Speaker-Independent Phoneme Classification using Convolutional Neural Networks on Varying Context Input Window Lengths

Oscar Alsing, Felix Engström, Albin Söderholm
{oalsing, felixen, albinso}@kth.se

KTH Royal Institute of Technology
Faculty of Computer Science
Stockholm, Sweden

Abstract

The use of Deep Neural Networks (DNN) is widely spread in Speech Recognition, as they have been proven to outperform the traditional Hidden Markov Models (HMM) and Gaussian Mixture Model (GMM) recognition architecture. Whereas the traditional model uses HMMs for temporal variability of speech, and GMMs determines the probability of each HMM state belonging to an acoustic input, the DNN takes multiple frames of data and outputs posterior probabilities over HMM states. Furthermore, Convolutional Neural Networks (CNN) have been proven to outperform the DNN. Sharing the benefits of DNNs, CNNs achieve better generalization and more robust models.

This paper aims to investigate the performance of Convolutional Neural Networks using Context Input Windows of varying size, and our experiments clearly indicates improved classification accuracy as the length of context input window is increased, with diminishing returns as the length increases.

1 Introduction

The aim of Automatic Speech Recognition (ASR) is to transcript human speech into spoken words. For many years state-of-the-art ASR systems modelled the relationship between speech signals and phonemes using spectral-based feature extraction followed by training using GMM-HMM. Recently, Deep Neural Networks (DNN) have been proven to outperform GMM [1, 2, 3, 4, 5, 6] for estimating probabilistic distributions associated with HMM states. The superior performance of the DNN classifier is due to the powerful discriminative model, capable of representing arbitrary classification surfaces in contrast to the *one* hidden expert assumption by the GMM. Furthermore, the DNNs have proven superior to HMM in ASR, as the former is not restricted by the classical independence assumptions [1]

Previous research on the subject have proven that the CNN is able to outperform a DNN [7, 8, 9, 10], and attributes its robustness to learned features from small frequency shifts [11]. Furthermore, deep CNN architectures have proven useful in end-to-end speech recognition with no data preprocessing, as they explicitly harness locality in the spectral feature space through shared weight filter[12]. The use of CNNs in Speech Recognition is driven by the successes of Deep CNNs in Computer Vision, and investigates performance by adding depth of processing using more non-linarites and expressive power, while keeping the number of parameters manageable [1].

CNNs have in earlier research been stated to capture translational invariance in signals with fewer parameters than DNNs, by replicating weights across frequency and time [10], and are inherently more well suited for speech recognition.

This paper aims to to further explore the use of the Convolutional Neural Network (CNN) [13] by evaluating the phoneme classification performance of the network using on Mel Frequency Spectral Coefficients (MFCCs) [14] on the TIMIT [15] dataset, and the impact of various degrees of context frame size.

1.1 Tensorflow

Tensorflow [16] is an open source library for machine learning developed by Google to compute numerical operations efficiently. Tensorflow has the ability to run on multiple CPUs and GPUs and has support for general-purpose computing these utilizing CUDA [17], which we utilize to run our computations faster.

1.2 Keras

Keras [18] is a high-level open source neural network API written in Python, able to run on top of Tensorflow. It focuses on offering developers a minimal, modular and extensible interface enabling fast experimentation through layers of abstraction in deploying neural networks.

1.3 TIMIT dataset

The TIMIT dataset [15] provides speech data for acoustic-phonetic studies, and is one of the most commonly used datasets for evaluation and development of ASR systems. The dataset contains recording from 630 speakers of various dialects, and hand verified corpus transcriptions for the sentences read by the speakers and includes time-aligned transcriptions for each utterance. The dataset was produced in a collaboration between Massachusetts Institute of Technology (MIT), SRI International (SRI) and Texas Instruments, Inc. (TI).

