Deep Convolutional and LSTM Neural Networks for Acoustic Modelling in Automatic Speech Recognition

Xiaoyu Liu
Pearson Education Inc.
4040 Campbell Avenue, Suite 200, Menlo Park, CA, 94025, USA
xiaoyu.liu@pearson.com

Abstract

In recent years, various deep models have been used for improving Automatic Speech Recognition (ASR). Among these advanced models, Convolutional Neural Networks (CNNs) and Long-Short-Term-Memory (LSTM) networks achieve state-of-the-art recognition accuracy, which generally outperforms feed-forward Deep Neural Networks (DNNs). In this work, CNN and LSTM networks with very deep structures were investigated as ASR acoustic models, and their performance was analyzed and compared with that of DNNs. The results show that CNN and LSTM significantly improves ASR accuracy for various tasks.

1. Background and related work

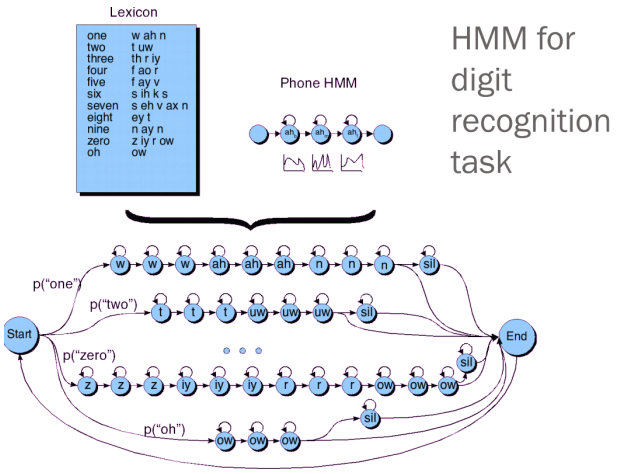

Figure 1: The structure of an ASR system, including HMM-GMM acoustic model, and a language model, which are combined to form the decoding graph. This example is cited from [1].

Figure 1 shows an ASR system that recognizes digits from continuous speech. An ASR system consists of a speech front end, an acoustic model, and a language model. The front end segments a speech waveform in time domain into overlapping short frames. Typically, each frame is 25-ms long, with 10 ms spacing (stride) between adjacent frames. The front end converts each frame to a speech feature vector which compactly encodes useful spectral-temporal information (speech feature learning will be illustrated in detail in Section 2). The entire sentence becomes a sequence of feature vectors. The objective of ASR is to obtain the mostly likely word sequence $\widehat{W}$, out of a wide range of possible word sequences W, given the feature sequence O, as:

$$\widehat{W} = argmax_W\{P(W|O)\} \quad (1)$$

Using Bayesian rule, Eq.(1) can be rewritten as:

$$\widehat{W} = argmax_W\{P(O|W)P(W)\} \quad (2)$$

in which the denominator term $P(O)$ is dropped since this term does not depend on W. The term $P(O|W)$ is defined as the acoustic model, and $P(W)$ is estimated by a language model. In modern ASR systems, the acoustic model is typically a set of Hidden Markov Models (HMMs) [2], which learns the temporal patterns in speech. At training time, words in the transcription of each sentence are decomposed into phones according to a pronunciation lexicon, and each phone is modeled by an HMM. Each HMM typically has 3 emitting hidden states, with each state corresponding to a segment of a phone. An HMM state can only transition to itself or to the next state due to the sequence property of speech. Conventionally, each state emission distribution is represented by a Gaussian Mixture Model (GMM) [3]. The phone HMMs are connected to form word HMMs and further sentence HMMs. Then, the HMM-GMM parameters are optimized to maximize the likelihood of the speech data using the EM algorithm [4]. At test time, the language model (a unigram in Figure 1) forms a word search graph, in which each path specifies a candidate sequence of words and the inter-word transition probabilities, and each word consists of the connected phone HMMs. The Viterbi decoding is performed to find the optimal word sequence over the graph.

From Eq.(1), it can be seen that speech recognition is a discriminative task. However, GMM is a generative model, in which the data likelihood is maximized during training. The use of DNN models, first proposed in [5], is a milestone for ASR. DNN replaces GMM, in that instead of

maximizing the likelihood of a feature vector o emitted by an HMM state s, i.e. $P(o|s)$, a DNN maximizes $P(s|o)$, which is the probability of a target HMM state given each observation. The DNN objective function maximizes the state level discrimination. As shown by literature [6], as the structure of a DNN gets deeper, the learned features become more discriminative over different states, and become more robust against distortions, such as noisy environment, different speakers, etc. Generally speaking, HMM-DNN acoustic model reduces the recognition error rate by 20% to 30% relative to the HMM-GMM framework.

CNN models have been adopted from computer vision tasks by ASR systems in recent years. The early CNN acoustic model, proposed in [7] uses simple one-dimensional convolution along the frequency dimension of speech spectrum, and outperforms DNNs. Later, much deeper CNN models with carefully designed network structures, such as the VGG network [8], Residual Network (ResNet) [9] greatly advance the cutting-edge ASR acoustic modelling. Nowadays, deep CNN models are widely used in state-of-the-art ASR, such as in the Microsoft 2016 evaluation [10]. In addition to recognition for clean speech, CNNs significantly enhance the noise robustness of ASR as well compared with DNN [11].

