Learning Adaptive Downsampling Encoding for Online End-to-End Speech Recognition

Rui Na, Junfeng Hou, Wu Guo, Yan Song and Lirong Dai
National Engineering Laboratory for Speech and Language Information Processing,
University of Science and Technology of China, Hefei, Anhui, China
E-mail: {nrnn, hjf176}@mail.ustc.edu.cn, {guowu, songy, lrdai}@ustc.edu.cn

***Abstract*—Attention based encoder-decoder models have shown promising performance for various sequence-to-sequence problems. However, for speech recognition, the very long input speech consumes a lot of computation and memory resource when performing encoding and soft attention over the input sequence. While fixed-rate downsampling is usually employed to reduce the computation steps, it fails to consider the variable durations of phonemes. Motivated by this, we propose a differentiable adaptive downsampling approach which encodes the input sequence with a recurrent layer by keeping crucial frames and discarding redundant frames adaptively in real-time. Therefore, the proposed downsampling approach can dynamically generate input hidden representations and is suitable for online end-to-end speech recognition. Experiments show that our proposed method can reduce phone error rate (PER) by 7.0% relative without loss of speed compared with fixed downsampling technique. In addition, the adaptive encoding makes the model robust to variable speed speech.**

I. Introduction

Attention based encoder-decoder models [1] are widely applied in various sequence-to-sequence problems and have shown competitive performance for tasks like speech recognition [2]. As a new class of speech recognizer, the attention model combines all components of conventional hybrid systems — acoustic, pronunciation and language models together and is optimized jointly with a single objective. Acoustic models like recurrent neural networks (RNNs) are often employed as encoder of an attention model which encodes the input speech frames into hidden representations. Afterwards, a recurrent decoder equipped with attention mechanism is usually used to produce output sequence based on the input representations. With proper output units like character, word or subword [3], the whole model can be trained in an end-to-end manner.

However, for online speech recognition, the commonly used recurrent encoder updates its hidden state whenever a new frame of the speech sequence is received. This makes the encoder consume a lot of computation and memory resource for very long input speech which may degrade the real-time recognition capability. Further, more resources are demanded while performing soft (online) attention over the encoded input sequence. Meanwhile, for decoder these hidden representations are overly precise and contain much redundant information [4]. Therefore, it is straightforward to introduce downsampling mechanism into encoder so as to reduce resolution and speed up learning and inference.

A solution to reduce computation complexity is to skip some frames that are less important for speech recognition. This so-called frame skipping technique was first proposed in [5] for deep neural network (DNN) acoustic models. In their method, RNN computation is not performed on every speech frame. Instead, a sequence is processed on part of the frames, one frame out of every two, for example, and the predicted label for a non-skipped frame is copied to the next skipped frame. In [6], this frame skipping idea was utilized on connectionist temporal classification (CTC) models with context dependent (CD) phones. Such method was also applied to RNNs in [7]. Similarly, pooling frames over time [4] and pyramidal encoding [8] are also introduced in encoder-decoder models. All these approaches downsample sequences with fixed steps, process less frames in all utterances, and thus decrease processing time.

Although frame skipping methods with a fixed skip rate can accelerate a model, they ignore the variable durations of phonemes in speech signal. As a consequence, some short phonemes may be thoroughly skipped, while some long phonemes still have redundant information. Therefore this method may introduce degradation of recognition performance. To relieve this problem, dynamic frame skipping is intended. Different from static frame skipping that always skips the frames with fixed frame rate, the dynamic frame skipping is intended to skip the speech frames with variable frame rate adaptive to different phonemes, different phoneme durations and speaking speed, etc. In this way, dynamic frame skipping can remain important information and throw away spare frames, decreasing processing time more reasonably.

To the end discussed above, in this paper, we propose an online encoding method which integrates adaptive downsampling mechanism into an encoder layer. The adaptive encoding layer inspects previous layer's output at each time step and determines whether to skip the current frame or not. Although the proposed approach involves hard decisions about skipping redundant frames and keeping crucial frames during inference, these decisions can be trained by expectation over probability instead of sampling, which is a great advantage. In the following paragraphs, we first introduce the related works in Section II. Then a detailed description of our method is presented in Section III. Experiments and discussions about the processing speed, recognition accuracy and speech speed robustness of our method on TIMIT dataset are given in Section IV. Finally, the paper is concluded in Section V.

