Convolutional Neural Networks for Speech Recognition

Helder Martins
850919-R113
helder@kth.se

Xinyi Lin
930724-5481
xinyil@kth.se

Abstract

Deep neural networks (DNNs) have recently been proven to improve significantly the performance of speech recognizers when comparing to the classical Gaussian Mixture Model. This is mainly attributed to its ability to model complex correlations from the input without assuming that the data belongs to a specific type of distribution. Convolutional neural networks (CNN) share the benefits of the DNNs, also adding the ability to effectively model the representational invariance of the input which has been recently shown to improve the quality of predictions when compared to a standard DNN. For this task we modeled a Convolutional Neural Network reference architecture and trained it with utterances from the TIMT dataset composed of more than 6.300 samples. We further experimented and assessed some different model variations as to understand how the classifier accuracy behaves. Finally, the paper proposes some suggestions based on the current work to improve the performance of the model.

1 Introduction

Automatic speech recognition (ASR), which is the transcription of unbounded human speech into the corresponding spoken words, is a very challenging task. This is mainly due to the high amount of variations that the speech signal can have: different speakers, accents, noise, gender, all these variables can highly influence the output (and thus the accuracy) of the recognizer. Historically, the temporal behavior modeled by Hidden Markov Models (HMMs) allied with the powerful probabilistic distribution estimation performed by Gaussian Mixture Models (GMMs) have been the *de-facto* standard for solving this problem. Recently however, a new trend has arisen with the successful use of Deep Neural Networks (DNNs) as a predictor of observation-state sequences instead of the classical GMM method. Although the theoretical approach is not entirely new, only with the advent of more powerful hardware was it possible to obtain significant results good enough to spark the interest of the research community.

Convolutional Neural Network is a variant of DNNs where the idea of using fully connected layers is dropped in favor of sharing a smaller set of weights over the whole input by using a convolution layer. This address one of the main difficulties about training DNNs, which is how the huge amount of weights required for learning for any reasonably sized input may end up leading to overfitting the parameters to the training data. By enforcing a local connectivity pattern between neurons of adjacent layers, CNNs have the ability to create a different representation of the input first based on local correlated features which increases in size as more layers are added to the network. Another feature of CNNs is that by using a shared set of weights the same feature is bound to be detected in each part of the input, disregarding where the feature is located. This representation invariance is crucial for modelling the variations that speech signal can have, and our expectation is that by using a model that can leverage this property a greater accuracy can be obtained when compared to a standard DNN model.

The goal of this project is to experiment on using different CNN models as to evaluate their precision regarding the phoneme classification task. Since CNNs have been used successfully used for image recognition tasks[5], we aim to use similar improvements that worked consistently in that domain and evaluate how they perform when applied to the speech recognition domain. We first describe in Section 2 our reference model which is going to serve as a basis for comparison to variant architectures. After that, details about how the experiment is being performed are carefully described in Section 3. Finally, Section 4 contains the results of the experiments and the discussion which follows.

2 Method

2.1 Architecture

To serve as a basis of comparison for our experiments, a reference network was created heavily inspired by [5], scaled down due to time and resources constraints. The model, which can be seen in Figure 1, will be used as a reference point to assess the performance (regarding the accuracy of phoneme classification) of models that are subtle variations of it.

Our so called *Baseline* model is composed of 2 convolutional layers, which is responsible for learning specific local features of the data. This is accomplished by applying a set number of *filters* (also called *kernels*) to a small part of the input at each time through a *convolution* operation. The kernels is where the learning will happen, as they will specialize over time to extract different features at some spatial position of the input. As we keep stacking convolutional layers one after the other, the reception field, that is the area of the input covered, will increase in size as features which are correlated are grouped together. This means that that convolutions first allow us to learn a good representation over a small part of the input, becoming increasingly more global as more layers are added.

After computing the convolutions for each frame sequence, a *Rectifier Linear Unit*(ReLU) activation function is applied as to make our network non-linear. It has been show by [4] that his specific function yields better results in DNNs when compared to the more classical sigmoid or hyperbolic tangent, so it is used throughout our whole architecture.

Another important component of a CNN is the *pooling* layer. In this operation, a function is applied to a small window of the input as to return a value which is independent of the order of how the features in this window are organized. The reasoning behind this is that, once our kernels have learned a feature, the exact position of them is irrelevant and thus can be disregarded. Traditionally, *max pooling* is used where the feature with largest value is chosen to represent the entire window.