Speech is sequence data, in which the identity of the HMM state given a frame at time t is dependent on both the previous and future frames. Recurrent Neural Networks (RNNs) and their advanced LSTM version are able to exploit long historical information to predict the current class label. Deep RNN and LSTM have been powerful sequence acoustic models recently [12]. Based on LSTMs, researchers developed Bidirectional LSTM (BLSTM) [13]. Each layer in a BLSTM unfolds the time sequence both from the first to last, and from the last to first time instances. Thus, BLSTM takes advantage of both historical and future frame information to determine the current target label, which yields better performance than uni-directional LSTMs.

In this work, various deep models, including DNN, CNN, LSTM and BLSTM are extensively studied, compared and evaluated for various ASR tasks. The models implemented in this work (including the HMM-GMM model) cover almost all the model types in the CS231n course materials. In addition to evaluate recognition accuracy, feature visualization and image gradient are used to analyze the results. It is also of great interest to interpret speech as image data and gain insights. In Section 2, a detailed illustration of feature learning is provided, and in Section 3 and 4, ASR experiments are carried out. Finally, Section 5 provides a brief summary and future work.

2. Speech feature learning

This section provides a detailed description of learning speech patterns from raw features using various models. We start from illustrating raw speech features.

2.1. Raw speech features (FBANK features)

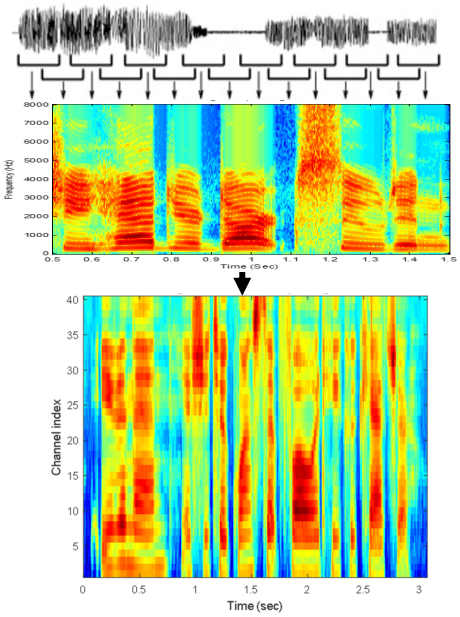

Figure 2: Raw speech features are computed by taking the weighted sum of the STFT of each frame, weighted by the 40-channel Mel filter bank. Each feature vector has 40 dimensions.

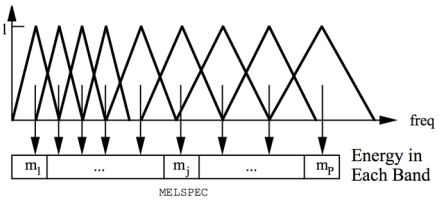

Figure 3: Frequency response of a Mel filter bank. The bandwidth becomes wider for higher frequencies (in Hertz), denoting lower resolution. This figure is cited from [14].

Figure 2 illustrates the raw feature extraction processing. The speech front end segments the time domain waveform into 25 ms short frames, with 10 ms frame spacing. For each frame (indexed by m), its magnitude-squared Short Time Fourier Transform (STFT) is taken to convert the time domain signal to its frequency domain spectrum:

$$X(m, e^{j\omega_k}) = \left| \sum_{n=0}^{N-1} s_m(n)\, e^{-jk\frac{2\pi}{N}n} \right|^2 , k = 0,1,\dots,N-1 \quad (3)$$

in which N=512 is the number of discrete frequencies in the resulting spectrum, $s_m(n)$ is the samples of the zero-padded mth frame (since each frame has 0.025 sec * 16000 samples/sec = 400 samples), $\omega_k = k\frac{2\pi}{N}$ denotes the kth discrete frequency (Figure 2 uses continuous frequency f in Hertz, $f_k = \frac{F_s}{N}k$, where F_s =16000 Hz is the sampling

frequency of the speech), and j denotes imaginary number. The STFT transforms a speech waveform to a time-frequency (TF) plane (also referred to as a spectrogram) by stacking the spectra along the time axis and using colors to represent the power value, i.e. $X(m, e^{j\omega_k})$ of each TF bin (red means large values, blue means small values).

However, human ears are not equally sensitive to all frequencies. It's easier to discern low frequencies than high frequencies. In high frequency region, a wider range of frequency components in a waveform are perceptually similar. This perceptual frequency resolution is quantified by the Mel frequency scale [15]:

$$mel(f) = 2595\, log_{10}\left(1 + \frac{f}{700}\right) \quad (4)$$

in which f denotes the linear frequency in Hertz. In ASR, a Mel filter bank (plotted in Figure 3) that typically consists of 40 channels groups the linear (along frequency) spectrum $X(m, e^{j\omega_k})$ to the Mel scale by:

$$P(m, l) = \sum_{k=0}^{\frac{N}{2}-1} X(m, e^{j\omega_k})\, H_l(e^{j\omega_k}) \quad (5)$$

where m is the frame index, l=1,2,…,40 is the channel index, N=512 is the number of STFT points (the higher half of the spectrum is not used due to symmetry), and $H_l(e^{j\omega_k})$ is the lth Mel filter frequency response, which takes a triangular shape. Note that these Mel filters are equally spaced over the Mel scale. As a result, on the linear frequency scale, the filter bandwidth becomes wider for higher frequencies, which represents lower resolution, since frequency components in the same band are perceptually identical. Finally, a logarithm is taken to compress the range of feature values, resulting in the FBANK features:

$$FBANK(m, l) = 10 log_{10} P(m, l) \quad (6)$$

2.2. MFCC features for HMM-GMM

As Figure 3 shows, there is large overlapping between Mel filters. Therefore, the FBANK features are heavily correlated among dimensions. In HMM-GMM, it's often assumed that each Gaussian mixture has a diagonal covariance matrix. Thus, the FBANK features are decorrelated by a Discrete Cosine Transform (DCT), and are reduced to 12 coefficients. The resulting features are termed as Mel Frequency Cepstral Coefficients (MFCCs) [16]. For each frame, the ith MFCC feature c_i is:

$$c_i = \sqrt{\frac{2}{N}} \sum_{j=1}^{N} FBANK_j \cos\left(\frac{\pi i}{N}(j - 0.5)\right), i = 1,2,\ldots,12 \quad (7)$$

where N=40 is the total number of FBANK features. Usually, the log energy of each frame is used as another feature. To characterize the feature trajectory over time, the time difference of the MFCCs (including the log energy feature), referred to as delta and acceleration (delta of delta) features [17] are often appended to the MFCCs, yielding 39 features (3*(12+1)) for each frame. These 39 MFCC features have been "standardized" for HMM-GMM acoustic models.

2.3. DNN and CNN based feature learning

From Section 2.1 and 2.2, it can be seen that both the FBANK and MFCC features are derived from signal processing, combined with human perception of speech. As will be shown in Section 3, these features lack discrimination and invariance over phone classes or HMM states, since there is no discriminative learning involved.

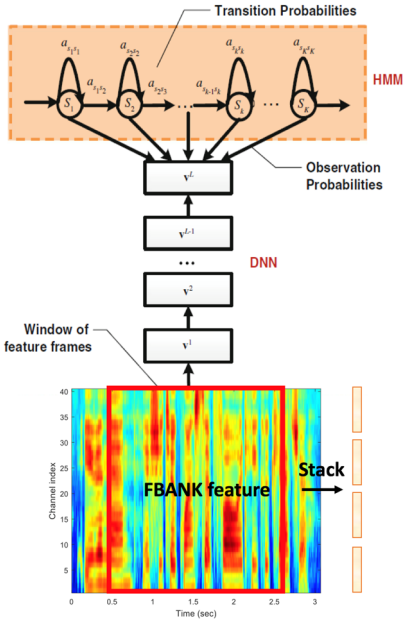

Figure 4: HMM-DNN feature learning. Input to the DNN is a stacked vector of FBANK features surrounding the center frame, and the input is mapped to the HMM state posterior probability.

Figure 4 depicts the structure of the HMM-DNN acoustic model. For each frame, the input to the HMM-DNN model is a context window of FBANK features (mean and variance normalized) surrounding the center frame. The features in this window are stacked into a long vector, then passed through the DNN, which maximizes the probability of the HMM state for the center frame of the context window. The target HMM states are obtained by first training a seed HMM-GMM model, which does not require supervised HMM state information, and search for the optimal state sequence given the correct word sequence transcription in the training set. As in the decoding stage mentioned in Section 1, each word sequence from the

training set is converted to a chain of concatenated phone HMMs, then, the optimal state sequence is obtained by the Viterbi algorithm, and used as DNN targets. At test time, the decoder needs the state emission probability $P(o|s)$, which can be computed as:

$$\hat{P}(o|s) = \frac{P(s|o)}{P(s)} \tag{8}$$

in which the term $P(o)$ is dropped since it's irrelevant to the state. $P(s|o)$ is provided by the DNN output, and $P(s)$ is estimated by a frequency count of how many times the state s is visited as a byproduct of obtaining DNN targets from the training set.

Note that when training HMM-GMM models, MFCC features are used, whereas when training the DNN, FBANK features without the DCT transform are used. It's pointed out in literature [18] that in order for the DNN to learn meaningful speech patterns, its input should have less hand-crafted components. The DCT in MFCC is a lossy data compression processing, thus is eliminated from DNN inputs. In some other literatures, such as [19], even the Mel filter bank (which is also human devised) is not used, and the STFT features are directly input to the DNN, and the filters are learned from data by inserting a "filter bank layer" in front of the hidden layers.

The general HMM-CNN structure is similar to that of HMM-DNN. The important difference is that the frames in the context window are no longer stacked into a long vector, instead, they are left as an "image" as shown in Figure 5. Thus, the FBANK TF plane surrounding the center frame becomes an image, and the state classification is analogous to an image classification task. Sometimes, the delta and acceleration features are appended to the FBANK features, in which case there are 3 feature maps, analogous to the RGB channels in a color image.

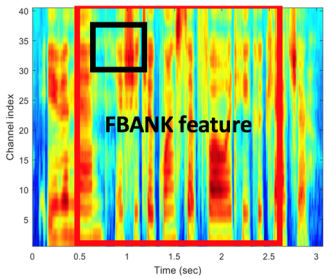

Figure 5: CNN input features. The frames in the context window form an image. Each neuron (small block) learns local pattern.