II. Related work

As a more flexible mechanism, adaptive downsampling is expected to outperform fixed-rate downsampling in terms of model performance and running speed. In [9], a reinforcement learning (RL) enabled skip-policy network is introduced to allow acoustic model to dynamically skip frames. Although recognition accuracy and running speed are improved in their experiments, RL methods usually suffers from high variance [10] and not easy to handle. An algorithm which enables end-to-end speech recognition models to dynamically decide how many frames should be processed to predict a linguistic output is proposed in [11]. The core algorithm used to establish the model, known as adaptive computation steps, is realized by a halting layer, which consists of a CNN and a sigmoidal unit. However, for smaller linguistic units that badly overlap in time, such as phoneme, it is difficult for the algorithm to give explicit alignments of the targets all the time.

Meanwhile, a novel online attention mechanism is recently proposed in [12], yielding an end-to-end differentable method for learning monotonic alignments. During inference, the decoder inspects encoder hidden representations and chooses a single item for output generation at each decoding step. Inspired by this hard monotonic process, we propose a new adaptive downsampling approach which uses standard back-propagation(BP) for training and facilitates dynamic frame skipping for inference.

III. Model

Compared to online attention machanism in [12], our method has some modifications. We realize adaptive downsampling between two encoder layers. This process is nondifferentiable because of sampling, so we also show an algorithm for computing its expected output in order to train our model with standard BP. Furthermore, we apply penalty term to make skipping probalilities more discrete.

A. RNN Layer in the Encoder

We first review the structure of a RNN layer to motivate our approach. In RNNs, recurrent connections between current hidden state and previous hidden state are shared along time, which allows RNNs to capture temporal information of a sequence. In the encoder, we usually use multilayer reucrrent network, in which previous layer output can be passed to the next layer as input. Specifically, given the hidden state h_{i-1}^{l} at timestep $i-1$ and the input h_i^{l-1} at timestep i, we have h_i^l:

$$h_i^l = \mathrm{RNN}(h_{i-1}^l, h_i^{l-1}) \tag{1}$$

B. A Hard Adaptive Downsampling Progress

Different from previous hard monotonic attention work [12] where monotonic process is located between encoder and decoder, our downsampling mechanism is performed between two encoder layers. Therefore, given an output representations $(h_1^{l-1}, ..., h_T^{l-1})$ of encoder layer $l-1$, we need to downsample the sequence adaptively by inspecting h_j^{l-1} and updating hidden state h_i^l of encoder layer l when necessary. Index i and j are timesteps of layer l and $l-1$ respectively . After h_{i-1}^l is updated with $h_{t_{i-1}}^{l-1}$, the downsampling process continues from $h_{t_{i-1}+1}^{l-1}$, where t_i is the index of previous layer states chosen to update h_i^l. For $j = t_{i-1}+1, t_{i-1}+2, \cdots$

$$e_{i,j} = a(h_{i-1}^l, h_j^{l-1}) \tag{2}$$

$$p_{i,j} = \sigma(e_{i,j}) \tag{3}$$

$$z_{i,j} \sim Bernoulli(p_{i,j}) \tag{4}$$

where $a(\cdot)$ is an energy function and $\sigma(\cdot)$ is the logistic sigmoid function. When $z_{i,j} = 1$ for some j, we set $t_i = j$ and then compute h_i^l

$$h_i^l = \mathrm{RNN}(h_{i-1}^l, h_j^{l-1}) \tag{5}$$

We repeat this operation, always starting from $t_{i-1}+1$, until the end of an entry. With this process, adaptive downsampling can be realized during inference with introducing only a small amount of computations in Eqs. (2,3).

C. Training in Expectation

As described in [12], the process above cannot be trained directly with standard backpropagation because of sampling. Therefore the expected value of $h_{t_i}^{l-1}$ is computed for model training.

We compute $e_{i,j}$ and $p_{i,j}$ as in Eqs. (2,3). In order to compute the expected value of $h_{t_i}^{l-1}$, we also need $\alpha_{i,j}$ which can be interpreted as a weighting on the hidden states. Specifically, h_k^{l-1} ($k \in \{1, \cdots, j-1\}$) is selected at timestep $i-1$ and h_j^{l-1} is selected at timestep i instead of $h_{k+1}^{l-1}, \cdots, h_{j-1}^{l-1}$. In consideration of all the values of k, we have

$$\alpha_{i,j} = p_{i,j} \sum_{k=1}^{j-1} \left[\alpha_{i-1,k} \prod_{l=k+1}^{j-1} (1 - p_{i,l}) \right] \tag{6}$$

