WaveNet for Automatic Speech Recognition

Yonk Shi
pyshi@kth.se

Clément Morin
clemor@kth.se

Thomas Peterson
thpeter@kth.se

Abstract

In this work we adapt WaveNet, an innovative deep neural network for generating speech, to automatic speech recognition tasks. After establishing an appropriate architecture, we first train the network to model the posterior probability of every possible letter at a given time step of the input sequence. Thereafter, the network is trained to model phoneme posterior probabilities after having translated the text transcriptions from the original dataset into phonetical transcriptions. All learning was made using data from the LibriSpeech corpus. The acquired results suggest that WaveNet might be a plausible candidate for ASR. Additionally, they indicate that WaveNet's training behavior when using phonemes is similar to the behavior obtained with characters. Finally, they show that the conducted phoneme augmentation did not have any major positive effects.

1 Introduction

Automatic speech recognition (ASR) has seen many changes in the course of its history, going from template-matching-related techniques [1] to end-to-end training methods using recurrent neural networks [2]. Traditionally, mixture models have been used to compute emission probabilities for training with a Hidden Markov model (HMM) [3].

Neural networks (NNs) have been successfully used in the ASR field, mainly as acoustic models when combined with a HMM [4]. The use of deeper models such as deep feedforward neural networks (DNNs) has yielded significant improvements for the acoustic modeling part of ASR tasks [5]. A new kind of ASR systems entirely based on NNs has recently emerged and breaks free from HMMs: end-to-end NN-based models, in particular end-to-end recurrent neural networks (RNNs) [2].

The goal of the work was to work with the adaption to the ASR field of an end-to-end generative model called WaveNet [6], which was originally designed for speech generation and has proven to be very successful for this task [7][8]. The authors shortly described the possibility to perform the adaptation of the network [6] and announced promising results on the TIMIT dataset. This created motivation to explore this path, along with WaveNet's ability to embed a good internal representation of speech.

To achieve our goals, we used Connectionist Temporal Classification (CTC) [9] as a training method that allows to work directly with text transcriptions without requiring any alignment between input sequences and target sequences. CTC has demonstrated state-of-the-art performance when used in conjunction with RNNs [10][11]. In addition, we replaced the original raw waveform input used by WaveNet by mel-frequency cepstral coefficients (MFCCs) [12]. MFCCs allow us to decrease the dimensionality of the input while ensuring that we use an efficient representation of the speech.

2 Method

The implementation was conducted by reimplementing the original architecture but feeding MFCCs as input and providing character or phoneme probabilities as the output. The phoneme probabilities

could then be used to generate a predicted text via lookup in a phoneme-to-word dictionary. For an overview of the implementation the reader is encouraged to consult figure 1 which provides an overview of both the character-level and phoneme-level implementations.

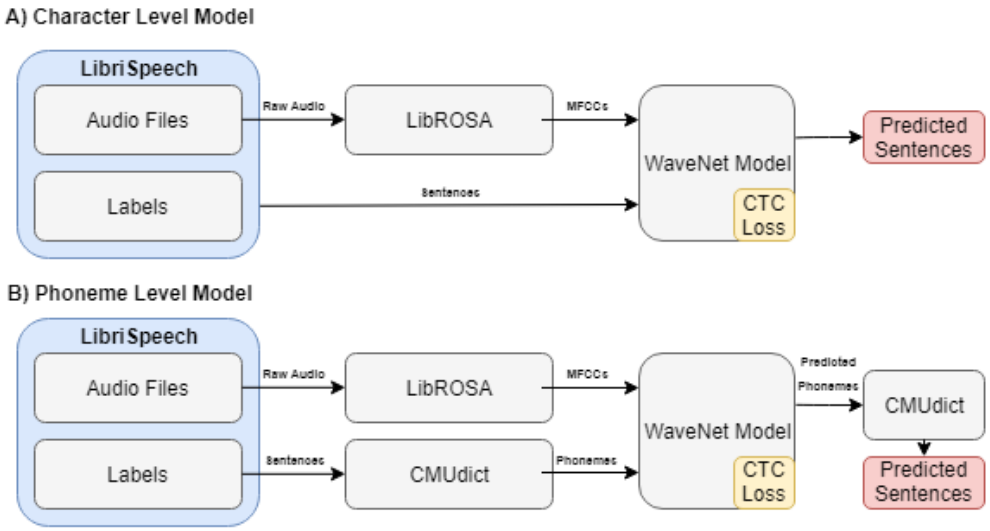


Figure 1: An overview of the two implementations considered in this work.

