INTERSPEECH 2016
September 8–12, 2016, San Francisco, USA

Towards Online-Recognition with Deep Bidirectional LSTM Acoustic Models

Albert Zeyer, Ralf Schlüter, Hermann Ney

Human Language Technology and Pattern Recognition, Computer Science Department,
RWTH Aachen University, 52062 Aachen, Germany
{zeyer, schlueter, ney}@cs.rwth-aachen.de

Abstract

Online-Recognition requires the acoustic model to provide posterior probabilities after a limited time delay given the online input audio data. This necessitates unidirectional modeling and the standard solution is to use unidirectional long short-term memory (LSTM) recurrent neural networks (RNN) or feed-forward neural networks (FFNN).

It is known that bidirectional LSTMs are more powerful and perform better than unidirectional LSTMs. To demonstrate the performance difference, we start by comparing several different bidirectional and unidirectional LSTM topologies.

Furthermore, we apply a modification to bidirectional RNNs to enable online-recognition by moving a window over the input stream and perform one forwarding through the RNN on each window. Then, we combine the posteriors of each forwarding and we renormalize them. We show in experiments that the performance of this online-enabled bidirectional LSTM performs as good as the offline bidirectional LSTM and much better than the unidirectional LSTM.

1. Introduction

Recently, deep bidirectional LSTM based acoustic models have been shown to yield state-of-the-art results in speech recognition [1, 2, 3]. It is known that bidirectional LSTMs are more powerful than unidirectional LSTMs [4] and also much more powerful than FFNNs [1]. Unidirectional LSTMs and sometimes even FFNNs are however often used in production systems because only these seem to allow online recognition in a straight-forward way.

We do some comparison between unidirectional and bidirectional LSTMs and show their great gap in performance. To investigate this further, we do several experiments on other similar topology variants. This leads to the conclusion that it is favourable to use bidirectional LSTMs.

Our main motivation for this work was to develop a method which enables to do online recognition also with a bidirectional LSTMs acoustic model. We present our method and we show that this method is as good as offline bidirectional LSTMs. We need to add a small delay of less than 1 second to achieve the best performance. The method needs some additional effort in computation for decoding which however can be easily parallelized. The method can be seen as an extended acoustic model which embeds a conventional acoustic model. The embedded acoustic model will then estimate posteriors only on the windows. The embedded acoustic model can be trained in a conventional way or it can be trained as part of the extended model, which would be more computationally expensive. In our experiments, we use a conventionally trained model.

In this work, we concentrate on the online-ability of the acoustic model only. To do online decoding, one would also need the feature extraction pipeline to work online. In our experiments, we use an offline feature extraction pipeline and the features are normalized segment-wise. We did not change our feature extraction to focus the comparisons on the differences introduces by the corresponding LSTM topologies.

2. Related work

[4] is an early work where unidirectional and bidirectional RNNs and LSTMs are compared and it is shown that bidirectional models outperform unidirectional ones.

In [5], a similar idea of a sliding window over the input features is introduced. The authors focus on the comparison to windowed input in a FFNN which approximates the posteriors of the center frame of the window. Thus, in their comparison, the bidirectional LSTM also only approximates the posteriors of the center frame of the given window. This is different from our work where the bidirectional LSTM approximates the posteriors for all frames of the window. The approach in [5] requires that the window is moved on a frame by frame basis, to get posteriors for all frames of the whole input, and it also needs some initial padding for the initial windows of the input. The authors focus on the question whether the LSTM/RNN gains performance when using longer context like the whole input sequence or if a window is enough. Their conclusion is that at least for their experiments, the LSTM performed just as good when it is used on a window as long as the window is large enough. Note that they did not use a conventionally trained acoustic model for their method but they always trained new models to keep training and recognition more consistent. Note that training such a model is much more computationally expensive than training a conventional model. Our work shows that we can train a model in a conventional way and still use it for online decoding.