2 Method

2.1 Data preprocessing

We have preprocessed the TIMIT dataset for MFCC feature extraction. The speech for each utterance is analyzed in a 20ms hamming Windows with 10ms frame rate, and the feature vectors are computed by an Fast Fourier Transformation, mel filterbank log spectrum and with cosine transform and liftering - as well as the first and second temporal derivatives. Furthermore, all data is normalized to have $\mu = 0$ and $\sigma = 1$.

Using the first 13 coefficients, this gives us an input vector of 39 features for each frame with the first and second temporal derivative included. The three features is split up into three different arrays, giving us frames $f \in \mathcal{R}^{13x3}$. The frames are partitioned into windows of some odd width N, giving us new data points $\in \mathcal{R}^{13x3xN}$ with labels corresponding to the central frame. Following the transformation, the size of our data is represented as seen in figure 1

Data-set	Data points	Frames	Features	Dims
Train	1359661	N	13	3
Validation	147731	N	13	3
Test	1527014	N	13	3

Figure 1: Table over data dimensions

Where the corresponding labels data are represented as $(len(data) \cdot 64)$, where 64 is the number of unique phonemes to be classified from the TIMIT dataset.

2.2 Convolutional Neural Networks

A CNN differs from a basic DNN in that each layer consists of a 3-d tensor of neurons that is convolved over the input data rather than a matrix multiplication with a 2-dimensional tensor of neurons, as seen in figure 2.

CNNs were originally used in image recognition with the width and height of the input volume corresponding to the width and height of the image. The depth of the volume would represent the different color channels, most commonly red, green and blue.

Each neuron is only connected to a local area of the input area (though the entire depth). This vastly reduces the number of weights in the network and diminishes many of the negative effects of increased data size. For ASR we use locality in the MFC Coefficients and time rather than spatial locality which is used when the input data is an image.

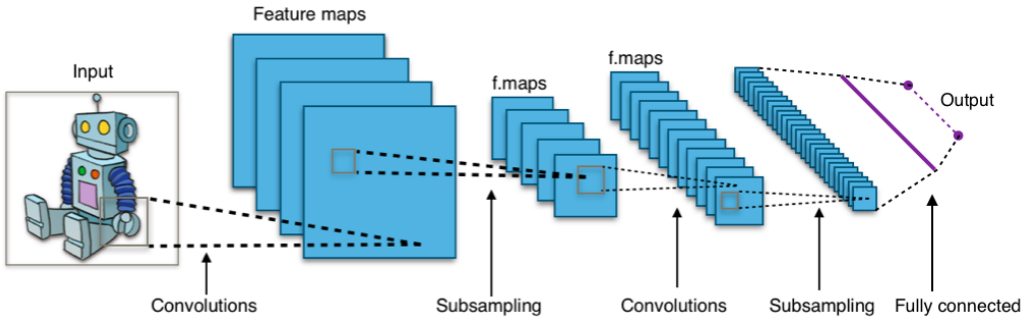


Figure 2: Typical CNN Architecture [19].

2.3 Model

We implemented our network using Keras [18] running an CUDA [17]-powered Tensorflow [16] back-end.

Our model will consist of C convolutional layers where each layer c_i consist of k filters with size m_i x n_i. Each filter will be convolved over the input from the previous layer with a step size of 1. For each convolution layer there will be an pooling layer consisting of a mp_i x np_i max pooling layer, applied with a step size one on the features from the convolution layer. Following the convolution layers is an S deep fully connected MLP with ReLU activation functions were layer s_j will have h_j hidden units. Finally the estimated distribution over the states will be calculated as the softmax function of the output of the MLP. Our loss is calculated as the cross entropy between our prediction and the true labels.