The LibriSpeech dataset was used for training [13]. This dataset contained raw audio files as well as character-level labels. During the preprocessing, all of the audio files where encoded as vector sequences of 20 MFCCs. While the initial idea behind the paper we took inspiration from [14], was to make inference from raw audio, the authors of this implementation decided to use MFCCs as input because of hardware limitations. Because of time constraints and as MFCCs have proven their efficiency in the ASR field, we decided to explore this approach too. To only keep 20 coefficients rather than more or less of them was motivated by earlier work suggesting that only a couple of the first coefficients are enough for ASR and that varying the number of coefficients does not result in a strong performance impact [15]. Additionally, all labels were converted from character level to phoneme level using CMUdict [16].

2.1 MFCCs – Extracting Relevant Features

When sound is created by a human it is filtered by the shape of the vocal tract, the tongue and the front of the mouth. By determining the shape of these it would in theory be possible to acquire a good idea about which phonemes are being uttered. The shape of the vocal tract is represented by the envelope of the short time power spectrum of the sound [17]. To extract this property, what is referred to as MFCCs has been developed. The procedure for extracting MFCCs is summarized in figure 2.

Figure 2: The process of extracting MFCCs from raw audio.

The first step of the MFCC extraction procedure is to conduct pre-emphasis. Afterwards, the signal is enframed into short frames. This is because the signal is considered quasi-stationary during short periods of time [17]. Thereafter, a power spectrum is calculated for each frame by taking the square of the absolute value of the Fourier transform. This power spectrum mimics the behavior of the human ear in the sense that it identifies which frequencies are present in the sound. In the human body this is conducted by the cochlea which contains hair cells that vibrate at different frequencies which in turn fires neural signals which travel to the primary auditory cortex located in the temporal lobe [18].

The next step is to apply a bank of filters that are evenly spaced on the mel scale to extract the energies in different frequency regions [17]. The mel scale is a modification of the frequency scale adapted to the fact that humans do not hear on a linear scale but instead closer to a logarithmic one [18]. Thereafter, the discrete cosine transform (DCT) of the logarithm of the filterbank energies is taken. The DCT is used to decorrelate the energies from each other [17]. Finally, only a specified number of the first coefficients are kept (depending on how coarsely or finely the MFCCs should represent the signal) and the remaining are discarded. This is motivated by the property of DCT that the first coefficients model slow changes in the filterbank energies whereas the last model fast changes and that these fast changes have been shown to not facilitate ASR in any particular way [17].

2.2 CMUdict – Word-to-phoneme Conversion

CMUdict is a free pronouncing dictionary for North American English maintained by the Speech Group in the School of Computer Science at Carnegie Mellon University [19]. The dictionary contains over 134,000 words as well as their corresponding pronunciation. The pronunciation consists of phonemes from the ARPAbet phoneme set and thus contains 39 phonemes where vowels can have three different stress levels. These stress levels are marked with 0 for no stress, 1 for primary stress and 2 for secondary stress. Since the data structure for the dictionary is easily handled, the dictionary can not only be used to convert words to phonemes but also to convert phonemes back to words. However, the latter requires more computational resources than the former.