Recently, end-to-end sequence-to-sequence models for speech recognition were proposed [6, 7]. In [8], a method is developed to enable online decoding with such a model by operating on blocks of data. In this work, the encoder is an unidirectional RNN and the blocks are non-overlapping windows. The transducer is also a unidirectional RNN and operates on each block. Also, the recurrent state is fed from one block into the next in both the encoder and the transducer. We don't do this in our work and there is also no straight-forward way to do this in our approach, but it might be an interesting extension. To enable online decoding in encoder-transducer approaches, the encoder needs to operate online, i. e. a unidirectional RNN or the approach presented in our work can be applied. The attention model also cannot operate globally in online recognition and if the history of past encoder output is restricted, then the attention must be local and monotonic. The approach in [9] penalizes non-monotonic soft alignments but does not enforce them in the model. A local attention mechanism is proposed

http://dx.doi.org/10.21437/Interspeech.2016-759

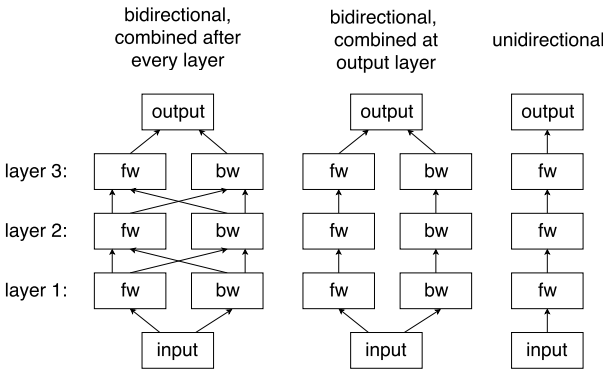

Figure 1: *Bidirectional vs. unidirectional topologies.*

in [10] but the corresponding monotonic local attention variant is just a linear alignment and the other local attention variant presented is not monotonic. A local monotonic attention like in [11] would allow online recognition.

In [12, 13] it is argued that a time delay neural network can be used instead of an RNN to model long term temporal dependencies. Due to its feed-forward nature, this can also be applied for online decoding.

3. Unidirectional vs. Bidirectional LSTMs

Unidirectional LSTMs seem to be a natural choice to enable online decoding. In this work, we want to confirm earlier results that bidirectional LSTMs outperform unidirectional ones. Thus we compare unidirectional LSTMs with bidirectional LSTMs and a few other related topologies.

When having deep bidirectional models, one can combine the forward and backward directions after each layer or only at the output layer. We compare both topologies (see Figure 1) and we see some noticeable degradation when the combination is only in the output layer (see Table 1). One third comparison was to add the forward and backward LSTM outputs and divide by 2, i. e. average them. This can be seen as a variant of combination after every layer with some dimension reduction. This reduces the number of parameters even more than the model where we combine only in the output layer and is also slightly better. This leads to the conclusion that the combination after every layer overall is better than combining only in the output layer.

Our initial unidirectional baseline experiment has very similar parameters than the bidirectional model except that we have only one forward LSTM with 500 cells in every layer. As expected, this performs much worse than the bidirectional model. We also did one experiment with a backward unidirectional LSTM which was surprisingly a bit better than the forward unidirectional LSTM. In [4], a similar result for a backward LSTM was obtained. It can be argued that the comparison is not fair because this model has less parameters. To consider this, we did an experiment using two separate LSTMs per layer, which are combined after each layer in the same fashion as the bidirectional baseline model which results exactly in the same number of parameters as the bidirectional baseline model. The result shows that it gives only some improvement though still being much worse than the bidirectional model.

In the unidirectional case, we can argue that one big disadvantage is that we have no context information of future frames. Nevertheless, as shown in [4], introducing a certain amount of local future context/delay does improve performance to some extend. Currently, we perform ongoing experiments to confirm these results.

Table 1: *Comparison between different bidirectional and unidirectional topologies. 3 layers, hidden layer size 500, Adadelta with WER reported on eval10. More details in Section 3. Setup described in Section 5.*

dir.	topology	#params[M]	WER[%]
bidir	combine after every layer	18.7	**15.1**
	combine at output layer	14.7	16.0
	average after every layer	12.5	15.8
unidir	forward	7.4	19.6
	backward	7.4	19.3
	two separate LSTMs per layer	18.7	19.1

4. Enabling Online Recognition with Bidirectional LSTMs

In the previous section, we outlined the potential advantage bidirectional LSTMs have over unidirectional ones by fully exploiting the complete input for each frame. The motivation is thus to use bidirectional LSTMs also for online decoding. This is not possible with the conventional bidirectional approach using the complete input of a segment. The basic idea in this work was to operate always only on a fixed window, thus the future context is limited and we can use it for online decoding.

