WINDOWED ATTENTION MECHANISMS FOR SPEECH RECOGNITION

Shucong Zhang, Erfan Loweimi, Peter Bell, Steve Renals

Centre for Speech Technology Research, University of Edinburgh, Edinburgh, UK

ABSTRACT

The usual attention mechanisms used for encoder-decoder models do not constrain the relationship between input and output sequences to be monotonic. To address this we explore windowed attention mechanisms which restrict attention to a block of source hidden states. Rule-based windowing restricts attention to a (typically large) fixed-length window. The performance of such methods is poor if the window size is small. In this paper, we propose a fully-trainable windowed attention and provide a detailed analysis on the factors which affect the performance of such an attention mechanism. Compared to the rule-based window methods, the learned window size is significantly smaller yet the model's performance is competitive. On the TIMIT corpus this approach has resulted in a 17% (relative) performance improvement over the traditional attention model. Our model also yields comparable accuracies to the joint CTC-attention model on the Wall Street Journal corpus.

Index Terms— end-to-end, speech recognition, attention

1. INTRODUCTION

Attention-based encoder-decoder models have offered impressive performance gains in tasks such as image caption and machine translation [1, 2, 3]. However, for speech recognition, without a large amount of training data, the attention-based encoder-decoder model often does not outperform traditional models [4].

A weakness of this model for speech recognition is that the attention mechanism does not guarantee a monotonic alignment between the input and output sequences, which usually follow a left-to-right order. Since the attention mechanism relies heavily on the content of the hidden representations of the input units, a monotonic alignment is easily corrupted by similar input speech fragments or noise. Furthermore, the attention mechanism considers all input frames when calculating the alignment vector. However, since one output unit usually only corresponds to a small input time span, it is arguably unnecessary to consider the entire input sequence when estimating the alignment.

Several mechanisms have been proposed for *location-aware* or *coverage-aware* attention. At a decoding time step, the attention vector at the previous time step or the alignments of all previous output units are exploited to generate the current attention vector [5, 6]. Thereby, the attention mechanism has an awareness of the location or the coverage [7]. However, this approach does not guarantee the best performance [8].

Other methods which add a location constraint to the attention mechanism include a model in which both connectionist temporal classification (CTC) and an attention model share the same encoder – joint CTC-attention [9, 10]. In this case, CTC provides a soft left-to-right constraint. Windowing methods put a hard location constraint on the attention mechanism, restricting attention to a window of inputs [8, 11, 12, 13, 14]. When the method of moving the window is rule-based, it is necessary to set the window size to be large (typically comparable to the utterance length) for accurate results [8, 11, 12, 13]. When the step size is learned, a short fixed-size window can yield good results [14]. There is no analysis on why with a trained step size small window can suffice. Furthermore, the window size is not trainable in these methods.

Inspired by the local attention mechanism [3], this paper proposes a novel fully-trainable windowed attention for speech recognition. Within the window, a Gaussian distribution or a concatenation of two sigmoid functions are used to predict a location score. We also employ a model to predict the step size and window size, consequently make these two window parameters trainable. To generate content scores, we used content-based attention. In the proposed approach the final attention score was the product of the location score and the content score. Compared to rule-based window methods [8, 11, 12, 13], the learned window size in our method is notably smaller. In terms of accuracy, the proposed model outperforms content-based attention on TIMIT and has comparable results to joint CTC-attention on WSJ.

During our experiments, we found a key aspect in predicting the window size is to prevent it from shrinking to zero. We also found that even given a rough segmentation, it is still hard for the content-based attention to generate the best alignment. We observed that the silence parts in an utterance harm the performance of the model significantly. To cope with this issue, we padded the silence part in the beginning of each utterance with some padding characters which led to noticeable performance improvement of the model.

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

2. ATTENTION MECHANISMS

The attention based encoder-decoder model computes the probability $P(\boldsymbol{Y}|\boldsymbol{X})$ of an output sequence $\boldsymbol{Y} = \boldsymbol{y}_1, \cdots, \boldsymbol{y}_n$ conditioned on an input sequence $\boldsymbol{X} = \boldsymbol{x}_1, \cdots, \boldsymbol{x}_t$. The encoder reads the input sequence $\mathbf{X}$ and use an RNN to map it to a sequence of encoder RNN hidden states $\mathbf{H} = (\boldsymbol{h}_1, \cdots, \boldsymbol{h}_s)$.