To allow the network to train on the phoneme level, all the labels of the LibriSpeech dataset were translated to phonemes using CMUdict. Phonemes were thought to be favorable to use instead of characters since one phoneme always has the same pronunciation whereas a character can be pronounced differently depending on the context. For example, the character "a" in "family" and "face" are pronounced differently but the a sound is mapped by different phonemes. This transformation was believed to facilitate the learning for the network in the sense that it would remove ambiguities since each output token would only map to one sound. However, this proved to introduce other issues, such as phonemes mapping to the same word since two words could be pronounced in the same way. For example, the words "Eye" and "I" had the same pronunciation. To solve this issue, our conclusion was that a lexical model would be needed. However, that was out of the scope for this project and was thus left as a challenge to address in future work.

2.3 WaveNet – The Generative Model for Raw Audio

WaveNet is a generative model based on dilated and causal convolutions [6]. It was primarily designed for TTS applications but has proved to perform well for music generation and speech recognition as well. Its architecture is built out of residual blocks. An illustration of a residual block can be observed in figure 3. These blocks consist of a dilated causal convolution layer whose output is fed into a gated activation block, another convolution whose product is sent back to the input of the dilated convolution via a 1x1 convolution. The input is firstly fed into a convolutional layer. Thereafter, the output goes through a hyperbolic tangent function and a sigmoid function. Then, the output of these are multiplied element-wise. Thereafter the product is sent back to the input of the convolutional layer. This process is repeated K times and is what is referred to as a residual block.

The convolutions in the convolutional layer are *causal* and *dilated*. The causal aspect of these convolutions comes from the fact that the successive stacked layers are not allowed to model dependencies on future timesteps, while the dilated aspect is due to their ability to expand the receptive field(The coverage of the input elements mapping to one output element) of the network by skipping timesteps of the input. The network is thus capable of modelling long-term dependencies in a cheaper way.

Special activation units called *gated activation units* were also used, as they had been observed to perform better at modeling audio signals than the more traditional rectified linear unit (ReLU) [20]. Finally, the depth of the network is leveraged by the use of residual connections and parametrized skip connections. These residual connections connect layers that are not successively connected while skip connections connect the output of the 1x1 convolution to the output pipeline. Residual and skip connections allow deeper architectures to be trained by facilitating the propagation of gradient values during backpropagation, avoiding vanishing gradients [6]. Additionally they speed up the convergence of the network [6].

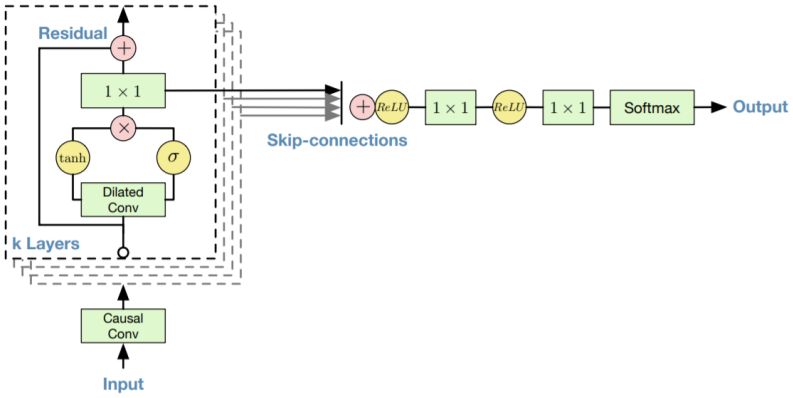


Figure 3: A k-layered residual block from the WaveNet architecture [6].

The WaveNet paper did not provide much details regarding how to adapt the network to ASR. To fill in the blanks, we initially took inspiration from an independent Python implementation based on TensorFlow [14]. Something that differed in our implementation was that we didn't apply global and local conditioning as was done in a more specialized version of the network in the original paper.