We use an existing bidirectional model to operate on a given window to estimate posteriors for each frame of the window. Then we move the window a few frames forward and repeat. We can get overlapping windows with this approach and we had the intuition that multiple posterior estimations for a given time frame from overlapping windows will only help. We will see in our experiments in Section 6 that this is the case.

We can use any existing already trained bidirectional LSTM model. Let $x_1^\infty \in \mathbb{R}^{\mathbb{N}\times D}$ be the possibly infinite input feature stream and D is the input observation vector dimension.

Given the input sequence x_1^∞, we move a window over it with size T_w and move the window always by T_s steps, i. e. the i-th window is $X_i := x_{1+i\cdot T_s}^{T_w+i\cdot T_s}$ with $i \in \mathbb{N}_0$. This means that we have up to $\lceil T_w/T_s \rceil$ windows at some time frame which can overlap. For each window X_i, we do one evaluation with the bidirectional LSTM and we get some estimated posterior probabilities $p_t(s|X_i)$ for every state s for each time frame t in the window given the window X_i. We then combine all these outputs by averaging over all overlapping windows for a given time frame, optionally with some weighting.

The recurrent initial states for each window are always initialized with zero in our experiments. As the windows are usually overlapping and also not necessarily aligning with past windows, there is no straight-forward way to initialize at least the forward LSTMs with states from past windows. If $T_w \equiv 0 \pmod{T_s}$, we could use the last aligning window but we did not do this experiment.

Let $W_\tau(t) \in \mathbb{R}_{\geq 0}$ be the weighting for time frame $t \in \{\tau+1, \ldots, \tau+T_w\}$ in the window $x_{\tau+1}^{\tau+T_w}$. We estimate the acoustic model posterior probability $q_t(s|x_1^\infty)$ to be in state s in time frame t given the input sequence as

$$q_t(s|x_1^\infty) = \frac{\sum_{i=0}^{\infty} W_{i\cdot T_s}(t) \cdot p_t(s|X_i)}{\sum_{i=0}^{\infty} W_{i\cdot T_s}(t)} \tag{1}$$

where $p_t(s|X_i)$ is the state posterior probability estimated by the bidirectional LSTM which operates on the window X_i.

This means, to evaluate the posterior probability $q_t(s|x_1^\infty)$ at some time frame, we have to look up to $T_w - 1$ frames into the future. In an online recognizer, this would be some added delay. Note that all $p_t(s|X_i)$ can be calculated in parallel for every window X_i.

We have the intuition that the bidirectional LSTM can better estimate the posteriors for frames in the center. Thus, we applied different weightings across the frames in one window:

- Uniform: $W_\tau(t) = 1$.
- Triangle: $W_\tau(t) = 1 + \min\{t - \tau - 1, \tau + T_w - t\}$.
- Hamming: $W_\tau(t) = \alpha - (1-\alpha)\cos\left(\frac{2\pi(t-\tau-1)}{T_w-1}\right)$ with $\alpha = 0.53836$.
- Gauss: $W_\tau(t) = \exp\left(-\frac{1}{2}\left(\frac{(t-\tau-1)-(T_w-1)/2}{\sigma(T_w-1)/2}\right)^2\right)$ with $\sigma \leq 0.5$.

Note that the approach in [5] can be seen in this formulation as a weighting where only the center frame is 1 and we have 0 everywhere else.

5. Experimental Setup

We use a subset of 50 hours from the Quaero English database *train11* [14] to train our bidirectional LSTM based acoustic model. The development *eval10* and evaluation *eval11* sets consist of about 3.5 hours of speech each. The recognition is performed using a 4-gram language model.

The input features are globally mean- and variance normalized 50-dimensional VTLN-normalized Gammatone features [15]. We don't add any context window nor delta frames.