At each decoding time step i, the decoder RNN uses the previous output $\boldsymbol{y}_{i-1}$, the previous decoder RNN hidden state $\boldsymbol{q}_{i-1}$ and the previous context vector $\boldsymbol{c}_{i-1}$ to generate the current decoder hidden state $\boldsymbol{q}_i$. The attention mechanism uses the current decoder hidden state $\boldsymbol{q}_i$ and the sequence of encoder hidden states $\boldsymbol{H}$ to compute an alignment vector $\boldsymbol{\alpha}_i$. Given the alignment vector $\boldsymbol{\alpha}_i$ as weights, a context vector $\boldsymbol{c}_i$ is computed as a weighted sum of the source hidden states. Finally, the decoder uses the context vector $\boldsymbol{c}_i$ and the current decoder hidden state $\boldsymbol{q}_i$ to estimate the current label distribution $P(\boldsymbol{y}_i|\boldsymbol{X}, \boldsymbol{Y}_{1:i-1})$. The decoder can be described as follows:

$$\boldsymbol{q}_i = \text{RNN}(\boldsymbol{q}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_{i-1}) \quad (1)$$

$$\alpha_{ij} = \text{Attention}(\boldsymbol{q}_i, \boldsymbol{h}_j) \quad (2)$$

$$\boldsymbol{c}_i = \sum_j \alpha_{ij} \boldsymbol{h}_j \quad (3)$$

$$\boldsymbol{y}_i \sim \text{LabelDistribution}(\boldsymbol{c}_i, \boldsymbol{q}_i) \quad (4)$$

The attention mechanism estimates the alignment between the current output $\boldsymbol{y}_i$ and all the input units. There are several approaches to compute the attention vector. The following equations describe two approaches:

$$e_{ij} = \boldsymbol{v}^T \tanh(\boldsymbol{W}\boldsymbol{q}_i + \boldsymbol{V}\boldsymbol{h}_j + \boldsymbol{b}) \quad \text{(Content)} \quad (5)$$

$$e_{ij} = \boldsymbol{v}^T \tanh(\boldsymbol{W}\boldsymbol{q}_i + \boldsymbol{V}\boldsymbol{h}_j + \boldsymbol{U}\boldsymbol{f}_{ij} + \boldsymbol{b}) \quad \text{(Location)} \quad (6)$$

$$\boldsymbol{f}_i = F * \boldsymbol{\alpha}_{i-1} \quad (7)$$

where $\boldsymbol{v}, \boldsymbol{W}, \boldsymbol{V}, \boldsymbol{U}, F, \boldsymbol{b}$ are trainable parameters. The final attention vector is:

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{t} \exp(e_{ik})} \quad (8)$$

2.1. Rule-based windowed attention

A weakness of the attention mechanism for speech recognition is that it is too flexible and does not guarantee the generation of a monotonic alignment, since similar input units or noise can cause misalignment. Furthermore, the above attention mechanisms consider the entire sequence of encoder hidden states, which may not be necessary for speech recognition, since each output label (typically a phoneme or a character) corresponds to a small span of input signal.

Windowed attention mechanisms, in which the attention is constrained to consider a window of the input, are proposed to alleviate these problems. The window moves from left to right along the input time axis, helping to generate a monotonic alignment. In rule-based windowed attention methods [8, 11, 12, 13], the attention vector $\boldsymbol{\alpha}_i$ can be described as:

$$\alpha_{ij} = \begin{cases} \frac{\exp(e_{ij})}{\sum_{k=m_i-D_l}^{m_i+D_r} \exp(e_{ik})}, & j \in [m_i - D_l, m_i + D_r] \\ 0, & \text{otherwise} \end{cases} \quad (9)$$

where m_i is the centre of the window, and D_l and D_r are the size of the left half and right half of the window respectively. The method of moving the window and the window length are pre-defined. The window shift and window length are not trainable.