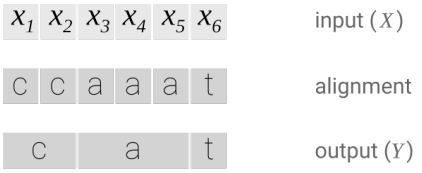
2.4 Connectionist Temporal Classification

CTC [9] is a technique which allows *alignment-free* training of ASR systems. It is indeed traditionally required to know, for each timestep in the input sequence (which could be a raw waveform or a sequence of MFCC vectors), what the corresponding character, phoneme or phone state in the target output sequence is [3][4][5]. This is what is referred to as alignment. For each such timestep and its aligned target character, phoneme or phone state, the *inner model* is trained so that it assigns the highest possible probability to that target character given the input features at that timestep (or vice versa using Bayes' rule). This inner model could for instance be a NN or a Gaussian mixture model (GMM) and is generally used in combination with an HMM [3][4][5]. As ASR datasets usually give the transcription (made of characters or phonemes) of each audio clip without any alignment data [13][21][22], a preexisting ASR system must be used to produce this alignment information provided that its architecture makes such computation possible. For example, in order to develop a HMM-DNN system, a HMM-GMM model can be used to recover the most probable sequence of hidden phone states that have generated each input sequence.

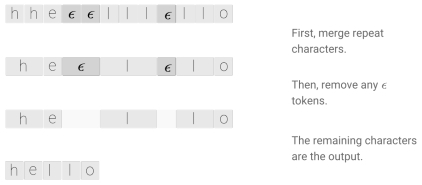
CTC makes it possible to break free from this traditional training pipeline. To do so, it requires that the ASR system being developed respects two constraints:

1. The model that will be trained must be defined so that it outputs, for each input step and each possible output token (one of the target symbols or the blank token which will be presented below), the probability that this output token corresponds to the input features at that timestep.
2. The length of the target sequences must be shorter than the length of the input sequences.

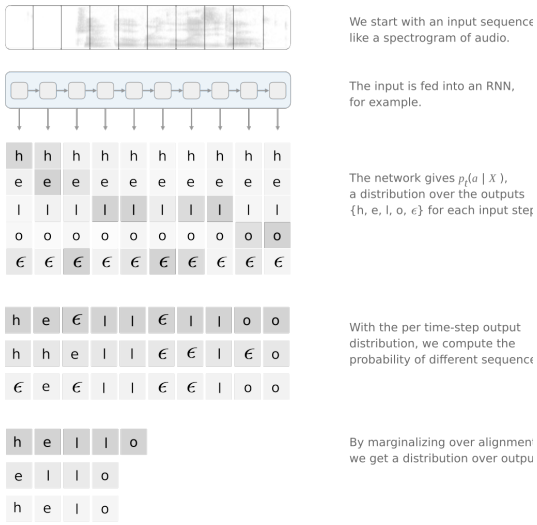
The first requirement can be met by various models, for instance by a RNN. It is met in our case by the convolutional neural network (CNN) made of causal convolutions (as described in section 2.3) which is featured in WaveNet. WaveNet's network takes the whole sequence as input and applies a softmax function at the last layer so that a vector of probabilities can be obtained at each timestep. These applications of the softmax function are made on a vector whose dimensions match the size of the token vocabulary, e.g. 26 alphabet letters plus some punctuation signs plus the blank token. The purpose of this architecture is to enable the computation of the probability of any possible output sequence (made of output tokens except the blank token) by considering all possible alignments that lead to this output sequence and using the per-timestep probabilities given by the network. The alignments use the blank token to represent output sequences that contain the same token multiple times in a row such as "l" in "hello". This whole process is summarized in figure 4. The second requirement is supposed to be always met in the context of ASR as one phoneme



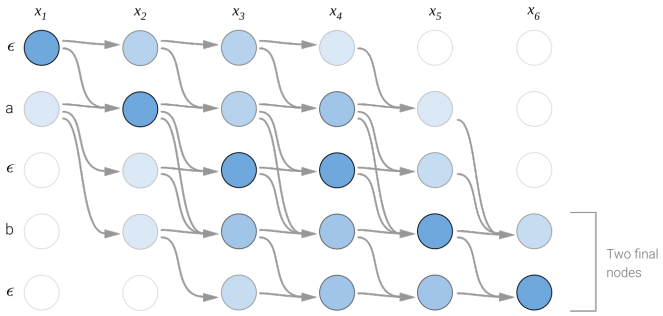
(a) Illustration of the concept of alignment.