One minibatch for the LSTM network training consists of several chunks, i. e. several parts of a sequence. From the corpus sequences, every t_{step} frames, we select a chunk of up to T frames, until we have n_{chunks} number of chunks. This minibatch construction is similar to e. g. [16] and described in more detail in [3]. Our baseline model was trained with $T = 50, t_{\text{step}} = 25, n_{\text{chunks}} = 40$, i. e. with a minibatch size of 2000 frames. Note that this scheme is similar to the online recognition scheme but this is no requirement in the training. In [3], we did several experiments with different minibatch and chunk sizes and we usually don't see much improvement for longer chunks, which is analogous to the results presented in this work for the recognition part.

More details about the setup and a comprehensive study of different variants of LSTM topologies, hyperparameters and different optimization methods can be seen in [3].

Our baseline acoustic model is a 3 layer bidirectional LSTM with 500 cells each for the forward and the backward direction in every layer which is combined after every layer. We use dropout and L_2 regularization and Adam [17] for optimization in addition to learning rate scheduling as described in [3]. Our model has 1.87M parameters and one training epoch takes 1:22h on a GeForce GTX 980 with RETURNN, our Theano-based training framework as described in [18, 19]. On *eval10*, we get a WER of 13.7% with conventional offline recognition. Note that this has been improved over the model reported in section 3.

6. Experimental Results

All recognition experiments were done with the same acoustic model, which we described in the previous section and we always report the WER on *eval10*. Note that the features still are normalized segment-wise, so the feature set is not yet online-enabled. However, this allows a consistent comparison with the offline modeling performance.

We are first interested in the effect of the window size T_w, i. e. how much context do we need. We compare different window sizes T_w in Table 2. Note that 100 frames correspond to 1 second of audio. As expected, bigger windows yield better performance. We see that with this setting, we can reach the original performance. Note that we have quite a few overlapping windows in this setting.

Table 2: *Comparing window sizes T_w, $T_s = 10$, uniform distribution. We also note the number of overlapping windows.*

T_w	#windows	WER[%]
20	2	16.3
40	4	14.1
60	6	13.9
80	8	13.9
100	10	**13.7**

Then, for a given window size, we compare different window steps T_s in Table 3. It is interesting to see the big impact on the performance. We interpret that there is some variability in each forwarded window and we gain some information by combining them. To further analyze this effect, we compare the number of overlapping windows in Table 4. We conclude that there is the clear trend that more windows are better, provided the windows itself are not too short. Thus, $T_s = 1$ is likely the best choice, although this comes at a performance cost which is parallelizable but we keep using wider window steps for simplicity.

Table 3: *Comparing window steps T_s, $T_w = 40$, uniform distribution. We also note the number of overlapping windows.*

T_s	#windows	WER[%]
1	40	**13.7**
5	8	13.9
10	4	14.1
20	2	15.4

Table 4: *Comparing number of overlapping windows $\lceil T_w/T_s \rceil$, uniform distribution.*

#windows	T_w	T_s	WER[%]
2	20	10	16.3
	40	20	15.4
4	40	10	14.1
8	40	5	13.9
10	10	1	18.5
	40	4	13.7
	100	10	13.7
20	100	5	**13.6**
40	40	1	13.7

So far, we have only used a uniform weight distribution for the averaging over windows. It is conceivable that the frames in the center of a window might provide better posterior estimates. Thus we compare different frame posterior weight distributions in Table 5 and we get to the conclusion that the performances are very similar and the triangle distribution seems to perform slightly better than the others, also better than Hamming and

Gauss, even though differences are not significant.

Table 5: *Comparing distributions.*

T_w	T_s	distribution	WER[%]
40	5	uniform	13.9
		triangle	13.8
		hamming	13.8
		gauss $\sigma = 0.4$	13.8
	4	uniform	13.7
		triangle	13.7
100	5	uniform	**13.6**
		triangle	**13.6**
		hamming	13.7
		gauss $\sigma = 0.4$	13.7

We have seen that we can reach the original (offline) performance, the more windows the better, and the triangle distribution might be slightly better. With these settings, the optimization over window sizes was repeated, cf. Table 6. Interestingly, we get even slightly better than our baseline and it seems as if 50 frames (half a second) are enough to get the best result.