We define $\alpha_{0,j} = (1-p_{1,1})\delta_j$ (i.e. $\alpha_{0,1} = 1 - p_{1,1}$ and $\alpha_{0,j} = 0$ for $j \in \{2, \cdots, T\}$) and $\prod_n^m x = 1$ when $n > m$ to make Eq. (6) complete. Also we can compute $\alpha_{i,j}$ recursively from $\alpha_{i-1,j-1}$ and $\alpha_{i,j-1}$ as shown in Eq. (7):

$$\begin{aligned} \alpha_{i,j} &= p_{i,j} \left\{ \sum_{k=1}^{j-2} \left[\alpha_{i-1,k} \prod_{l=k+1}^{j-1} (1 - p_{i,l}) \right] + \alpha_{i-1,j-1} \right\} \\ &= p_{i,j} \left[(1 - p_{i,j-1}) \frac{\alpha_{i,j-1}}{p_{i,j-1}} + \alpha_{i-1,j-1} \right] \end{aligned} \tag{7}$$

We can explain Eq. (7) intuitively. The augend $(1 - p_{i,j-1})\, \alpha_{i,j-1}/p_{i,j-1}$ multiplying by $p_{i,j}$ can be comprehended as the probability that the model does not select hidden states item $j-1$ at timestep $i-1$ and could have selected hidden states item $j-1$ at timestep i, but it selects hidden states item j instead. The addend $\alpha_{i-1,j-1}$ multiplying by $p_{i,j}$ represents the probability that the model selects hidden states item $j-1$ at timestep $i-1$ and hidden states item j at timestep i afterwards.

Let $q_{i,j} = \alpha_{i,j}/p_{i,j}$, we have:

$$q_{i,j} = (1 - p_{i,j-1}) q_{i,j-1} + \alpha_{i-1,j-1} \tag{8}$$

Move the augend to the left:

$$q_{i,j} - (1 - p_{i,j-1})q_{i,j-1} = \alpha_{i-1,j-1} \tag{9}$$

Divide the same express $\prod_{k=1}^{j}(1 - p_{i,k-1})$:

$$\frac{q_{i,j}}{\prod_{k=1}^{j}(1 - p_{i,k-1})} - \frac{q_{i,j-1}}{\prod_{k=1}^{j-1}(1 - p_{i,k-1})} = \frac{\alpha_{i-1,j-1}}{\prod_{k=1}^{j}(1 - p_{i,k-1})} \tag{10}$$

Add up the equations from index 1 to j:

$$\sum_{l=1}^{j}\left(\frac{q_{i,l}}{\prod_{k=1}^{l}(1 - p_{i,k-1})} - \frac{q_{i,l-1}}{\prod_{k=1}^{l-1}(1 - p_{i,k-1})}\right) = \sum_{l=1}^{j}\frac{\alpha_{i-1,l-1}}{\prod_{k=1}^{l}(1 - p_{i,k-1})} \tag{11}$$

Transpose the terms:

$$q_{i,j} = \left(\prod_{k=1}^{j}(1 - p_{i,k-1})\right)\left(\sum_{l=1}^{j}\frac{\alpha_{i-1,l-1}}{\prod_{k=1}^{l}(1 - p_{i,k-1})}\right) \tag{12}$$

Note that we define $q_{i,0} = 0$ and $p_{i,0} = 0$ in this process. Now we can compute $q_{i,j}$ and $\alpha_{i,j}$ ($j \in \{1, \cdots, T\}$) in parallel. With $\alpha_{i,j}$ we can then compute the expectation of $h_{t_i}^{l-1}$:

$$c_i = \sum_{k=1}^{T}\alpha_{i,k}h_k^{l-1} \tag{13}$$

Then we have h_i^l:

$$h_i^l = RNN(h_{i-1}^l, c_i) \tag{14}$$

In order to realize adaptive downsampling with a factor of 2, we restrain the value of i from 1 to T/2 and j from 1 to T in the training process.

D. Penalty Term

In the proposed downsampling approach, the probability of choosing an index in the hidden states $p_{i,j}$ is expected to be discrete, approximately 0 or 1 at best. So we add the entropy into our loss function as a penalty term in order to make the probabilities closer to 0 or 1. Specifically, for the probability of choosing a frame p, we can compute the entropy as follow:

$$\begin{aligned} H(U) &= E\left[-\log p_i\right] \\ &= -p\log p - (1-p)\log(1-p) \end{aligned} \tag{15}$$

IV. Experiments

A. Data

We trained and evaluated our models on the TIMIT corpus [13]. We used the train-dev-test split from the Kaldi [14] TIMIT s5 recipe. Training was done on the standard 462 speaker set with all SA utterances removed. We used the 50 speaker dev set for early stopping, and tested on the 24 speaker core test set. As input features, we used 40 mel-scale filterbank coefficients together with the energy in each frame, and first and second temporal differences, yielding in total 123 features per frame. In the training set, each feature was rescaled to have zero mean and unit variance. To indicate the end of the utterance, we concatenated an all-zero frame at the end of each input sequence. As training targets, we used the full 61-phone set with an extra end-of-sequence token that was appended to each target sequence. We reported the phone error rate (PER) after applying the standard mapping to 39 phonemes.