(b) Alignments with blank tokens.



(c) Process of computing the probability of any output sequence.



(d) Illustration of the computation of the loss function for one target sequence (here, "ab") by graph traversal. Each node represents a per-timestep probability.

Figure 4: Main concepts featured in CTC. Figures obtained from [23].

generally corresponds to many more than one input timestep so that transcriptions are shorter than input sequences, even with punctuation and spaces taken into account.

These two requirements and the process described above (as well as in figure 4) give all the elements necessary to the construction of both the loss function defined by CTC and the CTC decoder which is used to perform inference. The CTC loss is made of the negative log-likelihood of the training output sequences given the input sequences. Each probability is computed by marginalizing over all possible alignments of the output sequence to the input sequence and, for each such alignment, computing the probability assigned by the network using a graph traversal dynamic algorithm as in figure 4d. This is summarized in the following formula:

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{a \in A} p(a \mid X)$$

where Y is some target output sequence, X is the corresponding input sequence, $\mathcal{A}_{X,Y}$ is the set of all possible alignments of Y to X and a is some token from one alignment. For its part, the inference method is made of a modified version of the beam search algorithm: to know what is the most probable output sequence given the output of the network for a given input sequence, the graph is progressively traversed by retaining the n most probable sequences encountered so far. The variations to the original beam search algorithm are due to the collapsing of the repetitions of identical tokens

in the n tentative sequences, and to the necessity to properly handle the blank tokens that represent paths where a token appears multiple times in a row in the output sequence.

3 Experiments

The original WaveNet paper had many crucial details missing. There was no specification on the number of layers, loss function or dataset. Because of the lack of detail, we relied on community effort by searching for similar implementations online. We found three credible implementations [14][24][25], each implementation having a slightly different interpretation of WaveNet. Our final model consisted of nine three-layered residual blocks in serie and the implementation can be found on github[26].

The WaveNet model was implemented as a hybrid of Keras and TensorFlow. Due to a documented bug inside TensorFlow [27], we relied on Keras for the causal convolution modules, and the output of the model was then fed through TensorFlow's `ctc_loss` function [28] and backpropagated. The chosen optimizer was TensorFlow's implementation of the Adam optimization algorithm (`AdamOptimizer`), as it is an adaptive optimizer and relatively stable. Additionally, it had been observed that numerous community implementations used the `AdamOptimizer` and reported good results [29].

The dataset used was the LibriSpeech development set (LibriSpeech dev) [13]. LibriSpeech dev is a specialized ASR library with 360 hours of labelled speech. The data from the acquired dataset was converted to 20-feature MFCCs in order to reduce the training time. MFCC conversion was handled by LibROSA [30].

In the experiment we examined two main different types of data while using WaveNet as an underlying model. The first model was a character-level model where MFCC inputs were converted straight into characters which formed sentences. In the second approach we adapted WaveNet to predict phonemes rather than characters. The phonemes were then converted into words and sentences using language models. However, due to time constraints, we used a basic language approach based on Levenshtein distance. Thus implementing the accuracy calculation to be independent of the underlying phoneme or character architecture.

The accuracy calculations were based on Levenshtein distance lev and were calculated according to equation 1 where N is the batch size, x is the WaveNet input, $f_{wavenet}$ is the WaveNet output function and y is the label.

$$accuracy = \frac{1}{N} \sum_{N} lev(f_{wavenet}(x), y) \tag{1}$$

In order to train the phoneme-level WaveNet, we used the pronounciation dictionary called CMUdict through the python library nltk [19]. This allowed us to convert English words into phonemes. However, some data samples had words with no entry in the dictionary. These samples were discarded, reducing the available training set from 28,539 down to 17,880 examples. Since CMUdict was only used for the phoneme-level implementation, the character-level implementation still used all 28,539 examples for training.

