

LECTURE 12

Deep Neural Networks for Speech Recognition

DEEE725 Speech Signal Processing Lab

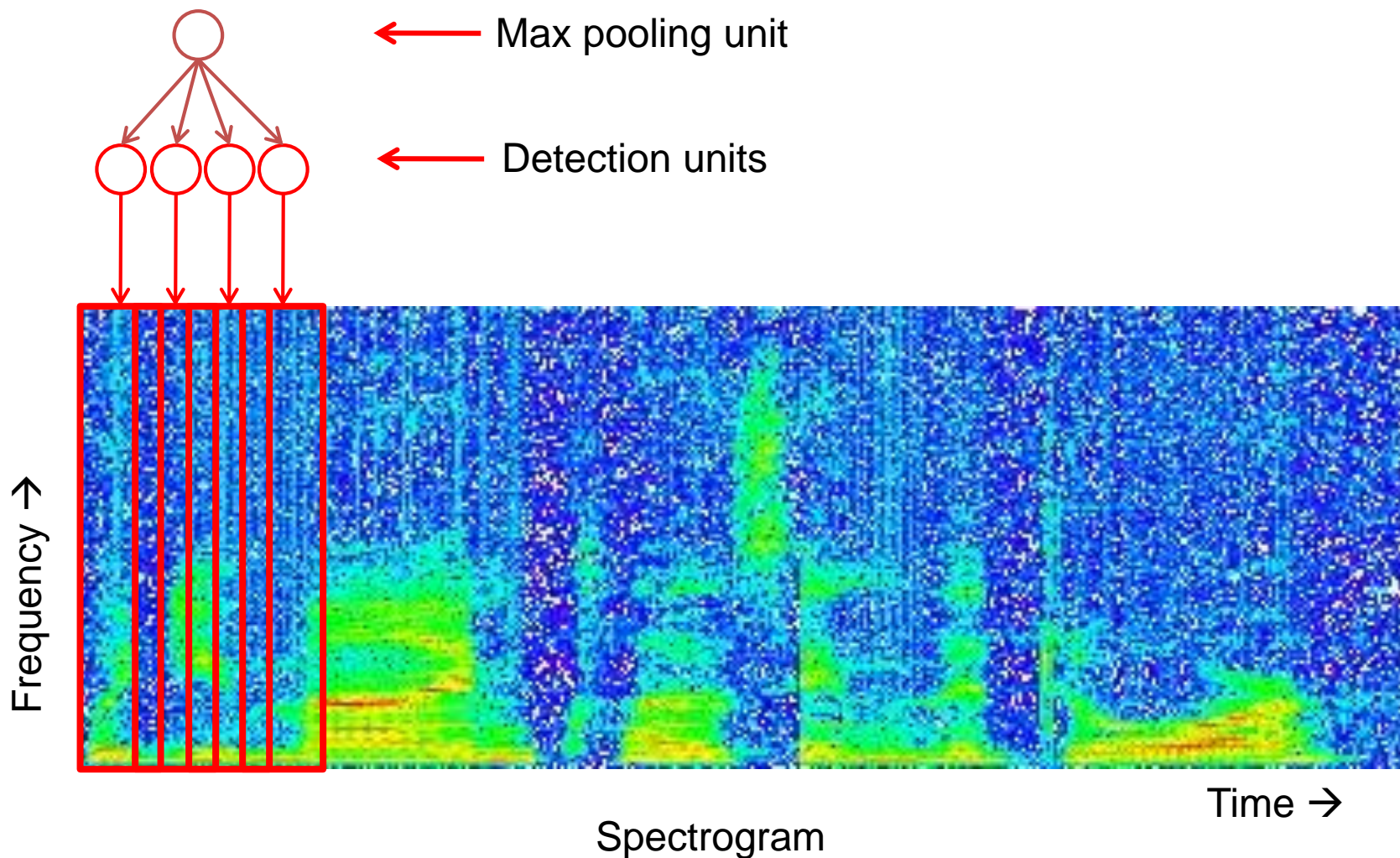
Gil-Jin Jang

Original slides from:

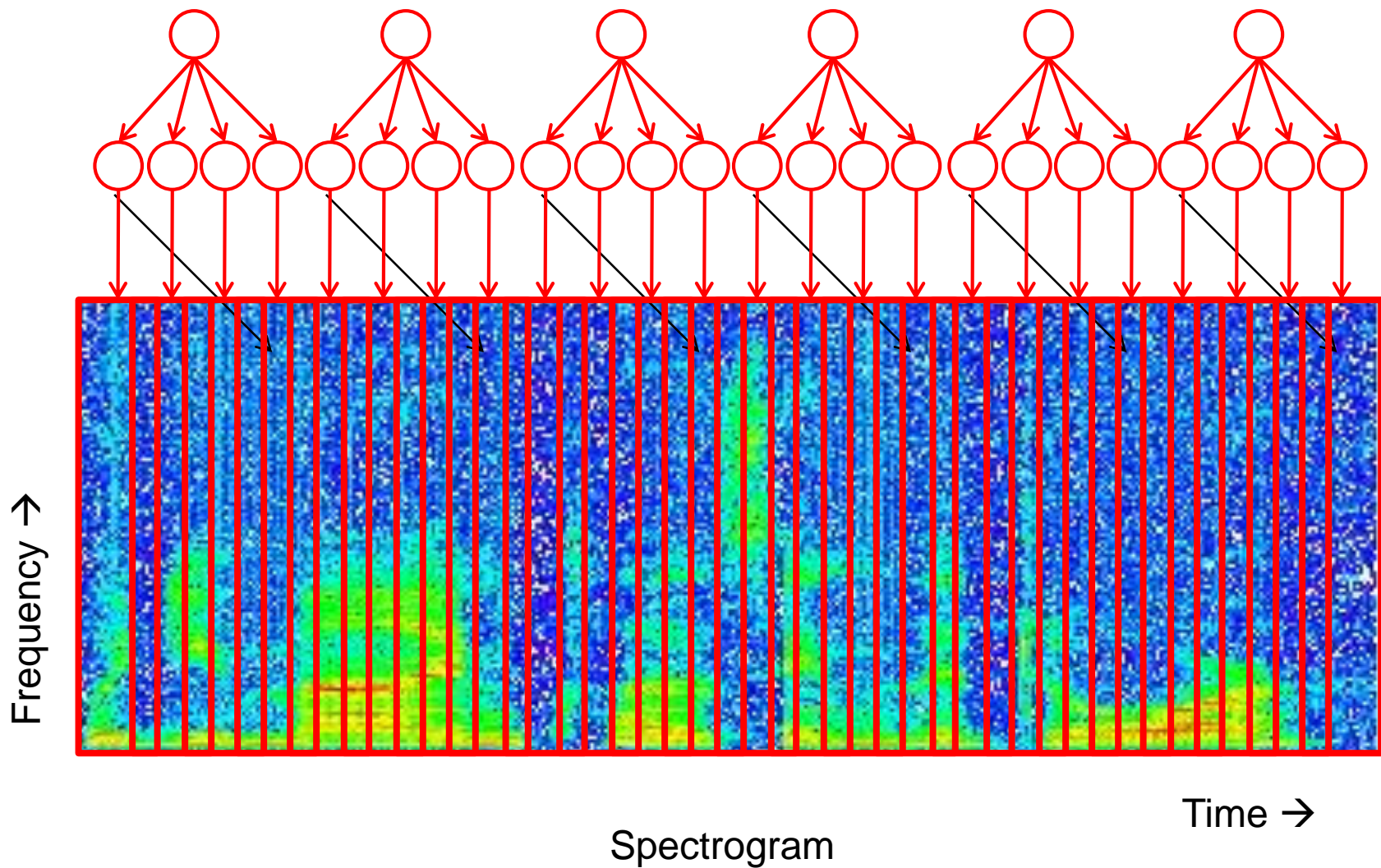
Andrew Ng, Honglak Lee, ECCV 2010 tutorial

Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang,
Li Deng, Gerald Penn, Dong Yu, IEEE Trans. Audio, Speech,
Lang. Proc., 2014