Such rule-based window methods have been found to require a large window size – often close to the length of the entire utterance – in order to get good performance [8, 13]. However, if the window size is comparable to the utterance length, then it does not particularly help to generate a monotonic alignment nor to reduce the computational effort.

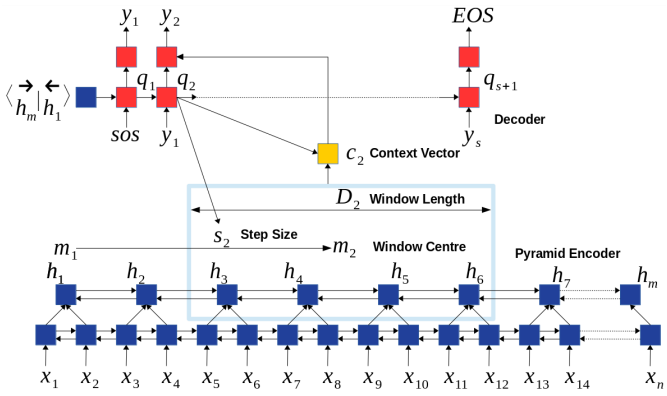


Fig. 1: The windowed attention mechanism. At decoding time step i, the system uses an MLP to estimate a step size s_i. The centre of the window is moved from m_{i-1} to m_i. Only the encoder hidden states which are within the window will be exploited to calculate the context vector. The window length can either be set as a hyperparameter or learned.

2.2. Fully-trainable windowed attention

To alleviate these problems, we propose a fully-trainable windowed attention (Figure 1). In this model, allthe window parameters are trainable and the learned window size is small. The step size s_i is predicted by an MLP,

$$s_i = N \cdot \text{sigmoid}(\text{MLP}(\boldsymbol{q}_i)), \quad (10)$$

where N is the maximum allowed step size and set as a hyperparameter. The model records the window centre at the previous decoding time step.

Within the window, we use content-based attention to compute the content score e_{ij}. We use a differentiable function to calculate the location score l_{ij}. The differentiable function also

makes the step size and window size trainable. We choose two differentiable functions, the Gaussian distribution,

$$l_{ij} = \begin{cases} \exp(-\frac{(j-m_i)^2}{2(D_{il}/2)^2}), & j \in [m_i - D_{il}, m_i] \\ \exp(-\frac{(j-m_i)^2}{2(D_{ir}/2)^2}), & j \in (m_i, m_i + D_{ir}] \end{cases} \quad (11)$$

and the concatenation of two sigmoid functions,

$$l_{ij} = \begin{cases} \sigma(k \cdot (j - m_i) + b), & j \in [m_i - D_{il}, m_i] \\ \sigma(k \cdot (m_i - j) + b), & j \in (m_i, m_i + D_{ir}] \end{cases} \quad (12)$$

where D_{il} and D_{ir} denote the length of the left and the right half windows respectively while k and b are hyperparameters. The final weight is defined as:

$$\alpha_{ij} = \frac{\exp(e_{ij}) \cdot l_{ij}}{\sum_{k=m_i-D_{il}}^{m_i+D_{ir}} \exp(e_{ik}) \cdot l_{ik}}. \quad (13)$$

The Gaussian distribution makes the final weight tend to be large near the centre of the window. Thus, it provides a strong location constraint. The sigmoid functions, when the hyperparameter b is set large, has large values within the window. Thus, they nearly only provide the indices of the most related encoder hidden states and the content-based attention almost solely determines the final weight.

When the half window sizes D_{il} and D_{ir} are equal, the location score is symmetrically distributed within the window. When they are different, the location score makes the model tend to further rely on the past or the future context. The half window sizes can either be set as hyperparameters or learned by one MLP so the window is symmetric or learned by two MLPs so the window may be asymmetric. The window size MLP is defined as:

$$D_i = D \cdot \text{sigmoid}(\text{MLP}(\boldsymbol{q}_i)) \quad (14)$$

where D is the maximum allowed half window size.