Usually a set of *fully connected* layers are positioned at the end of network. For our specific task, it will help to set apart different phonemes from each other from the new representation of the data after the convolution and pooling operations. Our reference architecture is composed of 3 dense layers of this type, where the number of units gradually decreases after each consecutive layer.

Since our end goal is to find a probability distribution of phonemes, a *softmax* layer is added after the fully connected layers of our network, which is simply a function which reduces the dimensional size of the input into a set number of dimensions in the output which values sum up to one. The number of dimensions will be the number of different phonemes that we have, and the highest probability in this vector will be the phoneme that is going to be assigned as the result.

After assigning probabilities to every frame sequence belonging to each one of the classes, a cost function needs to be used to evaluate how well our network performed. A *cross entropy* function is suitable for a classification problem like the one we are addressing, so it was used in our network implementation. To improve our network at each iteration of the training algorithm, weights should be updated as to progressively reduce the value of the loss function. Gradient descent optimization method is a standard technique for minimizing loss which consists of calculating the gradients of the loss function w.r.t. the network's parameters and updating these parameters towards the direction where the loss is reduced. The *stochastic gradient descent* is an approximation of the gradient descent method which is frequently needed in practice since calculating the true gradients over the whole dataset is too costly to be computed [2]. In it, a subset of the input (a "mini-batch") is instead used to compute the derivatives, which ends up resulting in a smoother convergence towards the local minima, in comparison to one sample stochastic gradient descent. As for regularization, a *L2-norm*

was applied to every weight parameter as to penalize large values which frequently means the model is overfit to the training data.

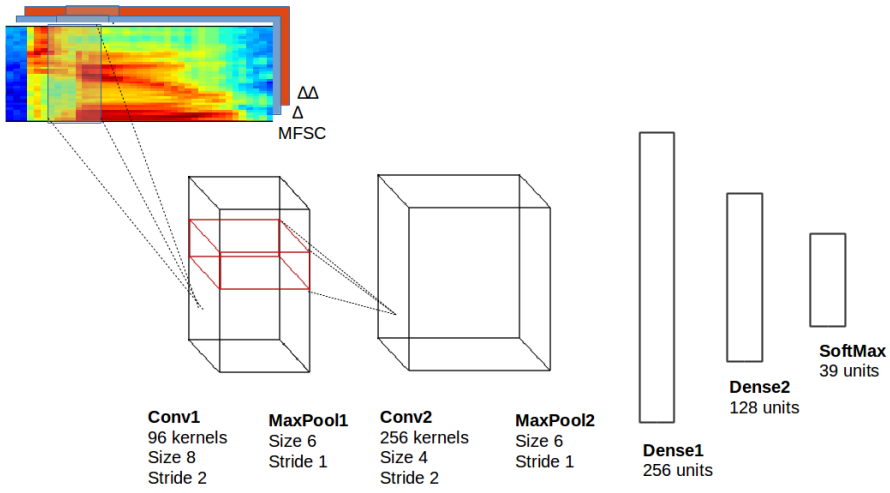

Figure 1: Baseline CNN model used in our experiments.

2.2 Setup

Using the TIMIT [3] dataset, we will transform the input sound signals into their corresponding mel-frequency spectral coefficients (MFSCs) as to preserve the locality of the features. These inputs will be mapped to their corresponding phoneme labels and are going to be used for training the CNN. The network will be modelled as a set of interchanging convolutional and pooling layers, with a set of fully connected layers near the end. The last layer, the softmax, will output the probability distribution of each input belonging to a phoneme, and that will be evaluated against our test set for accuracy check.

2.3 Data Representation

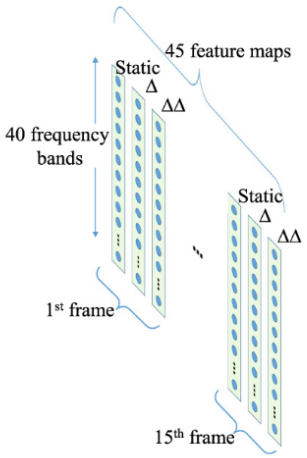

Figure 2: Example of organization one input to the CNN from [1]

Similar to the image recognition, the speech feature should be organized as the "image" to be fed in the CNN. In order to keep the frequency locality, the log-energy computed directly from the

mel-frequency spectral coefficients(without the discrete cosine transform), which denoted as MFSC features, will be the basic frame of the speech feature. Considering the three feature maps (red, green, blue) of the color images, here, the first and second temporal derivatives of the MFSC feature will also be calculated. Hence, for one frame, we have the static, delta and delta-delta three feature maps.