The image perspective of speech data brings benefits for ASR. Due to speaker and environment variations, the speech patterns are often distorted, and present strong locality. CNN employs weight sharing and pooling. Since a neuron in a convolutional layer learns the same local pattern over the TF plane, and pooling summarizes the local variations, CNN is more powerful than DNN at extracting local invariant patterns. Also, the weight sharing and pooling mechanisms in CNN greatly reduce the number of parameters relative to fully connected DNN, which enables much deeper hierarchical structures than DNN.

2.4. LSTM and BLSTM

CNN and DNN assume that the frames are independent. Though the context window contains neighboring frames, CNN and DNN are not able to exploit long term history or future information.

LSTM [20] are recurrent models, in which the current time prediction is dependent on all past time inputs. Also, by carefully designed gating structure, LSTM reduces the gradient vanishing and explosion problem associated with RNNs when training on long sequences. Therefore, LSTM are suitable for modeling speech, which is naturally sequence data. In addition, by concatenating LSTM layers, one on top of the other, the learned patterns possess two-dimensional depth, along the time dimension and the feature hierarchical dimension. For each layer, the LSTM progresses over time t by computing:

$$i_t = sigmoid(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \tag{9.1}$$
$$f_t = sigmoid\left(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}\right) \tag{9.2}$$
$$g_t = tanh\left(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}\right) \tag{9.3}$$
$$o_t = sigmoid(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \tag{9.4}$$
$$c_t = f_t c_{t-1} + i_t g_t \tag{9.5}$$
$$h_t = o_t \tanh(c_t) \tag{9.6}$$

in which i, f, o, g are the gates, c is the internal cell states, and h is the hidden states, which is also the input to the higher LSTM layer.

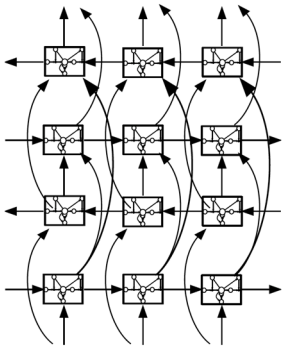

Figure 6: Deep BLSTM structure cited from [21]. The horizontal arrows denote two temporal directions, and other arrows show feature and layer stacking.

LSTM only takes advantage of past information. However, since we have the recording of the entire utterance, it's possible to make use of future information as well. Figure 6 depicts the BLSTM structure. In each BLSTM layer, there are a forward track, and a backward

track. The forward track unfolds the network from the first time instance to the last instance according to Eq.(9), whereas the backward track does the reverse by changing all t-1 to t+1in Eq.(9). The two tracks work in parallel, each keeps separate weights and biases. Their hidden states h are simply stacked together at each time t and are transmitted as input to the two tracks of the higher layer. The hidden states of the top layer are converted to state probabilities by a softmax classifier as in uni-directional LSTM. Therefore, to predict the identity of an HMM state at time t, all the historical and future frames are involved.

3. TIMIT phone recognition experiments

3.1. Database and system description

TIMIT database [22] is designed for English phone recognition from continuous read speech. This database is often used for algorithm verification and parameter tuning, because it has relatively small size, and thus moderate training time. As the typical use of this database, it is partitioned into a training set, a validation set, and a test set. The training set has 462 speakers, 8 utterances/speaker. The validation set consists of 50 speakers, totally 400 utterances, and the test set contains 192 sentences from 24 speakers. Each utterance is approximately 3.5 seconds long on average. For the purpose of data balance, the reading material of TIMIT is designed to be phonetically balanced among 61 target phones. Figure 7 shows these phones and their frequency counts in the training data (similar for the validation and test sets). At training time, we train a 3-state HMM-GMM model for each of the 61 phones, and each deep model thus has 183 target HMM states. At test time, the 61 phones are reduced to 39 phone categories, indicated by the "+" sign in Figure 7. It can be seen that except for the silence phone category, other phones are approximately uniformly distributed. For the language model, a phone bigram trained from the training set transcription was used for decoding.

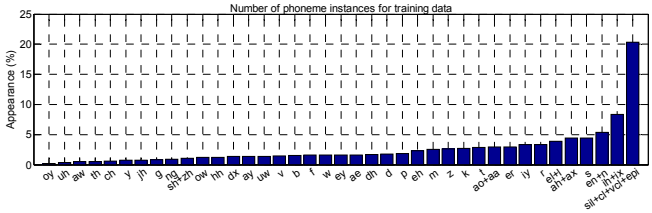

Figure 7: 61 TIMIT phones at training time and 39 phone categories at test time.

As illustrated in Section 2, 39 MFCC features including delta and acceleration per frame were extracted for HMM-GMM, and 40 FBANK features per frame were used for DNN and CNN. The context window for DNN contains 17 frames (8 frames on each side). It was found that increasing the context size for DNN did not affect the recognition accuracy. However, the context size makes a difference for CNN as will be shown. For LSTM and BLSTM, the input features consist of 40 FBANK and their delta and acceleration (120 features per frame). Since LSTM/BLSTM naturally makes use of context information, there is no frame concatenation. For LSTM/BLSTM, the first 5 frames of each sentence were removed, and the last frame was replicated 5 times and appended to the sentence. This is to facilitate predicting the current state label using future frames. The MFCC features were mean and variance normalized based on each sentence, and the FBANK features were normalized globally.