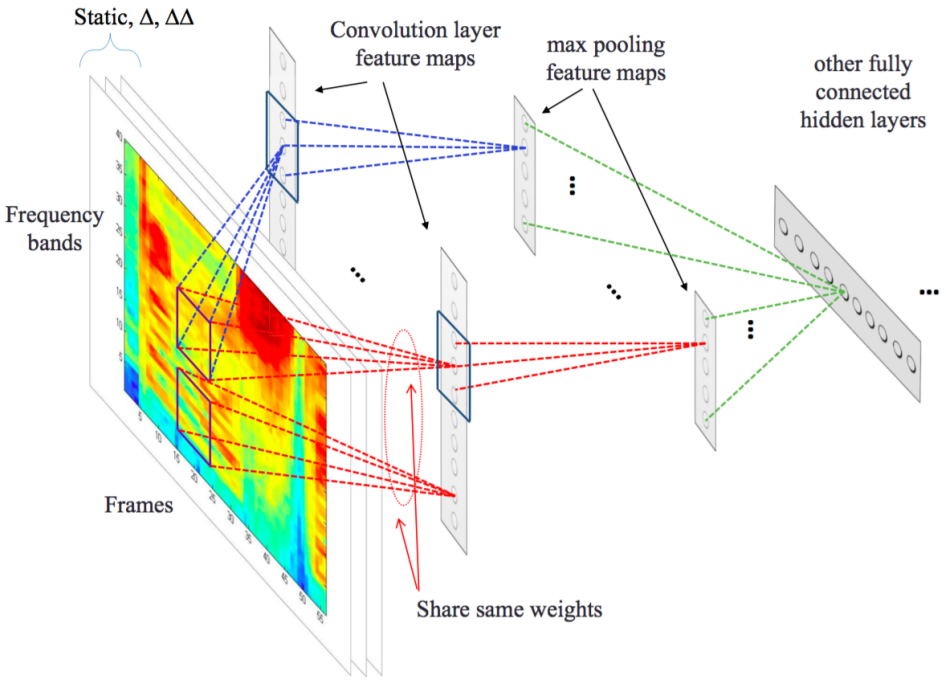


Figure 3: A Convolutional Neural Network with full weight sharing [11]

2.4 Model optimization

The optimal CNN architecture search began with an initial 1-layer CNN model. Thereafter, we continuously incremented the complexity of the model while carefully monitoring the accuracy and loss values on the validation and test set. During this process, SGD, Adagrad, RMSprop were used as optimizers interchangeably and independently evaluated. The optimal network architecture and hyper-parameter settings were found using random search [20].

3 Experiments

As mentioned in 2.4, our initial model consisted of a 1-layer CNN, and from this initial model we iterated over more complex models by continuously adding layers, in order to achieve improved error rates. Models with increased accuracy were naturally given more complexity in the search for higher performance. Our final models are presented in table 1, and the intermediary models were omitted for the sake of clarity. All network evaluations were conducted on the full TIMIT dataset.

As seen in table 2, the impact of increased the context input window length also drastically increases the size of the dataset due to the linear increase in data size. This could of course be avoided by creating the context windows on the fly. Naturally, the execution time for each epoch is increased accordingly as well. Given restricted computational power, restricting the context input window length to a upper bound might be necessary.

Table 1: Performance on the TIMIT dataset with different CNN models, along with the size of the context frame and the test error rate. 'F' is the number of feature maps, 'K' is the kernel size, 'P' is the pooling size, 'PA' is a boolean representing if padding is active or not, 'S' is the number of hidden nodes in a fully connected layer. All layers utilize the ReLU activation function, except for the last layer with a softmax activation function.