4 Results

This chapter provides the results of our experiments. Initially, the results of the character-level network are presented. Thereafter, the results of the phoneme-level network are detailed. Finally, a section provides the results of the augmented phoneme-level network.

4.1 Characters as Labels

In this experiment, we converted our labels into 27-dimensional alphabet labels, and trained the network with direct character output. The learning rate was set to 0.002 (standard for Adam optimizer). In figure 5, we see a rapid accuracy increase initially which then begins to plateau around 40%. It has been reported that Wavenet requires a large amount of data to train and long training time [6]. We thus believe that with a larger dataset and more computational power our network could continue to

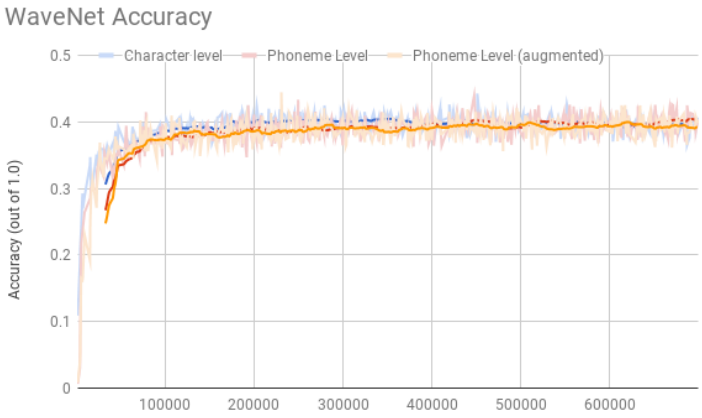


Figure 5: WaveNet accuracy for the character, phoneme and augmented phoneme model after 700,000 iterations.

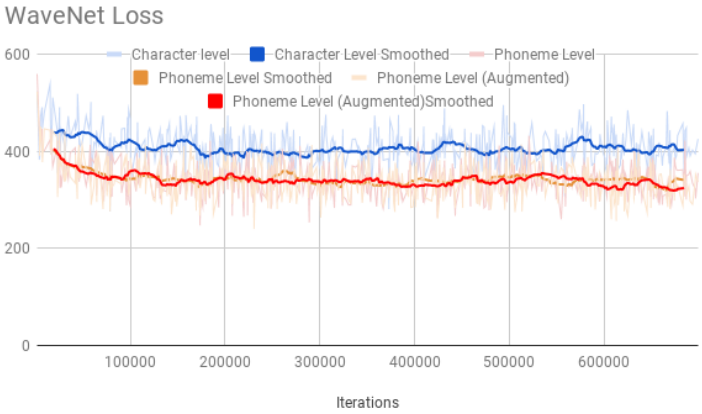


Figure 6: WaveNet loss for the character, phoneme and augmented phoneme model.

improve since we can still see that the loss is dropping and accuracy is increasing, albeit very slowly. Our experiments ran for 500,000 iterations with slowly improving predictions but they were still far from ideal.

labels : so christie turned a deaf ear [...]
predicted: t e e e e e e e e e

At the 30,000th update step, the network has learned to match the length of the label

labels : she stood for a moment on the [...]
predicted: esan peo en tono rein en sean te [...]

After 100,000 update steps, the network has learnt to print words that are phonetically similar to the correct word but has yet to learn many caveats of the English language.

labels : mode put the flour into [...]
predicted: mol a fotflor in oo asan [...]

labels : mister verloc was fully responsive now
predicted: mis o olot wos folonrs ols a

4.2 Phonemes as Labels

In this experiment, we switched from character level model to phoneme model. There were a total of 70 phonemes in CMUDict, thus our output dimension expanded from 27 in the character-level model to 70 in the phoneme-level model. Phoneme-level training added an extra layer of abstraction, but we were hoping that the system would learn better with phonemes by removing the ambiguities due to the necessity for the network to learn English spelling. However, as shown in figure 5, the accuracy is comparable to that character-level model. It follows a very similar trend as character-level training. The accuracy sees an initial quick accuracy jump and begins to learn very slowly after 40% mark. The loss is distinctively lower than character-level model, but this does not mean better results. This is because CTC loss is not normalized like many other loss functions, therefore longer sequences tend to have higher loss.