Like other DNNs for speech, a wider temporal contexts of acoustic frames (normally use 9 - 15 frames) will be used as one single input. Then, the input will be organized as a number of one dimensional feature maps (along with the frequency axis). For example, if we use 40 filter banks and 15 frames window as input, the MFSC features will be a vector of 40 elements and totally we have 45 feature maps(15 frames$\times$3 features).

3 Experiments

In this section, we present results for the *Baseline* model, as well as for variations of its architecture, by changing some parameters. Table 1 highlights the differences across the investigated models.

Table 1: Comparison of different models

Nmes	Meaning	1st conv. layer	2st conv. layer	LRN
Baseline	Baseline architecture	✓	✓	
NoConv3	No second conv. layer	✓		
UseLRN	Using Local Response Normalization	✓	✓	✓

Each model keeps the same architecture used in the *Baseline* model, with however a specific change. *NoConv2* model eliminates the second convolutional layer, while *UseLRN* model adds the Local Response Normalization described in Section 3.3.2 to the *Baseline* model.

3.1 Experimental Setup

In the experiments, the standard TIMIT dataset without SA records will be used to evaluate the performance of the proposed CNN model. The total training set consists of 438 speakers. In order to better select models, the TIMIT training dataset was split into 2 different sets: 75% of it was assigned to the training set which will be used to train all the different models, while 25% belongs to the validation set used to compare the accuracies in different architectural settings. Finally, a core test set from TIMIT test set, which containing 24 speakers, 2 male and 1 female from each dialect region, was used to report the result.

Table 2: Mapping from 61 original TIMIT phones to 39 classes

aa, ao	aa
ah, ax, ax-h	ah
er, axr	er
hh, hv	hh
ih, ix	ih
l, el	l
m, em	m
n, en, nx	n
ng, eng	ng
sh, zh	sh
uw, ux	uw
pcl, tcl, kcl, bcl, dcl, gcl, h#, pau, epi	sil
q	-

In the feature extracting part, records are analyzed by a 25-ms Hamming window with a fixed 10-ms frame rate, and 40 filter banks are used to generate the MFSC features with their first and second derivatives. Then, an input window consists of 9 frames and the corresponding label for this input is the same as the phone that the center frame belongs to. Instead of using the original 61 phone classes

in TIMIT dataset, we only use 39 classes for training and testing [6]. The mapping is shown in Table 2. In the left side of the table are the original labels and at the right side are the new effective classes. According to [7], the phone 'q' is discarded during mapping.

3.2 Base Result

Initially, the *Baseline* model was trained using the training set. The model was trained for 30 epochs and a batch size of 64 input samples. Results of the evaluation on the training set and validation set can be seen on the left graph of figure 3. For every epoch, one evaluation point was used. It can be noticed that the validation error rate always decreases across all the 30 epochs, which means that the *Baseline* model does not seem to overfit to the training set. The graph on the right side of Figure 3 shows the behavior of the batch accuracy and the batch loss during training. This graph shows that the loss presents a decreasing tendency during training, while the proportions of correctly classified images within batches increases.

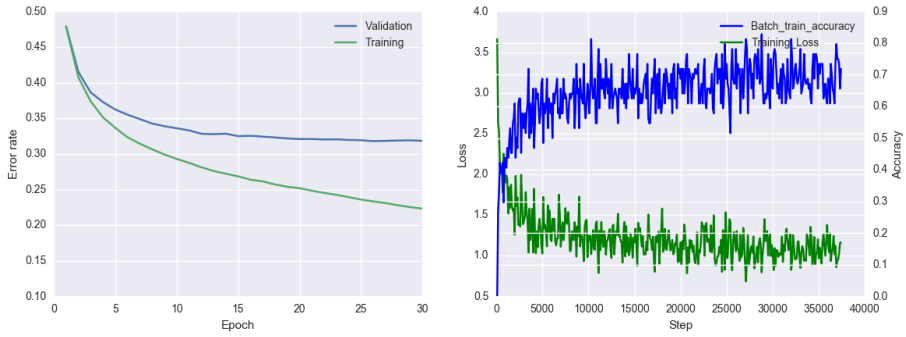

Figure 3: The performance of the baseline model

3.3 Experiments on variant models

In this section, we focus on analyzing the effect of singular architectural structures, namely the second convolutional layer and the Local Response Normalization(LRN). By evaluating the different models in the training set, we expect to extract insights about the effects of the mentioned structures on the model performance.

