ASRU*, 2011.

[23] N. Ketkar, “Introduction to Pytorch,” in *Deep Learning with Python. Springer*, 2017, pp. 195–208.