ID	Context Input Window Length	Network Structure	Test Error rate
1	1	Conv({F: 32, K: 1x3}) -> Pool({P: 2x2, PA: True}) -> Dense({S: 28}) -> Dense({S: 64})	30.33%
2	3	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Dense({S: 28}) -> Dense({S: 64})	24.75%
3	7	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Dense({S: 28}) -> Dense({S: 64})	21.53%
4	13	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Dense({S: 28}) -> Dense({S: 64})	20.1%
5	19	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Dense({S: 28}) -> Dense({S: 64})	18.08%
6	19	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Dropout 10% -> Dense({S: 28}) -> Dense({S: 64})	**17.61%**
7	1	Conv({F: 32, K: 1x3}) -> Pool({P: 2x2, PA: True}) -> Conv({F: 64, K: 1x3, PA: True}) -> Pool({P: 1x1}) -> Dropout 10% -> Dense({S: 64})	27.33%
8	3	Conv({F: 32, K: 1x3}) -> Pool({P: 2x2, PA: True}) -> Conv({F: 64, K: 3x3, PA: True}) -> Pool({P: 2x2}) -> Dropout 10% -> Dense({S: 64})	23.08%
9	7	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Conv({F: 64, K: 3x3, PA: True}) -> Pool({P: 2x2}) -> Dropout 10% -> Dense({S: 64})	20.21%
10	13	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Conv({F: 64, K: 3x3, PA: True}) -> Pool({P: 2x2}) -> Dropout 10% -> Dense({S: 64})	19%
11	19	Conv({F: 32, K: 3x3}) -> Pool({P: 2x2, PA: True}) -> Conv({F: 64, K: 3x3, PA: True}) -> Pool({P: 2x2}) -> Dropout 10% -> Dense({S: 64})	18.47%

Data-set	Context Input Window Length	Size	Epoch Run Time
Train	1	$0.45GB$	65s
Train	3	$1.4GB$	68s
Train	7	$3.2GB$	72s
Train	13	$5.8GB$	75s
Train	19	$8.5GB$	117s

Table 2: Dataset size and epoch run time

4 Results

The results from our conducted experiments as presented in table 1 clearly indicates an improvement in accuracy as the context input window length is increased.

In table 3 we see the average reduction of error rate as the context window is increased. Each entry is the improvement with respect to the last entry, meaning that when increasing the context window length from 13 to 19 we see a 1.28% improvement in error rate. It is notable that the improvement is quite drastic at a small number of context frames and then quickly diminishes.

The performance of model 5 increased by adding a dropout layer, indicating that the model suffered from overfitting. As seen in figure 4 and figure 5, representing the accuracy and loss graphs of the two models with dropout layers, this is no longer the case and the performance of the two models increase steadily.

Context Input Window Length	Average Test Error Rate Improvement
1	-
3	4.91%
7	3.05%
13	1.32%
19	1.28%

Table 3: Previous TIMIT Phoneme Error Rates (PER).

Paper	TIMIT test error rate
Ossama Abdel-Hamid et al. [8]	20.07%
Ossama Abdel-Hamid, Li Deng, and Dong Yu. [11]	20.05%
Ossama Abdel-Hamid et al. [10]	19.92%
Oscar Alsing, Felix Engström, and Albin Söderholm (this work)	17.61%
Lázló Tóth [21]	16.50%

Figure 4: Model 6 performance

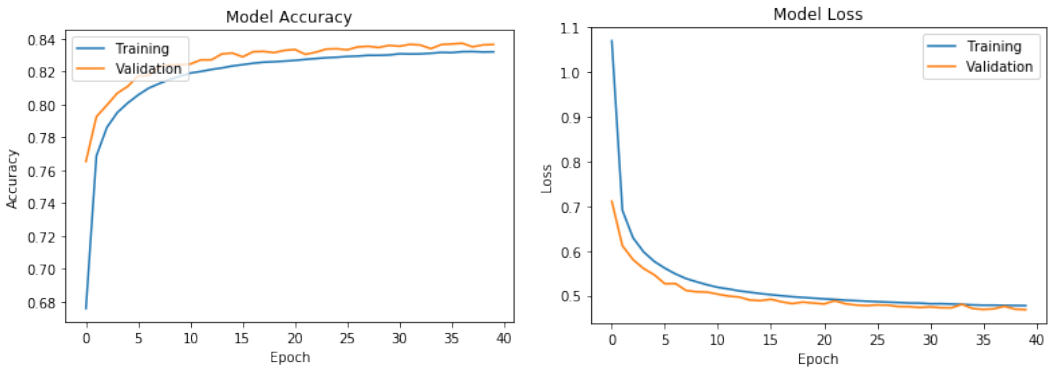
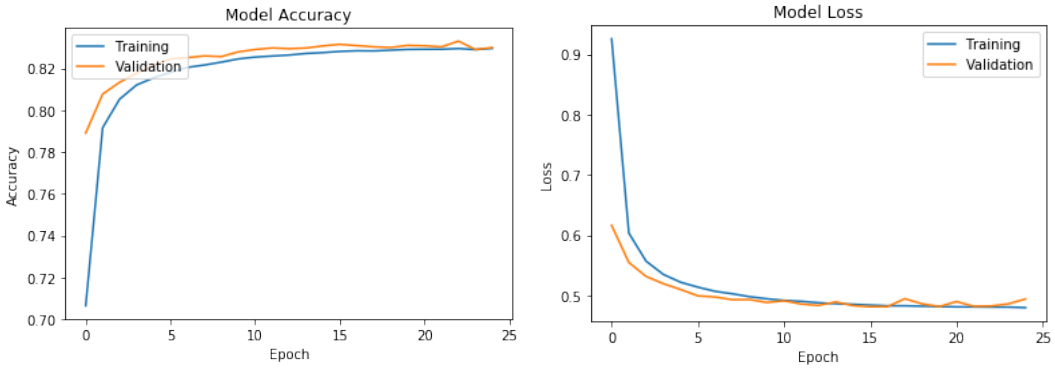