Phoneme-level training has added complexity as phonemes need to be converted back to words. In our experiments, we used the phoneme edit distance to pair phonemes to its closest resembling word, we found that the conversion is slow and difficult to translate when phonemes have greater than two comparing to dictionary. However, it is possible with more advanced language models to produce better predictions.

4.3 Augmented Phonemes as labels

In this experiment, we proposed a simple data augmentation technique in order to differentiate words that have similar pronunciations in hope to achieve better results than vanilla phoneme level prediction. E.g. **One** and **Won**. During the preprocessing stage, instead of sampling the first phoneme representation, we picked the most unique phoneme representation (The phoneme association with the least amount of associated words).

The rational behind this technique was that we would be able to teach WaveNet to differentiate between words with otherwise similar pronounciation. E.g. **bass** and **base** can both be pronounced as "B EY S" but **bass** can also be pronounced as "B AE S", therefore we favor "B AE S" whenever we encounter the word **bass**. The downside of this technique is that the association from sound samples(or MFCC) with phonemes will be weaker.

In the resulting accuracy of our experiments, which can be observed in figure 5, it can be seen that the augmented phonemes have similar accuracy growth pattern as the previous two methods, but that the accuracy is slightly lower than that of the unaugmented phoneme model counter part. Thus it can be concluded that the trade off of having unique phonemes is not worth having mismatching phonemes to audio samples.

5 Discussion and Conclusions

In this experiment, we attempted to reproduced the WaveNet [6] architecture. Due to the lack of transparency from DeepMind, it involved guessing and further research to 'fill in the gaps'. However, we were able to fill the missing pieces of the puzzle by adding the CTC loss function as well as training on both character-level predictions and phoneme-level predictions. Our results suggests that WaveNet might be a plausible solution for ASR. However, comparing our results with the results from a separate implementation [14], we did not manage to reach their level of precision. We believe that this might be because we did not let the network train with as much computational resources (time and hardware) as their network, that there is some significant difference between the VCTK and LibriSpeech datasets or that our architecture was different from theirs in some aspect. The results also suggest that the conducted phoneme augmentation did not have any major positive effects.

For future work we would suggest training with other datasets as well as training for longer periods of time and with various different hyperparameters. We would also suggest tweaking the architecture by, for example, applying other activation functions as well as residual block layouts. Additionally, it could be interesting to train the network on raw audio rather than MFCCs. Finally, we would suggest adding a lexical model to the final step of the prediction procedure as we believe that this could greatly increase the accuracy of the network.

References

[1] L Rabiner and S Levinson. Isolated and connected word recognition–theory and selected applications. *IEEE Transactions on Communications*, 29(5):621–659, 1981.

[2] Alex Graves, Abdel-rahman Mohamed, and Geoffrey Hinton. Speech recognition with deep recurrent neural networks. In *Acoustics, speech and signal processing (icassp), 2013 ieee international conference on*, pages 6645–6649. IEEE, 2013.

[3] Hermann Ney, Reinhold Haeb-Umbach, B-H Tran, and Martin Oerder. Improvements in beam search for 10000-word continuous speech recognition. In *Acoustics, Speech, and Signal Processing, 1992. ICASSP-92., 1992 IEEE International Conference on*, volume 1, pages 9–12. IEEE, 1992.

[4] Hynek Hermansky, Daniel PW Ellis, and Sangita Sharma. Tandem connectionist feature extraction for conventional hmm systems. In *Acoustics, Speech, and Signal Processing, 2000. ICASSP'00. Proceedings. 2000 IEEE International Conference on*, volume 3, pages 1635–1638. IEEE, 2000.