Table 6: *Comparing window size T_w. $T_s = 5$, triangle distribution.*

T_w	WER[%]
40	13.8
50	**13.6**
80	**13.6**
100	**13.6**

Another idea was that for each forwarding, we could enlarge the window to add more left-context, i. e. C_l additional frames, because we have that audio input anyway and the bidirectional LSTM might get better context information from it. We would only use the LSTM posterior estimations which are not part of the added left context. Our input window actually is of size $T_w + C_l$ then. We compare different settings in Table 7. For very short windows, as expected, it helps but not as much as having a bigger window in the first place. For bigger windows, an additional left context doesn't help and unexpectedly even seem to slightly degrade the performance.

Table 7: *Comparing added left context.*

T_w	T_s	distribution	left context C_l	WER[%]
10	1	triangle	0	18.3
			100	15.1
40	10	uniform	0	14.1
			100	14.1
		triangle	0	14.1
			100	14.2
	5	triangle	0	**13.8**
			100	13.9

Note that an equivalent formulation would be $T'_w = T_w + C_l$ and $W'_\tau(t) = 0$ for $t \leq \tau + C_l$ and $W'_\tau(t) = W_{\tau+C_l}(t)$ otherwise, although the number of overlapping windows considering the posterior estimations is less as well as the needed delay in online recognition.

Analogously we can add more right context. When we keep the window size T'_w fixed and we have both left and right context with $W'_\tau(t) = 0$, we effectively reduce the amount of overlapping posterior estimations. The extreme case where $W'_\tau(t) \neq 0$ only for the center frame is exactly the setting as in [5] and results in no overlaps at all. We did one experiment with a window size $T'_w = 49$ where we only take out the center frame posterior estimation and we get a WER of 14.0%, compared to $T_w = 50, T_s = 5$ and the triangle distribution on the full window where we get a WER of 13.6%. We conclude that the overlapping posterior estimations contribute to richer information density.

As outlined in Section 1, we can train also directly the extended acoustic model to have the training consistent with the recognition. In [5], they only use models trained in this scheme and did not try any conventionally trained models. Our Theano-based framework RETURNN [18, 19] allows for straightforward training of the extended models, although it is very slow in our current implementation and we didn't get any interesting results so far.

6.1. Experiments on Switchboard

We use the 300h Switchboard-1 Release 2 (LDC97S62) corpus for training and the Hub5'00 evaluation data (LDC2002S09) is used for testing. We report the WER for the Switchboard (SWB) and CallHome (CH) parts separately. We use a 4-gram language model which was trained on the transcripts of the acoustic training data (3M running words) and the transcripts of the Fisher English corpora (LDC2004T19 & LDC2005T19) with 22M running words. More details can be found in [20].

A good FFNN baseline yielded a total WER of 19.1% (13.1% WER on SWB and 25.6% WER on CH). We trained a 5 layer bidirectional LSTM and obtained a total WER of 17.1% (11.9% WER on SWB and 22.3% WER on CH). This is the model which we used for the online recognition experiments.

We did one recognition with $T_w = 100, T_s = 5$ with the triangle distribution and got 17.4% WER in total (11.8% WER on SWB and 23.1% WER on CH). The same setup with $T_s = 4$ yielded a total WER of 17.3% WER (11.6% WER on SWB and 23.0% WER on CH). We see the same effect as earlier that more overlapping windows improve the performance. Interestingly, we get even better on the SWB part than our baseline but we are a bit worse on the CH part.

7. Conclusions & Outlook

We presented a method which enables online recognition with bidirectional LSTM acoustic models. We demonstrated that this method yields comparable recognition performance. The model can be trained in a conventional setting, although online-enabled feature normalization is required to get a true online recognition setup.

This method can easily be combined with the attention based approach in [8] and also the CTC approach in [21] which are interesting directions for further research.

8. Acknowledgements

We thank Zoltán Tüske for the baseline FFNN Switchboard experiment.

This research was partially supported by Ford Motor Company and the Deutsche Forschungsgemeinschaft (DFG) under Contract No. Schl2043/11-1.

This work has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No. 644283.

9. References

[1] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition," *arXiv preprint arXiv:1402.1128*, 2014, http://arxiv.org/pdf/1402.1128.