Figure 5: Model 11 performance



5 Discussion and conclusions

In this paper, we have evaluated the performance of stacked CNN layers on the task of classifying phonemes in the TIMIT dataset using a wide spectrum of architectures on input windows with varying context lengths. Experimenting with these aspects of our implementation, our results show strong indications of higher performance with increased depth in the CNN network.

Furthermore, it is clear that the length of the context input window impacts the network performance, and lower degrees of context frame size lowers the predictive power of the network.

From our experiments, we can conclude that a Deep Convolutional Neural Network performs well on the phoneme classification task with reasonable architecture depth and context input window length on the TIMIT dataset given 13 MFCC coefficients with the first and second temporal derivative for each frame. Furthermore, our results strongly suggests context input window length as an important aspect for the performance of the CNN model.

References

[1] *Recent Advances in Deep Learning for Speech Research at Microsoft*. IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), May 2013. URL: https://www.microsoft.com/en-us/research/publication/recent-advances-in-deep-learning-for-speech-research-at-microsoft/.

[2] G. E. Dahl et al. "Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition". In: *Trans. Audio, Speech and Lang. Proc.* 20.1 (Jan. 2012), pp. 30–42. ISSN: 1558-7916. DOI: 10.1109/TASL.2011.2134090. URL: http://dx.doi.org/10.1109/TASL.2011.2134090.

[3] Geoffrey Hinton et al. "Deep Neural Networks for Acoustic Modeling in Speech Recognition". In: *IEEE Signal Processing Magazine* 29 (Nov. 2012), pp. 82–97. URL: https://www.microsoft.com/en-us/research/publication/deep-neural-networks-for-acoustic-modeling-in-speech-recognition/.

[4] Chao Weng et al. "Deep Neural Networks for Single-channel Multi-talker Speech Recognition". In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 23.10 (Oct. 2015), pp. 1670–1679. ISSN: 2329-9290. DOI: 10.1109/TASLP.2015.2444659. URL: https://doi.org/10.1109/TASLP.2015.2444659.

[5] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton. "Improving deep neural networks for LVCSR using rectified linear units and dropout." In: *ICASSP*. IEEE, 2013, pp. 8609–8613. URL: http://dblp.uni-trier.de/db/conf/icassp/icassp2013.html#DahlSH13.

[6] Yajie Miao and Florian Metze. "Improving low-resource CD-DNN-HMM using dropout and multilingual DNN training." In: *INTERSPEECH*. Ed. by Frédéric Bimbot et al. ISCA, 2013, pp. 2237–2241. URL: http://dblp.uni-trier.de/db/conf/interspeech/interspeech2013.html#MiaoM13.