3.3.1 Second convolutional layer

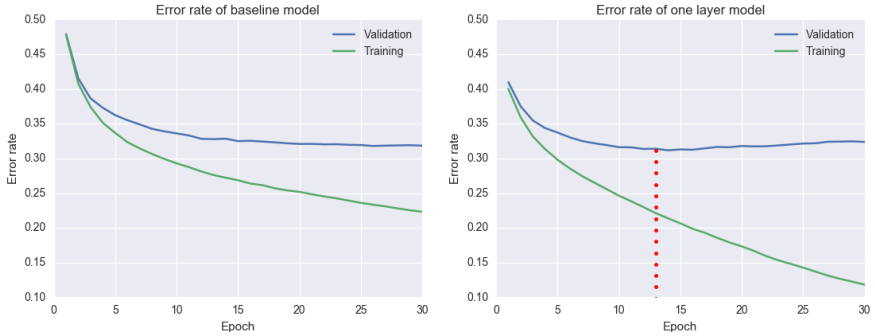

Figure 4: The comparison between baseline model and one layer model

Inspired by Krizhevsky *et al.* [5], we seek to replicate a similar architecture as much as resources constraints allow us to. However, it is still an interesting topic to discover the performance of the architecture with different number of convolutional layers. *NoConv2* model was established for this purpose, and only had one convolutional layers. In Figure 4, left part is the error rate for *Baseline* model while right part is the *NoConv2* model. The red line points out the best validation error rate

of the *NoConv2* model, which is 31.4% and it is just the same as the best error rates in the *Baseline* model, 31.8%. From this experiment, it is easy to find that removing the second convolutional layer causes the model overfitting very soon, and it does not help improve the accuracy of phone recognition.

3.3.2 Local Response Normalization

Local response normalization (LRN) is a technique used by [5] which consists in normalizing every feature of a convoluted feature map over the neighboring feature maps at the same position. This normalization of activation's response is inspired by how real neurons induce a form of local lateral inhibition between each other, creating a sort of "competition" between each other. Being $a^i_{x,y}$ is the activity of the neuron computed by applying convolution operation using the kernel i at position (x, y) of the input (followed by the ReLU activation function), the normalized feature $b^i_{x,y}$ can be formally defined as

$$b^i_{x,y} = a^i_{x,y} / \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} (a^j_{x,y})^2\right)^{\beta} \tag{1}$$

where the sum is ran over the n neighboring kernels, where N is the total number of kernels. The parameters k, n, α and β can be tuned using cross-validation, but due to resources constraints we utilized the same values used by [5], which are $k = 2, n = 5, \alpha = 10^-4, \beta = 0.75$.

Figure 5 compares the difference between the *Baseline* model and *LRN* model over 30 epochs. It can be observed the slight increase in accuracy and convergence speed that LRN provides us, even though it is suffering from overfiting after 16 epochs.

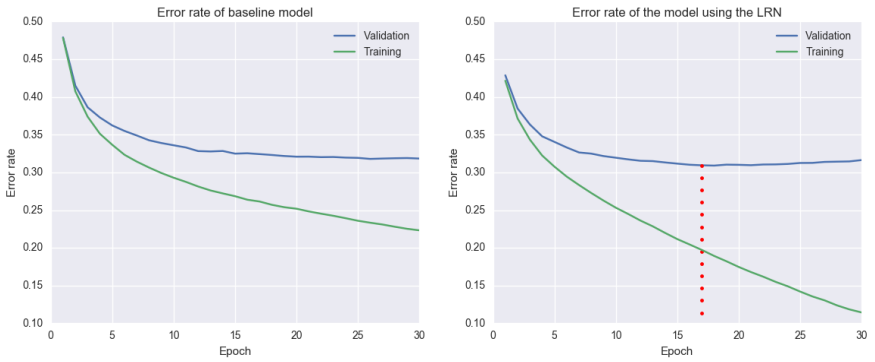

Figure 5: The comparison between baseline model and the model using the LRN

3.4 Parameters Experiment

Besides the architecture of the model, different parameters also have significant effect to the performance. In last part, we analyze the achievement of the model using different pooling size and different number of filters. By fixing all the parameters except the target one, we try to find the optimal setting.