The seed HMM-GMM model has 32 mixtures per state. The DNN has 5 hidden layers with 2048 RELU neurons per layer. Batch normalization was conducted for the DNN before every RELU nonlinearity. The CNNs have variable number of layers and pooling size to show their effects. The LSTM/BLSTM also have variable number of layers and hidden size. The optimization rule is Adam for all deep models.

The feature extraction, HMM-GMM training, language model training and decoding used the HTK 3.4 ASR toolbox [23]. This toolbox does not support decoding with learned features (posterior state probabilities). Thus, the source code was modified to enable this functionality. HTK saved the features in HTK format by default, thus Python code adapted from [24] was used to convert the HTK FBANK features to Numpy arrays. All the deep models were trained using Pytorch. At test time, the posterior state probabilities are saved back to HTK feature format, and input to the modified decoder.

3.2. Results and analysis for clean data

In this section, the TIMIT data is recorded in clean environment without noise. Table 1 shows the phone recognition accuracy on the test set using HMM-GMM and HMM-DNN respectively. The learning rate for the DNN was set to 0.001 initially, and was multiplied by 0.1 whenever the validation set state classification accuracy failed to increase by 0.25% after each epoch. Dropout was also used after each RELU layer, with the dropout rate set to 0.2.

Table 1: Test set phone recognition accuracy using HMM-GMM and HMM-DNN.

Acoustic model	Phone accuracy (%)
HMM-GMM	71.2
HMM-DNN	78.1
HMM-DNN+dropout	78.8

DNN significantly improves the results compared with GMM. From now on, the HMM-DNN model is used as the baseline for other comparisons.

Next, two ResNet models were implemented. The ResNet code was modified from [25]. The 7x7 Conv layer and the 3x3 max pooling layer were removed from the original ResNet proposed in [9] since these layers did not improve the recognition performance. Table 2 lists the structure of the ResNet models. Each pair of brackets denote a residual block. The residual blocks did not use bottleneck operations. ResNet does not have explicit intermediate pooling layers in addition to the final average pooling. The downsampling is performed by conv2_1, conv3_1 and conv4_1 with a stride of 2. Batch normalization was inserted before each RELU.

Table 2: Structures of the ResNet-17 and ResNet-33 acoustic models.

layer name	17 layers	33 layers	stride size
conv1_x	$\begin{bmatrix}3x3, & 64\\ 3x3, & 64\end{bmatrix}$x2	$\begin{bmatrix}3x3, & 64\\ 3x3, & 64\end{bmatrix}$x3	1
conv2_x	$\begin{bmatrix}3x3, & 128\\ 3x3, & 128\end{bmatrix}$x2	$\begin{bmatrix}3x3, & 128\\ 3x3, & 128\end{bmatrix}$x4	2 for conv2_1
conv3_x	$\begin{bmatrix}3x3, & 256\\ 3x3, & 256\end{bmatrix}$x2	$\begin{bmatrix}3x3, & 256\\ 3x3, & 256\end{bmatrix}$x6	2 for conv3_1
conv4_x	$\begin{bmatrix}3x3, & 512\\ 3x3, & 512\end{bmatrix}$x2	$\begin{bmatrix}3x3, & 512\\ 3x3, & 512\end{bmatrix}$x3	2 for conv4_1
average pooling, 183-dimensional FC, softmax			

Table 3 shows the test set recognition accuracy. The context window size was also varied. The initial learning rate for ResNet-17 was set to 0.001, and for ResNet-33 was 0.0005, and was decreased to 0.1 of the previous value after each epoch if the validation frame accuracy failed to increase by 0.25%.

Table 3: ResNet recognition accuracy with different context window size and network depth.

Window size	**Network depth**	**Phone accuracy (%)**
17 frames	17 layers	81.1
17 frames	33 layers	80.4
31frames	17 layers	81.1
31frames	33 layers	81.7
41frames	17 layers	81.5
41frames	33 layers	81.3

The recognition accuracy using ResNet significantly outperforms DNN, which shows the benefits of learning local patterns. The performance does not improve as the network becomes deeper. This might be due to difficulties of training deeper networks, also might be due to overfitting. We will see in a later section that deeper architecture does improve recognition accuracy when the data is corrupted by noise, or when we have a large training set. Also, for clean data, the window size does not significantly affect the recognition performance.

We can visualize the learned features by ResNet and compare the feature discrimination with the FBANK features. For this purpose, we randomly took 20,000 frames (images) from the validation set, and obtained the learned features from the output of the average pooling layer in the trained ResNet. Figure 8 (left) plots the projected FBANK features using t-SNE [26] over different phone classes, and the right panel plots the learned features. It's clear that the learned features show much better discrimination over phone classes than the raw features. Also, within each phone class, the learned features tend to cluster tightly, which indicates strong inner-class invariance.

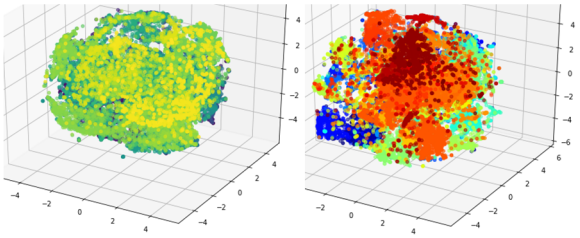

Figure 8: Raw FBANK features (left) vs. learned features by ResNet (right). Learned features show much better phonetic discrimination and invariance.