TABLE I
PER of different downsampling approaches on TIMIT corpus.

Downsampling	Test PER(%)	Time(sec)
Fixed	22.44	4.62
Adaptive	20.86	4.80

Also, our experiments involved different speeds of datasets. We changed the speed with STRAIGHT. STRAIGHT (Speech Transformation and Representation using Adaptive Interpolation of weiGHTed spectrum) is basically a channel vocoder [15], used as a speech analysis, modification synthesis system.

B. Training

Our model used a 3-layer RNN with 256 GRU units as encoder, and the activations of the 256 top-layer units were used as the representation h. In the baseline system, the output of each layer was downsampled with a fixed factor of two, yielding totally 1/8 downsampling rate. In our proposed approach, adaptive downsampling was used between the second and third layer, and we trained it to achieve approximately equal downsampling ratio with baseline (see section III). A windowing approach [4] was used to realize an online system. We set the length of window to 20 to make the time delay of our online system small enough. Note that this attention mechanism can be seen as an online strategy. Though it is not the advanced online attention, we used it for fundamental evaluation and will consider more effective online methods in our future work. For the decoder, we used a single unidirectional GRU layer with 256 units, fed directly into the output softmax layer. The output tokens were embedded via a learned embedding matrix with dimensionality 30. Our attention energy function used a hidden dimensionality of 512. Adam optimizing algorithm was used in our experiments with batch size set to 16. Learning rate was set to 1e-3 in the beginning and decays to 1e-4 and 5e-5 respectively when no PER improvement is found on valid set. Also, a dropout rate of 0.3 was used to prevent our neural networks from overfitting.

C. Results

We first evaluate our method on TIMIT to compare the performance of our adaptive downsampling method with fixed downsampling selecting every eighth of hidden states of the below layer. Our experiments show that our approach decreases the error rate of the model without significantly increasing the process time. Table 1 shows detailed experiment results, including phone error rates (PERs) and time for the decoding on the test set. In the proposed method, a 7% relative PER reduction is achieved. To demonstrate that our adaptive downsampling has the ability to select crucial frames, we show a test set example of the choices of frames by the adaptive downsampling layer and boundaries of true labels in Fig. 1.

TABLE II
PER OF DIFFERENT SPEED TEST SETS.

PER(%)	Mismatched speed						Matched speed					
	0.8	0.9	1.0	1.1	1.2	Average	0.8	0.9	1.0	1.1	1.2	Average
Fixed	29.65	26.57	22.44	28.02	30.23	27.38	24.31	23.95	24.19	24.30	24.98	24.34
Adaptive	26.10	24.12	20.86	25.11	26.64	24.56	22.58	22.16	22.68	23.40	23.33	22.83
True ratio	10.23%	11.17%	11.88%	12.76%	13.44%	11.90%	10.74%	11.78%	12.75%	13.63%	14.43%	12.67%

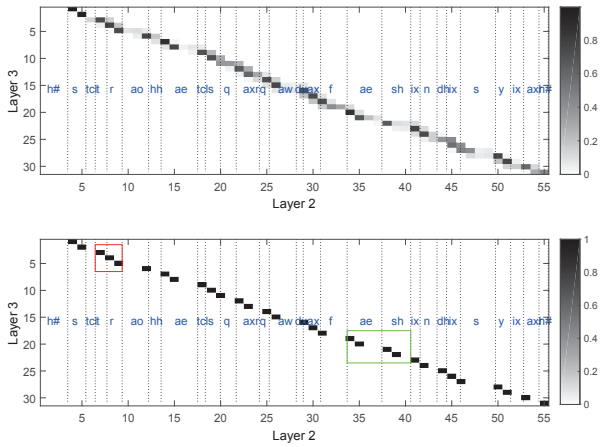

Fig. 1. The alignments of selected frames and the boundaries of ground truths.

Alignments, which we refer to attention on the second layer by the third layer here but not attention in decoder, are drawn in the way we can visualize attention. The y-coordinate and x-coordinate represent timesteps in the adaptive downsampling encoder layer and the layer below it seperately. The vertical lines are boundaries of true labels in the utterance. In the upper figure we show the expectation of soft alignments computed in the training process, while hard alignments are drawn in the lower figure. We can see that alignments crowd together where the corresponding phonemes are shorter such as the phonemes in the red rectangle, and skip further where the phonemes are longer as shown in the green rectangle. This phenomenon proves that the model can tell the similarity among continuous frames and has more attention on the crucial ones.