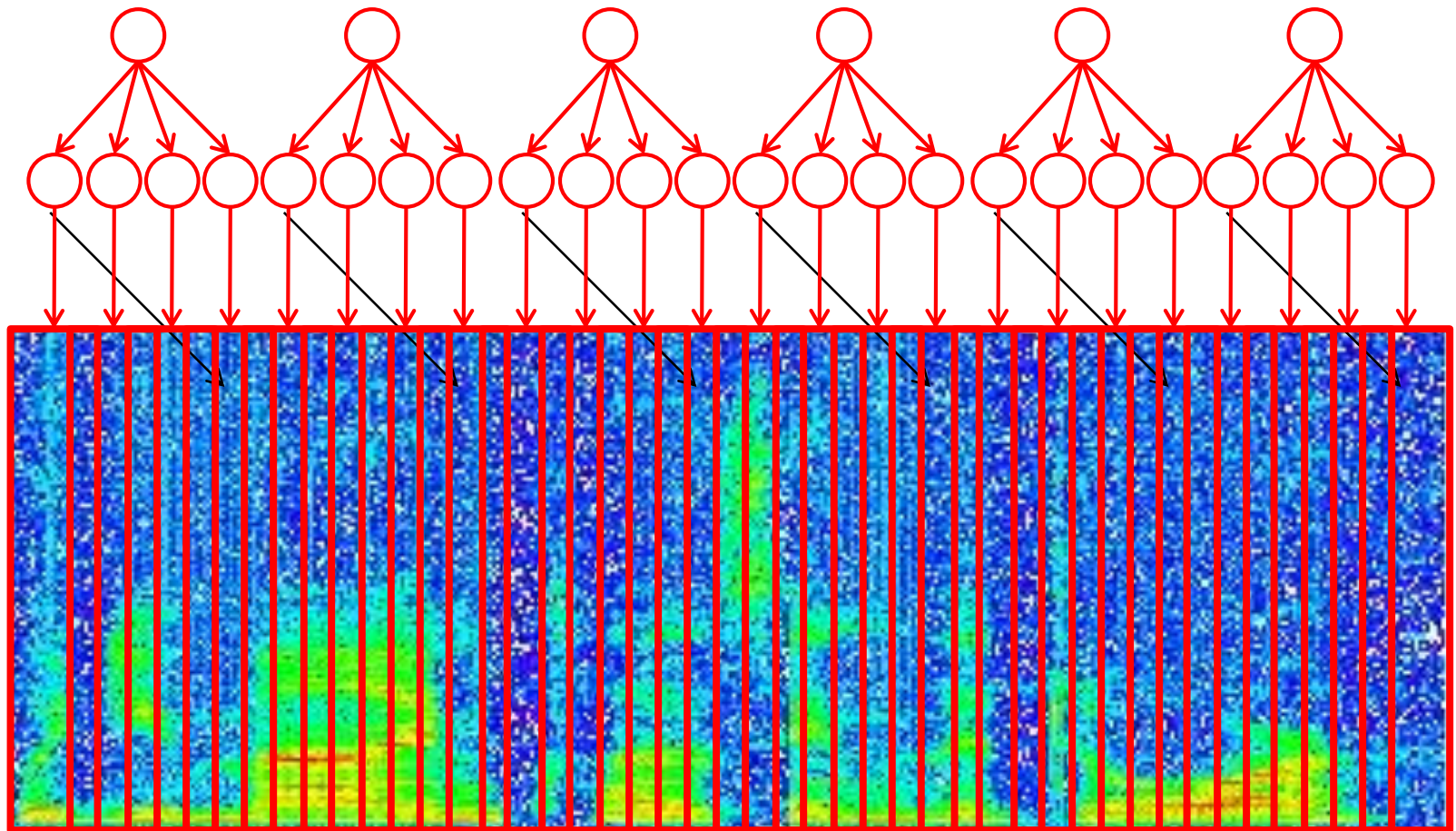
Convolutional DBN for audio



Convolutional DBN for audio

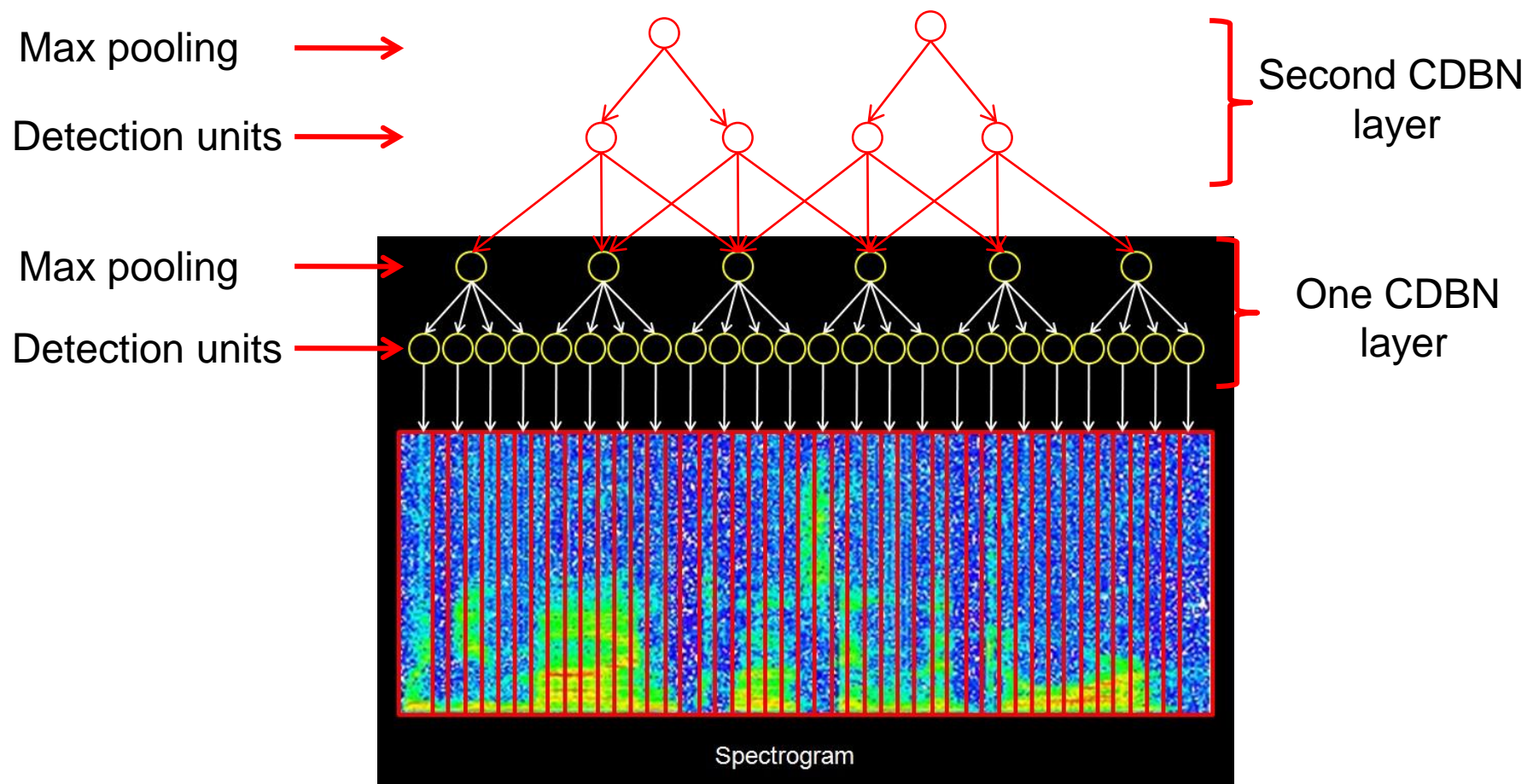


Convolutional DBN for audio



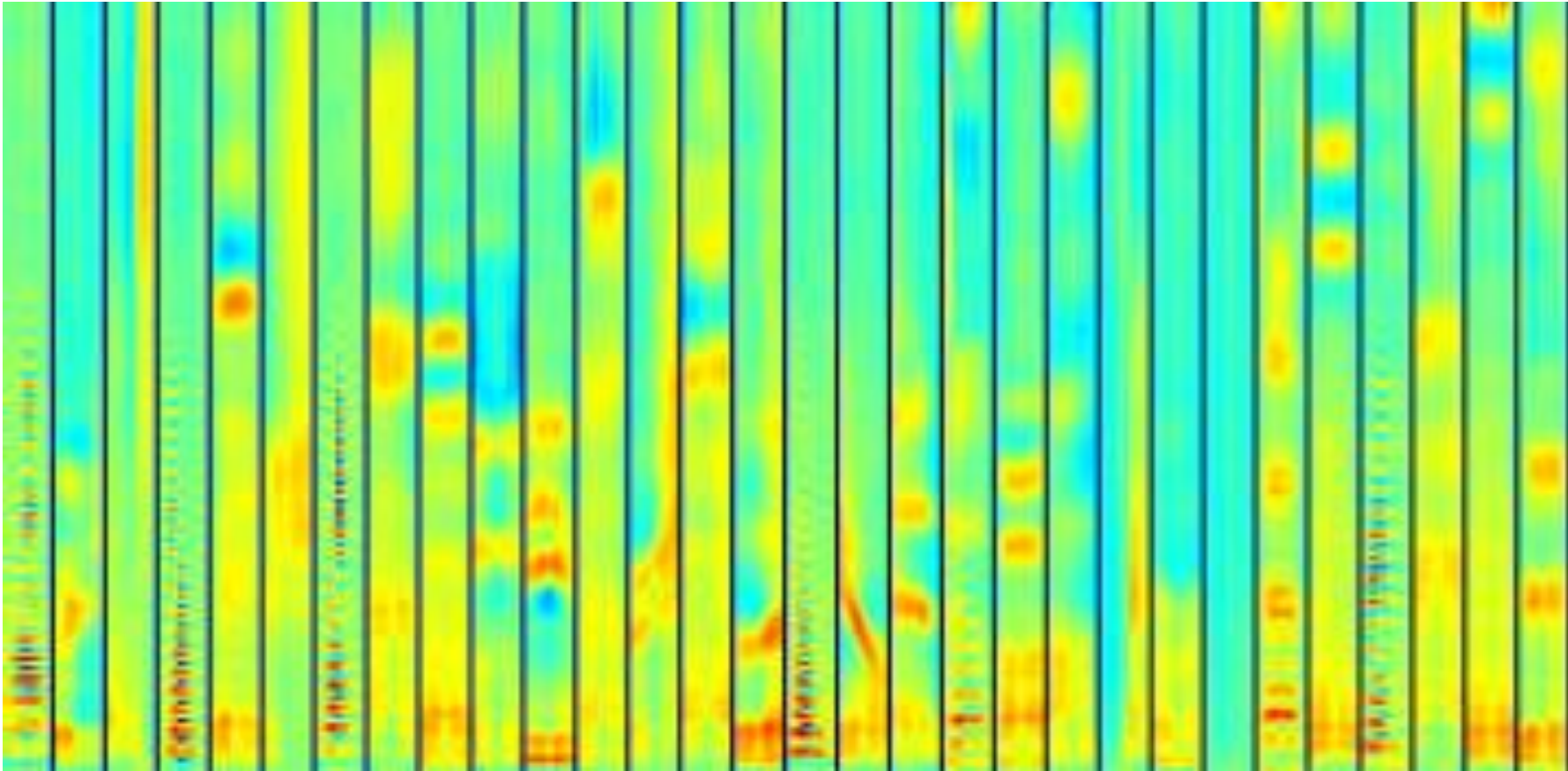
Spectrogram

Convolutional DBN for audio



CDBNs for speech