It's also interesting to visualize for different phones, which part of their TF plane is important for CNN. Figure 9 (top left) plots the TF plane of the vowel "ay" (as in bite), and the top right panel plots the saliency map by taking the gradient of the TF plane. It can be seen from the TF plane that the energy of this vowel concentrates in the low frequency region (indicated by the dark red area). Correspondingly, the saliency map shows that the low frequency region contains important patterns whereas the high frequency region does not carry useful information. In the bottom left panel, we plot the TF plane of the consonant "dh" (as in then). From the TF plane and the saliency map, it can be seen that the high frequency components and its contexts make more contribution for pattern learning whereas the low frequency components play less important role.

Next, a VGG network is implemented. All the Conv layers are 3x3. There are 2 Conv-64 layers, 2 Conv-128 layers, 2 Conv-256 layers, 3 Conv-512 layers and 2 FC layers with 2048 neurons per FC layer (not counting the classifier). However, for pooling, some literatures such as [27] propose that pooling along time degrades ASR performance. Thus, in one experiment, the temporal pooling was conducted only after the Conv-256 and Conv-512 layers, and in another experiment, the temporal pooling was performed after all consecutive Conv layers, as in the original VGG network [8]. In both experiments, the frequency pooling was conducted after all consecutive Conv layers. Batch normalization was performed before every RELU layer. The window size was fixed at 31 frames, since varying the window size did not bring significant impacts. The initial learning rate was 0.001 and the learning rate schedule was the same as in other network training.

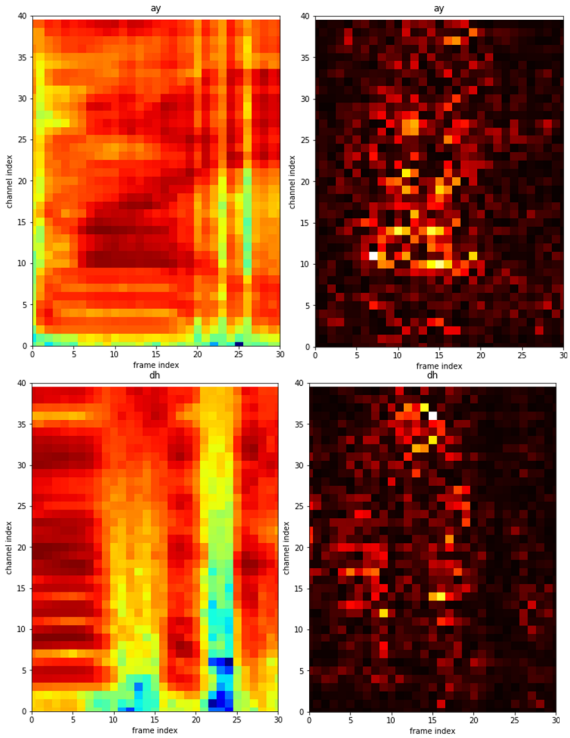

Figure 9: TF planes (left) and saliency maps (right) for the vowel "ay" (top) and the consonant "dh" (bottom).

Table 4 shows the test set recognition accuracy for the VGG network. It can be seen that using temporal pooling only in higher layers provides slightly better result.

Table 4: VGG network recognition accuracy with different temporal pooling positions.

Temporal pooling position	Phone accuracy (%)
After Conv-256 and Conv-512	81.7
After all Conv layers	81.2

Finally, the LSTM and BLSTM were implemented. Table 5 shows their performance varying with the depth and the hidden size. For BLSTM, the hidden size refers to the number of hidden cells per direction, and for LSTM, it means the hidden size per layer. Each batch contains 4 sentences, and the learning rate was initialized at 0.0005, with the same scheduling as in other network training.

As expected, BLSTM performs better than LSTM. For both network, the recognition accuracy increases as the hidden size gets larger and as the network becomes deeper.

Table 5: LSTM/BLSTM phone accuracy results.

Number of layers	Hidden size	LSTM Phone accuracy (%)	BLSTM Phone accuracy (%)
1	256	74.4	75.3
1	512	77.1	77.0
1	1024	77.6	77.9
2	256	78.2	79.1
2	512	79.6	80.0
2	1024	79.6	80.6
3	256	78.3	79.6
3	512	80.6	80.5
3	1024	80.8	80.8
4	256	79.2	80.7
4	512	80.4	81.3
4	1024	80.5	81.7

3.3. Results and analysis for reverberant data

Deep models not only improve the recognition accuracy for clean data, they also achieve better results when the data is corrupted. In this section, all the TIMIT data was reverberated as if the speakers were in a large room and were far away from the microphone. The reverberation effects were created using the tool SOX [28]. The previous experiments were repeated to test various effects for reverberant data. All the model structures and learning rules were identical as in the clean data cases.

Table 6 shows the results using HMM-DNN vs. HMM-GMM. Not surprisingly, the DNN substantially outperforms the GMM.

Table 6: Phone recognition accuracy for reverberant speech using HMM-GMM and HMM-DNN.

Acoustic model	Phone accuracy (%)
HMM-GMM	57.2
HMM-DNN	71.9

Table 6 shows the results using HMM-DNN vs. HMM-GMM. Not surprisingly, the DNN substantially outperforms the GMM.