[7] Danish bukhari, Yutian Wang, and Hui Wang. "Multilingual Convolutional, Long Short-Term Memory, Deep Neural Networks for Low Resource Speech Recognition". In: *Procedia Computer Science* 107 (2017). Advances in Information and Communication Technology: Proceedings of 7th International Congress of Information and Communication Technology (ICICT2017), pp. 842–847. ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2017.03.179. URL: http://www.sciencedirect.com/science/article/pii/S1877050917304544.

[8] Ossama Abdel-Hamid et al. "Applying Convolutional Neural Networks concepts to hybrid NN-HMM model for speech recognition." In: *ICASSP*. IEEE, 2012, pp. 4277–4280. ISBN: 978-1-4673-0046-9. URL: http://dblp.uni-trier.de/db/conf/icassp/icassp2012.html#Abdel-HamidMJP12.

[9] Tara N. Sainath et al. "Deep convolutional neural networks for LVCSR." In: *ICASSP*. IEEE, 2013, pp. 8614–8618. URL: http://dblp.uni-trier.de/db/conf/icassp/icassp2013.html#SainathMKR13.

[10] Ossama Abdel-Hamid et al. "Convolutional Neural Networks for Speech Recognition". In: *IEEE/ACM Trans. Audio, Speech and Lang. Proc.* 22.10 (Oct. 2014), pp. 1533–1545. ISSN: 2329-9290. DOI: 10.1109/TASLP.2014.2339736. URL: http://dx.doi.org/10.1109/TASLP.2014.2339736.

[11] Ossama Abdel-Hamid, Li Deng, and Dong Yu. "Exploring Convolutional Neural Network Structures and Optimization Techniques for Speech Recognition". In: *Interspeech 2013*. ISCA, Aug. 2013. URL: https://www.microsoft.com/en-us/research/publication/exploring-convolutional-neural-network-structures-and-optimization-techniques-for-speech-recognition/.

[12] Yu Zhang, William Chan, and Navdeep Jaitly. "Very deep convolutional networks for end-to-end speech recognition". In: *arXiv preprint arXiv:1610.03022* (2016).

[13] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems 25*. Ed. by F. Pereira et al. Curran Associates, Inc., 2012, pp. 1097–1105. URL: http://papers.nips.cc/paper/4824-imagenet-classification-with-deep-convolutional-neural-networks.pdf.

[14] Todor Ganchev, Nikos Fakotakis, and George Kokkinakis. "Comparative evaluation of various MFCC implementations on the speaker verification task". In: *in Proc. of the SPECOM-2005*. 2005, pp. 191–194.

[15] J. S. Garofolo et al. *DARPA TIMIT Acoustic Phonetic Continuous Speech Corpus CDROM*. 1993.

[16] Google. *Tensorflow*. 2017. URL: https://www.tensorflow.org/ (visited on 05/11/2017).

[17] Nvidia. *CUDA*. 2017. URL: http://www.nvidia.com/object/cuda_home_new.html (visited on 05/11/2017).

[18] François Chollet. *Keras*. 2017. URL: https://keras.io/ (visited on 05/11/2017).

[19] Aphex34. *Typical CNN architecture*. 2015. URL: https://upload.wikimedia.org/wikipedia/commons/6/63/Typical_cnn.png (visited on 05/13/2017).

[20] James Bergstra and Yoshua Bengio. "Random Search for Hyper-Parameter Optimization". In: *Journal of Machine Learning Research* 13.13 (Feb. 2012), pp. 1735–1780. URL: http://www.jmlr.org/papers/volume13/bergstra12a/bergstra12a.pdf.

[21] László Tóth. "Phone recognition with hierarchical convolutional deep maxout networks". In: *EURASIP Journal on Audio, Speech, and Music Processing* 2015.1 (2015), p. 25. ISSN: 1687-4722. DOI: 10.1186/s13636-015-0068-3. URL: http://dx.doi.org/10.1186/s13636-015-0068-3.