3. EXPERIMENTS

We performed two sets of experiments: phoneme recognition on TIMIT [15] and character recognition on WSJ1 [16] and subsets of WSJ1. The TIMIT dataset is split into training, development, and test sets following the Kaldi s5 recipe [17]. We use 80 mel-scale filter-bank features with energy as input features. The outputs are 39 phonemes with start/end sentence (*sos*/*eos*) and space tokens.

For the WSJ dataset, we use *train_si284* as the training set, *dev93* as the development set and *eval92* as the test set. The input features are 40 mel-scale filter-bank features with delta and delta delta. There are 33 output labels: 26 letters, and the apostrophe, period, dash, space, noise and *sos*/*eos* tokens.

All models are implemented using ESPnet [18]. For the experiments on TIMIT, the encoder is a bidirectional LSTM (BLSTM) layer on top of two pyramid BLSTM [19] layers with 256 hidden units in each direction. The decoder is a one layer LSTM with 512 hidden units. For the experiments on WSJ, the encoder is two BLSTM layers on top of two pyramid BLSTM layers with 320 hidden units in each direction. The decoder is a one layer LSTM with 320 hidden units. We use the AdaDelta [20] learning algorithm with gradient clipping. All weights are initialized uniformly within the range $[-0.1, 0.1]$. For decoding, we use a beam search with beam size 20.

Function Type	N, D_l, D_r	PER (Test)
Baseline: content-based attention		20.1%
Gaussian-Fixed window	5, 3, 3	17.0%
Gaussian-Fixed window	5, 4, 2	17.3%
Gaussian-Fixed window	5, 2, 4	17.3%
Gaussian-1 window MLP	5, 12, 12	16.8%
Gaussian-2 window MLPs	4, 6, 6	**16.7%**
sigmoid($\pm 1.5x + 3$)	5, 4, 4	17.8%
sigmoid($\pm 1.5x + 7$)	5, 4, 4	23.4%
sigmoid($\pm 1.5x + 7$)	5, 7, 7	19.1%
Hierarchical maxout CNN [21]		16.5%
Wavenet [22]		18.8%

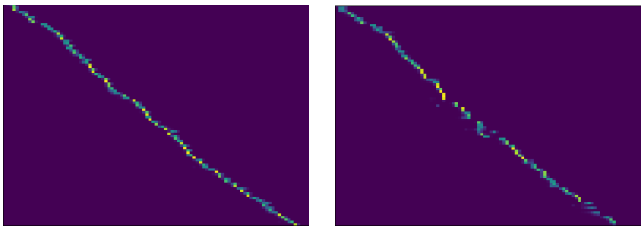
Table 1: PER on TIMIT test set. N, D_l, D_r denote the maximum allowed step size, left window size and right window size. One unit of the step/window size is 0.04s. If using fMLLR features and changing the activation function of the step size MLP and the window size MLP from tanh to LeakyRELU, the network achieves 14.9% PER on the test set.

3.1. Results and discussion on TIMIT

On the TIMIT data set, the results in Table 1 show our fully-trainable model outperforms the content-based attention baseline significantly, reducing the phoneme error rate (PER) by about 17% relative. Our model yields the best result when the window size is trained by two separate MLPs,

When the window size is learned, we set a minimum window size which covers 5 encoder hidden states (0.2s) – this is key to successfully learning the step size. Without using a minimum window size, the window sizes for some output units shrink to zero quickly. Other methods, such as pre-training the model with a fixed size window, making the window size MLP share one bottom layer with the step size MLP, or setting both the maximum step size and window size to a large value also alleviate this problem. However, these methods do not generate the best results. Furthermore, if using a fixed size window length of 0.2s, with the same maximum allowed step size, the model fails completely and is unable to learn a meaningful alignment vector, resulting in a very high PER.

As shown in Figure 2, the concatenation of two sigmoid functions can predict the proper position of the window. However, this approach is worse than the baseline when the sigmoid functions are flat within the window. Thus, if only provided with the small window, and without a strong location con-



(a) Gaussian (b) Sigmoid

Fig. 2: Attention vectors generated by different constraint functions. The utterance is fgjd0_sx279 in the TIMIT development set (*/sil ae l s ih z ah sil b ih l ah sil t iy sil t ah w er sil k w ih th aw sil s uw sil p er v ih sh ih n ih z n ow sil w er dh iy sil/*).