Next, Table 7 and 8 lists the results for ResNet and VGG network. From Table 7, it can be observed that as the ResNet gets deeper, better recognition accuracy is achieved. From both Table 7 and 8, using large context window significantly increases the recognition accuracy. These results show that deeper CNNs with larger input size are more powerful at learning invariant local patterns from corrupted speech.

Table 7: ResNet phone accuracy for reverberant data.

Window size	Network depth	Phone accuracy (%)
17 frames	17 layers	73.3
17 frames	33 layers	73.3
31frames	17 layers	74.4
31frames	33 layers	75.2
41frames	17 layers	74.3
41frames	33 layers	75.5

Table 8: VGG network phone accuracy for reverberant data with different context window size. These results use temporal pooling only after the Conv-256 and Conv-512 layers.

Window size	Phone accuracy (%)
17 frames	73.1
31frames	75.6
41frames	75.4

Table 9 compares the effect of the temporal pooling positions for the VGG network. The window size was fixed at 31 frames. It can be seen that for reverberant data, using temporal pooling only in higher layers provides obviously better result.

Table 9: VGG network recognition accuracy with different temporal pooling positions for reverberant speech.

Temporal pooling position	Phone accuracy (%)
After Conv-256 and Conv-512	75.6
After all Conv layers	74.4

Finally, LSTM with 3 layers and 1024 cells per layer achieved 73.7% phone accuracy, and BLSTM with 4 layers and 1024 cells per direction obtained 74.9% phone accuracy for the reverberant speech.

4. Large vocabulary ASR experiments

A large database was collected by the author's company for training their ASR products. This database contains natural conversational telephone speech made in life environments. The training set has about 600 hours of data, 39000 speakers, out of which 5% was randomly chosen as the validation set. The test set has 361 speakers. For large vocabulary ASR tasks, state-of-the-art systems use context-dependent triphone HMMs, instead of monophone HMMs, and the triphone HMM states are tied according to their acoustic similarity using a decision tree method proposed in [29]. As a result, 3797 triphone HMM states were generated for DNN, CNN and LSTM targets. The DNN structure was identical to the ones in the TIMIT experiments. The context window size for the ResNet and VGG is 31 frames. The VGG temporal pooling was only conducted after the top 2 Conv layers. The LSTM and BLSTM have 4 hidden layers, with 512 cells in each layer/direction. At test time, a bigram language model was used for decoding. Table 10 summarizes the word accuracies using different acoustic models, and similar conclusions can be drawn as in the TIMIT cases.

Table 10: Word accuracy obtained by different deep models for a large vocabulary ASR task.

Acoustic model	Word accuracy (%)
HMM-GMM	69.7
HMM-DNN	75.4
HMM-VGG	79.6
HMM-ResNet17	81.5
HMM-ResNet33	82.1
HMM-LSTM	81.4
HMM-BLSTM	82.3

5. Conclusions and future work

This work extensively implemented a wide range of deep models for various ASR tasks. Also, this work covers almost all the knowledge presented by the course CS231n, including generative model (HMM-GMM), DNNs, two state-of-the-art CNNs originated from compute vision: VGG and ResNet, and sequence models LSTM/BLSTM. In addition to evaluating the recognition accuracy, two model visualization techniques were also employed: t-SNE and saliency maps. It's of great value for the author, who is an ASR researcher, to implement and gain insights into these cutting edge acoustic models, and it's extremely interesting to visualize and interpret speech data from a computer vision perspective.

There are several research topics to proceed with in future work. First is the connectionist temporal classification (CTC) model [30], which has the same structure as BLSTM, but has different loss function. In the current work, all the model targets are obtained by a seed HMM-GMM by forced alignment. CTC does not require HMM at all. Instead, it learns a probabilistic phone alignment by considering the probability of all the possible frame level phone sequences. Thus, CTC has very elegant and concise structure than the hybrid model studied in this work, and more importantly, CTC maximizes the sequence level probability of the phone transcription, rather than the frame level probability. Second direction is to combine CNN and LSTM to fully make use of their advantages jointly, learning local patterns as well as long term context dependencies. A tentative idea is to use the last layer CNN features as inputs to a LSTM and jointly train the CNN and LSTM. Finally, in addition to improving the acoustic model, LSTM can be used to improve the language model, which is equally important as the acoustic model for ASR.

6. References

[1] D. Jurafsky, J. H. Martin, “Speech and Language Processing, 2nd Edition,” Prentice Hall, 2008.

[2] S. E. Levinson, L. R. Rabiner, and M. M. Sondhi, “An introduction to the application of the theory of probabilistic functions of a Markov process to Automatic Speech Recognition,” in *Bell Syst. Tech. J.*, vol. 62, no. 4, pp. 1035-1074, 1983.

[3] L. Deng, P. Kenny, M. Lennig, V. Gupta, F. Seitz, P. Mermelsten, “Phonemic hidden Markov models with continuous mixture output densities for large vocabulary word recognition,” in *IEEE Trans. Acoust. Speech Signal Process*, vol. 39(7), pp. 1677–1681, 1991.

[4] L. Baum, T. Petrie, “Statistical inference for probabilistic functions of finite state Markov chains,” in *Ann. Math. Statist*. vol.37(6), pp. 1554-1563, 1966.