[2] J. T. Geiger, Z. Zhang, F. Weninger, B. Schuller, and G. Rigoll, "Robust speech recognition using long short-term memory recurrent neural networks for hybrid acoustic modelling," in *INTERSPEECH*, 2014, pp. 631–635.

[3] A. Zeyer, P. Doetsch, P. Voigtlaender, R. Schlüter, and H. Ney, "A comprehensive study of deep bidirectional LSTM RNNs for acoustic modeling in speech recognition," *work in preparation, can be requested from the authors*, 2016.

[4] A. Graves and J. Schmidhuber, "Framewise phoneme classification with bidirectional lstm and other neural network architectures," *Neural Networks*, vol. 18, no. 5, pp. 602–610, 2005.

[5] A.-r. Mohamed, F. Seide, D. Yu, J. Droppo, A. Stolcke, G. Zweig, and G. Penn, "Deep bi-directional recurrent networks over spectral windows," in *Proc. IEEE Automatic Speech Recognition and Understanding Workshop*, ASRU. IEEE – Institute of Electrical and Electronics Engineers, December 2015, pp. 78–83. [Online]. Available: http://research.microsoft.com/apps/pubs/default.aspx?id=259236

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

[7] D. Bahdanau, J. Chorowski, D. Serdyuk, P. Brakel, and Y. Bengio, "End-to-end attention-based large vocabulary speech recognition," *arXiv preprint arXiv:1508.04395*, 2015.

[8] N. Jaitly, Q. V. Le, O. Vinyals, I. Sutskeyver, and S. Bengio, "An online sequence-to-sequence model using partial conditioning," *arXiv preprint arXiv:1511.04868*, 2015.

[9] J. Chorowski, D. Bahdanau, K. Cho, and Y. Bengio, "End-to-end continuous speech recognition using attention-based recurrent nn: first results," *arXiv preprint arXiv:1412.1602*, 2014.

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

[11] A. Graves, "Generating sequences with recurrent neural networks," *arXiv preprint arXiv:1308.0850*, 2013, http://arxiv.org/pdf/1308.0850.

[12] V. Peddinti, D. Povey, and S. Khudanpur, "A time delay neural network architecture for efficient modeling of long temporal contexts," in *Proceedings of INTERSPEECH*, 2015.

[13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *Acoustics, Speech and Signal Processing, IEEE Transactions on*, vol. 37, no. 3, pp. 328–339, 1989.

[14] M. Nußbaum-Thom, S. Wiesler, M. Sundermeyer, C. Plahl, S. Hahn, R. Schlüter, and H. Ney, "The RWTH 2009 Quaero ASR evaluation system for English and German," in *Interspeech*, Makuhari, Japan, Sep. 2010, pp. 1517–1520.

[15] R. Schlüter, L. Bezrukov, H. Wagner, and H. Ney, "Gammatone features and feature combination for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing, 2007. ICASSP 2007. IEEE International Conference on*, vol. 4. IEEE, 2007, pp. IV–649.

[16] X. Li and X. Wu, "Constructing long short-term memory based deep recurrent neural networks for large vocabulary speech recognition," in *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*. IEEE, 2015, pp. 4520–4524.

[17] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[18] P. Doetsch, A. Zeyer, P. Voigtlaender, I. Kulikov, R. Schlüter, and H. Ney, "RETURNN: The RWTH extensible training framework for universal recurrent neural networks," *work in preparation, can be requested from the authors*, 2016.

[19] "RETURNN: The RWTH extensible training framework for universal recurrent neural networks," http://www-i6.informatik.rwth-aachen.de/, https://github.com/rwth-i6/returnn, 2016.

[20] Z. Tüske, P. Golik, R. Schlüter, and H. Ney, "Speaker adaptive joint training of gaussian mixture models and bottleneck features," in *IEEE Automatic Speech Recognition and Understanding Workshop*, Scottsdale, AZ, USA, Dec. 2015, pp. 596–603.

[21] K. Hwang and W. Sung, "Character-level incremental speech recognition with recurrent neural networks," *arXiv preprint arXiv:1601.06581*, 2016.