Furthermore, we wonder whether the proposed approach is adaptive to different speeds of speaking or not. We change the speed of datasets and do experiments on them. In detail we seperately change the datasets into 0.8, 0.9, 1.1, 1.2 times of original speed. We first directly test our model which is trained with original training set on test sets of all five speeds. In this way the speed of test data is mismatched with training set. The results are shown in the left half of Table 2, which prove that the proposed method is somewhat adaptive to different speeds of speaking without seeing them during training. Our PER is improved by 10.3% relative on average. Moreover, considering the demands in real life that people's speaking speeds are quite different, we then train the model with training sets of all five speeds, i.e. data augmentation. As shown in the right half of Table 2, the accuracies of varied speed speech recognition are better than the former experiments without date augmentation, and have improvements compared with the fixed model, too, by 6.2% relative on average. We also count the ratio of the number of selected frames in adaptive downsampling to the number of all frames of each speed (true ratio is 0.125 for fixed downsampling). As in the last line of Table 2, when the speech is slower, the sentence length becomes longer and the ratio is smaller. Also when the speech is faster, the ratio is larger. This trend proves the adaptive ability of our approach.

V. Conclusions

In this paper, we aim to present an adaptive downsampling method in a RNN-based sequence-to-sequence model on speech recognition. We have realized this approach and proved that it not only improves the performance of speech recognition, but has comparable speed with fixed downsampling method as well. The approach can be trained with standard BP and sample crucial frames when decoding. Moreover, we show that this method has the ability to adapt sentences of different speeds. We do our experiments on both the original dataset and augmented dataset, and prove that the performances are impoved in both conditions.

We have validated our methed on the TIMIT corpus. Next we plan to evaluate its performance on larger datasets. Furthermore, we used a windowing soft attenion mechanism to decode online instead of using more effective online attentions. This is likely to be a part of our future work as well.

Acknowledgment

This work was supported by National Key R&D Program of China (Grant No. 2017YFB1002202) and the Key Science and Technology Project of Anhui Province (Grant No. 18030901016). The experiments were conducted using Tensorflow libraries [17].

References

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

[2] 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.

[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] 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.

[5] V. Vanhoucke, M. Devin, and G. Heigold, "Multiframe deep neural networks for acoustic modeling," in *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*. IEEE, 2013, pp. 7582–7585.

[6] H. Sak, A. Senior, K. Rao, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," *INTERSPEECH*, 2015.

[7] Y. Miao, J. Li, Y. Wang, S.-X. Zhang, and Y. Gong, "Simplifying long short-term memory acoustic models for fast training and decoding," in *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2016, pp. 2284–2288.

[8] W. Chan, N. Jaitly, Q. V. Le, and O. Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[9] 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.

[10] E. Greensmith, P. L. Bartlett, and J. Baxter, "Variance reduction techniques for gradient estimates in reinforcement learning," *Journal of Machine Learning Research*, vol. 5, no. Nov, pp. 1471–1530, 2004.

[11] M. Li, M. Liu, and H. Masanori, "End-to-end speech recognition with adaptive computation steps," in *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2019, pp. 6246–6250.

[12] C. Raffel, M.-T. Luong, P. J. Liu, R. J. Weiss, and D. Eck, "Online and linear-time attention by enforcing monotonic alignments," in *Proceedings of the 34th International Conference on Machine Learning-Volume 70*. JMLR. org, 2017, pp. 2837–2846.

[13] J. S. Garofalo, L. F. Lamel, W. M. Fisher, J. G. Fiscus, D. S. Pallett, and N. L. Dahlgren, "The darpa timit acoustic-phonetic continuous speech corpus cdrom," *Linguistic Data Consortium*, 1993.

[14] 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," IEEE Signal Processing Society, Tech. Rep., 2011.

[15] H. Kawahara, "Straight, exploitation of the other aspect of vocoder: Perceptually isomorphic decomposition of speech sounds," *Acoustical science and technology*, vol. 27, no. 6, pp. 349–353, 2006.

[16] D. B. Paul and J. M. Baker, "The design for the wall street journal-based csr corpus," in *Proceedings of the workshop on Speech and Natural Language*. Association for Computational Linguistics, 1992, pp. 357–362.

[17] S. S. Girija, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *Software available from tensorflow. org*, 2016.