[5] F. Seide, G. Li, D. Yu, “Conversational speech transcription using context-dependent deep neural networks,” in *Proceedings of Annual Conference of International Speech Communication Association (INTERSPEECH)*, pp. 437-440, 2011.

[6] D. Yu, M. L. Seltzer, J. Li, J. T. Huang, F. Seide, “Feature learning in deep neural networks—studies on speech recognition tasks,” in *Proc. ICLR*, 2013.

[7] O. Abdel-Hamid, A. Mohamed, H. Jiang, L. Deng, G. Penn and D. Yu, “Convolutional neural networks for speech recognition,” in *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 10, Oct. 2014, pp. 1533-1545.

[8] K. Simonyan and A. Zisserman, “Very deep convolutional networks for large-scale image recognition,” in *ICLR*, 2015.

[9] K. He, X. Zhang, S. Ren, J. Sun, “Deep residual learning for image recognition,” in *CVPR* (2016).

[10] W. Xiong, J. Droppo, X. Huang, F. Seide, M. Seltzer, A. Stolcke, D. Yu, and G. Zweig, “The Microsoft 2016 conversational speech recognition system,” preprint at https://arxiv.org/abs/1609.03528.

[11] Y. Qian, M. Bi, T. Tan, K. Yu, "Very deep convolutional neural networks for noise robust speech recognition," in *IEEE/ACM Trans. Audio Speech Language Process.*, vol. 24, no. 12, pp. 2263-2276, 2016.

[12] A. Graves, A. Mohamed and G. Hinton, “Speech recognition with deep recurrent neural networks,” in *Acoustics, Speech and Signal Processing* (*ICASSP*), 2013 *IEEE International Conference on*, pp. 6645–6649. IEEE, 2013.

[13] M. Schuster and K. K. Paliwal, “Bidirectional recurrent neural networks,” *IEEE Transactions on Signal Processing* , vol. 45, pp. 2673–2681, 1997.

[14] S. Young et al., “The HTK book.” can be retrieved from http://htk.eng.cam.ac.uk/.

[15] S.S. Stevens, J. Volkmann and E.B. Newman, "A scale for the measurement of the psychological magnitude pitch," *J. Acoust. Soc. Am*., vol.8, no.3, pp.185-190, 1937.

[16] J. S. Bridle and M. D. Brown, "An experimental automatic word-recognition system," *JSRU Report*, no.1003, Joint Speech Research Unit, Ruislip, England, 1974.

[17] S. Memon, M. Lech and N. Maddage, "Speaker verification based on different vector quantization techniques with Gaussian mixture models," *Third Int. Conf. on Network and System Security,* pp. 403-408, 2009.

[18] J. Li, D. Yu, J. T. Huang, Y. Gong, “Improving wideband speech recognition using mixed bandwidth training data in CD-DNN-HMM,” In *Proceedings of the IEEE Spoken Language Technology Workshop* (*SLT*), pp. 131-136, 2012.

[19] T. N. Sainath, B. Kingsbury, A. Mohamed, B. Ramabhadran, “Learning filter banks within a deep neural network framework,” in *Proceedings of the IEEE Workshop on Automatic Speech Recognition and Understanding* (*ASRU*), pp. 297-302, 2013.

[20] S. Hochreiter and J. Schmidhuber, “Long Short-Term Memory,” *Neural Computation* , vol. 9, no. 8, pp. 1735–1780, 1997.

[21] A. Graves, N. Jaitly and A. Mohamed, “Hybrid speech recognition with deep bidirectional LSTM,” in *Automatic Speech Recognition and Understanding* (*ASRU*), *2013 IEEE Workshop*, pp. 273-278.

[22] V. Zue, S. Seneff, and J. Glass, "Speech database development at MIT: TIMIT and beyond," in *Speech Communication*, vol. 9, pp. 351-356, 1990.

[23] The HTK toolbox can be downloaded from http://htk.eng.cam.ac.uk/.

[24] The code used to convert HTK format features to Numpy arrays was modified from the source code at http://www.cs.cmu.edu/~chanwook/MySoftware/rm1_Spk-by-Spk_MLLR/rm1_PNCC_MLLR_1/rm1/python/sphinx/htkmfc.py.

[25] The ResNet code was adapted from https://github.com/pytorch/vision/blob/master/torchvision/models/resnet.py.

[26] L. J. P. van der Maaten and G. E. Hinton, “Visualizing data using t-SNE,” in *Journal of Machine Learning Research*, 9(Nov): pp.2431–2456, 2008.

[27] T. N. Sainath, A.-r. Mohamed, B. Kingsbury, and B. Ramabhadran, “Deep convolutional neural networks for lvcsr,” in *Acoustics, Speech and Signal Processing* (*ICASSP*), 2013 *IEEE International Conference on*. IEEE, 2013, pp. 8614–8618.

[28] The SOX tool used to reverberate the data can be retrieved from http://sox.sourceforge.net/.

[29] S. J. Young, J. J. Odell, P. C. Woodland, “Tree-based state tying for high accuracy acoustic modelling,” in *Proceedings of the ARPA Human Language Technology Workshop*, Morgan Kaufmann, Princeton NJ, pp. 307–312, 1994.

[30] Alex Graves, Santiago Fernandez, Faustino Gomez, and Jurgen Schmidhuber, “Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks,” in *Proceedings of the 23rd international conference on Machine learning* , 2006, pp.369–376.