3.4.1 Pooling size

It is hard to analyze the consequence of using different pooling size on the *Baseline* model since it has two convolutional layers. Therefore, we choose the *NoConv2* model to do the experiment. Figure 6 shows the validation error rate of using pooling size 1, 3 and 5. It is easy to find out that the model achieves lower error rate as increasing the pooling size. Meantime, it can be seen from Figure 6 that the model used smaller pooling size is easier to overfit.

Figure 6: Model performance using different pooling size

3.4.2 Filter size

Different number of feature maps are used in this experiment based on the *Baseline* model. The details of the setting can be seen in Table 3, and Figure 7 is the validation error rate for different settings. It is obvious that the model behaves better as the number of filters increases. However, the PER of baseline, $2\times$ and $4\times$ settings are almost the same, which means increasing the filter size does not improve the performance after some point. Therefore, there is no need to use a very large number of filters.

Table 3: Setting details

Name	1/4	1/2	baseline	$2\times$	$4\times$
1st conv. layer	24	48	96	192	384
2st conv. layer	64	128	256	512	1024

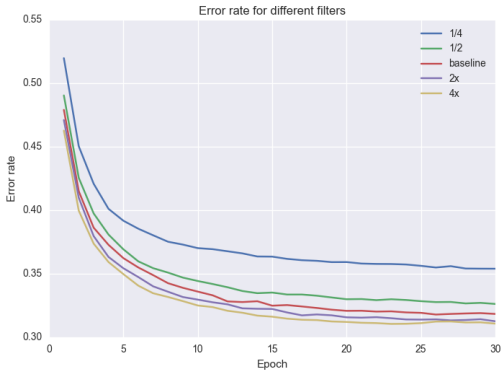

Figure 7: Model performance using different number of filters

3.5 Discussion

Our best performance is achieved by *UseLRN* model with the validation accuracy of 69%. Even though this model could attain a considerable satisfying phoneme accuracy, it suffered from an overfiting problem relatively soon, which could also be seen in some other variant models. We assume that an even better accuracy may be reached if we make use of harsher regularization techniques like increasing weight of the L2 norm or adding a dropout operation [8] after the fully-connected layers.

In the whole experiments session, different models only evaluated on the validation set. In fact, we also try to figure out the performance of different models in the core test set. But the result is not

good – the phone recognition accuracy only about 40% for different models, and the best accuracy only achieved 42%, which is also very low compared with the validation accuracy. One plausible reason is that it might exist the same speakers speaking in both training and validation sets, however, the core test set uses the totally strange speakers. The utterance contains speaker's identical character, which the proposed method might be very sensitive to. Accordingly, the model cannot obtain high accuracy on the test set.

4 Conclusions

In this report, we presented a classifier of phonemes using convolutional neural networks. We proposed a reference model to use as a basis of comparison for experiments and investigations about the effect of network's architectural structures. These different models were then compared to the reference one, and their accuracy in the classification task was measured. Similarly to image classification, our experiments have shown that using Local Response Normalization is also beneficial for speech domain, and with it we could achieve a satisfying accuracy in our validation sets.

References

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 22(10):1533–1545, 2014.

[2] Léon Bottou. Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer, 2010.

[3] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett. Darpa timit acoustic-phonetic continous speech corpus cd-rom. nist speech disc 1-1.1. *NASA STI/Recon Technical Report N*, 93, 1993.

[4] Xavier Glorot, Antoine Bordes, and Yoshua Bengio. Deep sparse rectifier neural networks. In *International Conference on Artificial Intelligence and Statistics*, pages 315–323, 2011.

[5] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, pages 1097–1105, 2012.

[6] Kai-Fu Lee and Hsiao-Wuen Hon. Speaker-independent phone recognition using hidden markov models. *Acoustics, Speech and Signal Processing, IEEE Transactions on*, 37(11):1641–1648, 1989.

[7] Carla Lopes and Fernando Perdigao. Phone recognition on the timit database. *Speech Technologies/Book*, 1:285–302, 2011.

[8] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.