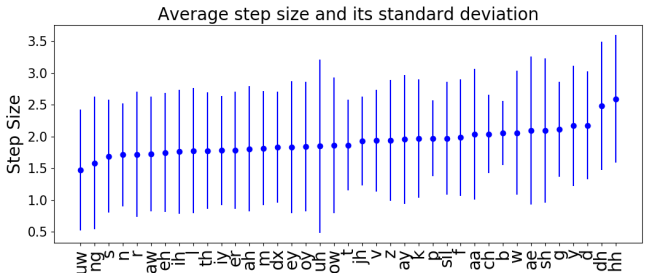


Fig. 3: The average learned step size for each phoneme and its standard deviation. The data is collected on TIMIT development set and test set.

straint, content-based attention alone cannot generate the best alignment.

We summarize the average step size and its standard deviation in Figure 3. Based on the statistics and our experiments using sigmoid window functions, we hypothesize two reasons for why a large window size is required when using a rule-based step size [8, 13]. First, the standard deviation is large. Thus, if the step size is fixed or rule-based, it will be often that the step size is smaller or bigger than the best step size. If the window size is small, then the most related source hidden states may not be included in the window. Second, even given the proper step size, our experiments indicate that without a strong location constraint within the window the attention mechanism is unable to generate the best alignment when the window size is small.

3.2. Results and discussion on WSJ

On the WSJ dataset our model again outperforms the baseline but does not surpass the CTC-attention model (Table 2). We believe a major reason for this is that long pauses often occur in WSJ utterances. Thus, the maximum allowed step size and window size have to be relatively large so the model can jump over the silence part. However, large maximum window and step sizes are not optimal for non-silence parts.

For WSJ, we set both the maximum window size and step size to 1.32s, which is small compared to the window size (8.0s) in the rule-based method with a fixed step size [13]. Furthermore, except for the silence part, the learned step size and window size are much smaller than the maximum size. When combined with CTC, our model yields almost identical results to CTC-attention, but the computational requirements of the CTC-Gaussian attention approach are noticeably lower.

Since the utterances in WSJ begin with silence of varying lengths, and the silence part makes our model difficult to train, we pad the beginning of the text for each utterance with a padding word which is made of eight padding characters. This approach improves the performance of the content-based attention significantly. The lower part of Table 2 gives results for reduced subsets of the *train_si284* training data, using 15k and 30k utterances.

Model(train)	CER(dev)	CER(eval)
train_si284	dev93	eval92
Baseline: content-based attention	11.1%	8.9%
location-based attention	9.6%	6.9%
Gaussian-2 window MLPs	9.0%	6.5%
CTC-attention	**7.7%**	5.9%
CTC-Gaussian	7.8%	**5.8%**
Previous results [9] (no padding):		
content-based attention	13.7%	11.1%
location-based attention	12.0%	8.2%
CTC-attention	11.3%	7.4%
train_si284 subset (30K)	dev93	eval92
Baseline: content-based attention	9.9%	7.9%
Gaussian-2 window MLPs	9.5%	7.2%
CTC-attention	**9.1%**	**6.9%**
train_si284 subset (15K)	dev93	eval92
Baseline: content-based attention	15.7%	13.7%
Gaussian-2 window MLPs	13.2%	9.6%
CTC-attention	**10.8%**	**8.3%**

Table 2: Character error rate (CER) on WSJ. Settings of the encoder-decoder and the optimizer are similar to [9]. After removing the padding characters in the decoding text generated by the baseline, the CER on eval92 is 7.5%, similar to CTC-attention without silence padding.

4. CONCLUSION

We have proposed a fully-trainable windowed attention mechanism. Compared to windowed attention approaches using a rule-based step size, our method learns a very small window and has competitive results. The proposed model outperforms the content-based attention on TIMIT and WSJ and has comparable results to CTC-attention on WSJ. We also find padding the silence part in the beginning of the utterances improves the performance of content-based models significantly. Our future work includes investigating our model on noisy data.