[5] Geoffrey Hinton, Li Deng, Dong Yu, George E Dahl, Abdel-rahman Mohamed, Navdeep Jaitly, Andrew Senior, Vincent Vanhoucke, Patrick Nguyen, Tara N Sainath, et al. Deep neural networks for acoustic modeling in speech recognition: The shared views of four research groups. *IEEE Signal Processing Magazine*, 29(6):82–97, 2012.

[6] 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.

[7] DeepMind Technologies Limited. Wavenet: A generative model for raw audio. `https://deepmind.com/blog/wavenet-generative-model-raw-audio/`, 2016. Accessed: 2018-06-08.

[8] DeepMind Technologies Limited. Wavenet launches in the google assistant. `https://deepmind.com/blog/wavenet-launches-google-assistant/`, 2017. Accessed: 2018-06-08.

[9] Alex Graves, Santiago Fernández, Faustino Gomez, and Jürgen Schmidhuber. Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In *Proceedings of the 23rd international conference on Machine learning*, pages 369–376. ACM, 2006.

[10] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International Conference on Machine Learning*, pages 1764–1772, 2014.

[11] Haşim Sak, Andrew Senior, and Françoise Beaufays. Long short-term memory based recurrent neural network architectures for large vocabulary speech recognition. *arXiv preprint arXiv:1402.1128*, 2014.

[12] S Davis and P Mermelstein. Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences. *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 28(4):357–366, 1980.

[13] LibriSpeech. `http://www.openslr.org/12/`, 2015. Accessed: 2018-04-30.

[14] Speech-to-text-wavenet : End-to-end sentence level english speech recognition using deepmind's wavenet. `https://github.com/buriburisuri/speech-to-text-wavenet`, 2016. Accessed: 2018-04-30.

[15] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. Comparative evaluation of various mfcc implementations on the speaker verification task. In *Proceedings of the SPECOM*, volume 1, pages 191–194, 2005.

[16] Kevin Lenzo. The cmu pronouncing dictionary. `http://www.speech.cs.cmu.edu/cgi-bin/cmudict`, 2018. Accessed: 2018-06-10.

[17] Xuedong Huang, Alex Acero, and Hsiao-Wuen Hon. *Spoken Language Processing: A Guide to Theory, Algorithm, and System Development*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 1st edition, 2001.

[18] D. Purves. *Neuroscience*. Oxford University Press, 2012.

[19] Source code for nltk.corpus.reader.cmudict. `http://www.nltk.org/_modules/nltk/corpus/reader/cmudict.html`, 2018. Accessed: 2018-04-30.

[20] Vinod Nair and Geoffrey E. Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA, 2010. Omnipress.

[21] CSTR VCTK Corpus. `http://homepages.inf.ed.ac.uk/jyamagis/page3/page58/page58.html`, 2010. Accessed: 2018-06-09.

[22] TED-LIUM. `http://www.openslr.org/7/`, 2012. Accessed: 2018-06-09.

[23] Awni Hannun. Sequence modeling with ctc. *Distill*, 2017. https://distill.pub/2017/ctc.

[24] A TensorFlow implementation of DeepMind's WaveNet paper. `https://github.com/ibab/tensorflow-wavenet/`, 2017. Accessed: 2018-04-30.

[25] WaveNet implementation in Keras. `https://github.com/basveeling/wavenet`, 2018. Accessed: 2018-04-30.

[26] Top-notch Automatic speech Recognition System (TARS). `https://github.com/yonkshi/TARS`, 2018.

[27] Support "causal" padding in tf.keras.layers.conv1d by adding support to tf.layers.convolutional.conv1d. `https://github.com/tensorflow/tensorflow/issues/14933`, 2017. Accessed: 2018-04-30.

[28] Api r1.8. `https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss`, 2018. Accessed: 2018-04-30.

[29] Strange growth in the loss. `https://github.com/ibab/tensorflow-wavenet/issues/143`, 2016. Accessed: 2018-04-30.

[30] LibROSA. `https://librosa.github.io/librosa/`, 2018. Accessed: 2018-04-30.