5. REFERENCES

[1] Kelvin Xu, Jimmy Ba, Ryan Kiros, Kyunghyun Cho, Aaron Courville, Ruslan Salakhudinov, Rich Zemel, and Yoshua Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015, pp. 2048–2057.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[3] Minh-Thang Luong, Hieu Pham, and Christopher D Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[4] Rohit Prabhavalkar, Kanishka Rao, Tara N Sainath, Bo Li, Leif Johnson, and Navdeep Jaitly, "A comparison of sequence-to-sequence models for speech recognition," in *Interspeech*, 2017, pp. 939–943.

[5] Jan K Chorowski, Dzmitry Bahdanau, Dmitriy Serdyuk, Kyunghyun Cho, and Yoshua Bengio, "Attention-based models for speech recognition," in *NIPS*, 2015, pp. 577–585.

[6] Albert Zeyer, Kazuki Irie, Ralf Schlüter, and Hermann Ney, "Improved training of end-to-end attention models for speech recognition," *arXiv preprint arXiv:1805.03294*, 2018.

[7] Jan Chorowski and Navdeep Jaitly, "Towards better decoding and language model integration in sequence to sequence models," in *Interspeech*, 2017, pp. 523–527.

[8] Rohit Prabhavalkar, Tara N Sainath, Bo Li, Kanishka Rao, and Navdeep Jaitly, "An analysis of attention in sequence-to-sequence models," in *Interspeech*, 2017.

[9] Suyoun Kim, Takaaki Hori, and Shinji Watanabe, "Joint CTC-attention based end-to-end speech recognition using multi-task learning," in *IEEE ICASSP*. IEEE, 2017, pp. 4835–4839.

[10] Takaaki Hori, Shinji Watanabe, Yu Zhang, and William 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.

[11] Navdeep Jaitly, Quoc V Le, Oriol Vinyals, Ilya Sutskever, David Sussillo, and Samy Bengio, "An online sequence-to-sequence model using partial conditioning," in *NIPS*, 2016, pp. 5067–5075.

[12] Tara N Sainath, Chung-Cheng Chiu, Rohit Prabhavalkar, Anjuli Kannan, Yonghui Wu, Patrick Nguyen, and Zhifeng Chen, "Improving the performance of online neural transducer models," *arXiv preprint arXiv:1712.01807*, 2017.

[13] Dzmitry Bahdanau, Jan Chorowski, Dmitriy Serdyuk, Philemon Brakel, and Yoshua Bengio, "End-to-end attention-based large vocabulary speech recognition," in *IEEE ICASSP*, 2016, pp. 4945–4949.

[14] Andros Tjandra, Sakriani Sakti, and Satoshi Nakamura, "Local monotonic attention mechanism for end-to-end speech and language processing," *arXiv preprint arXiv:1705.08091*, 2017.

[15] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David 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.

[16] Linguistic Data Consortium, "CSR-II (wsj1) complete," *Data Consortium, Philadelphia*, vol. LDC94S13A, 1994.

[17] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *IEEE ASRU*, 2011.

[18] Shinji Watanabe, Takaaki Hori, Shigeki Karita, Tomoki Hayashi, Jiro Nishitoba, Yuya Unno, Nelson Enrique Yalta Soplin, Jahn Heymann, Matthew Wiesner, Nanxin Chen, et al., "ESPnet: End-to-end speech processing toolkit," *arXiv preprint arXiv:1804.00015*, 2018.

[19] William Chan, Navdeep Jaitly, Quoc V Le, and Oriol Vinyals, "Listen, attend and spell," *arXiv preprint arXiv:1508.01211*, 2015.

[20] Matthew D Zeiler, "Adadelta: an adaptive learning rate method," *arXiv preprint arXiv:1212.5701*, 2012.

[21] László Tóth, "Phone recognition with hierarchical convolutional deep maxout networks," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, pp. 25, 2015.

[22] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," *arXiv preprint arXiv:1609.03499*, 2016.