Trained on unlabeled TIMIT corpus



Learned first-layer bases

Experimental Results

Lee et al., 2009

Speaker identification

TIMIT Speaker identification	Accuracy
Prior art (Reynolds, 1995)	99.7%
Convolutional DBN	100.0%

Phone classification

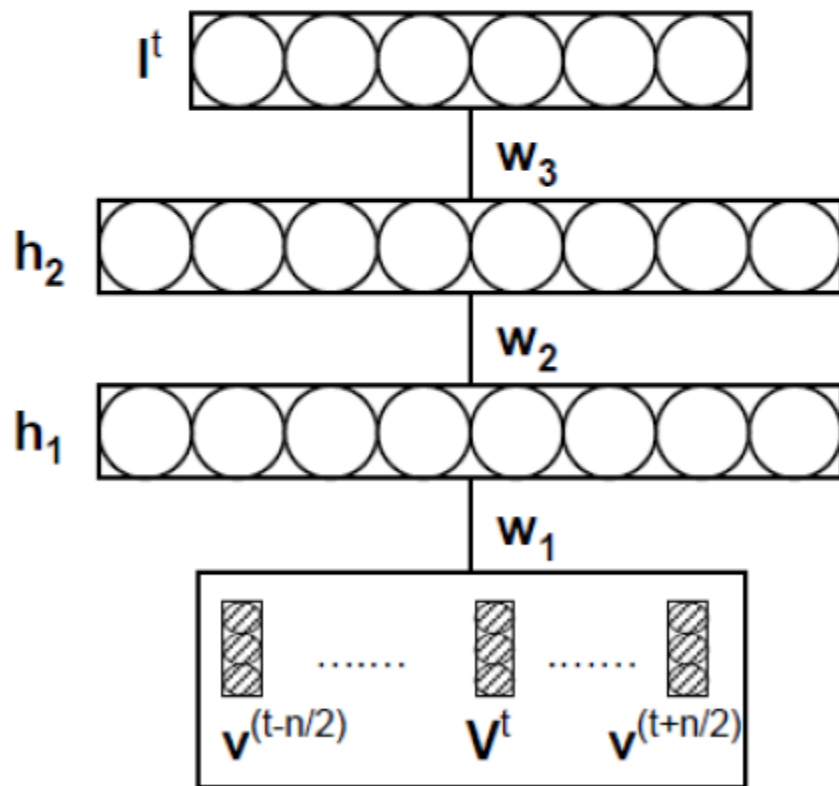
TIMIT Phone classification	Accuracy
Clarkson et al. (1999)	77.6%
Gunawardana et al. (2005)	78.3%
Sung et al. (2007)	78.5%
Petrov et al. (2007)	78.6%
Sha & Saul (2006)	78.9%
Yu et al. (2009)	79.2%
Convolutional DBN	80.3%

Slide credit: Andrew Ng, Honglak Lee

Phone recognition using DBNs

(Dahl et al., 2010)

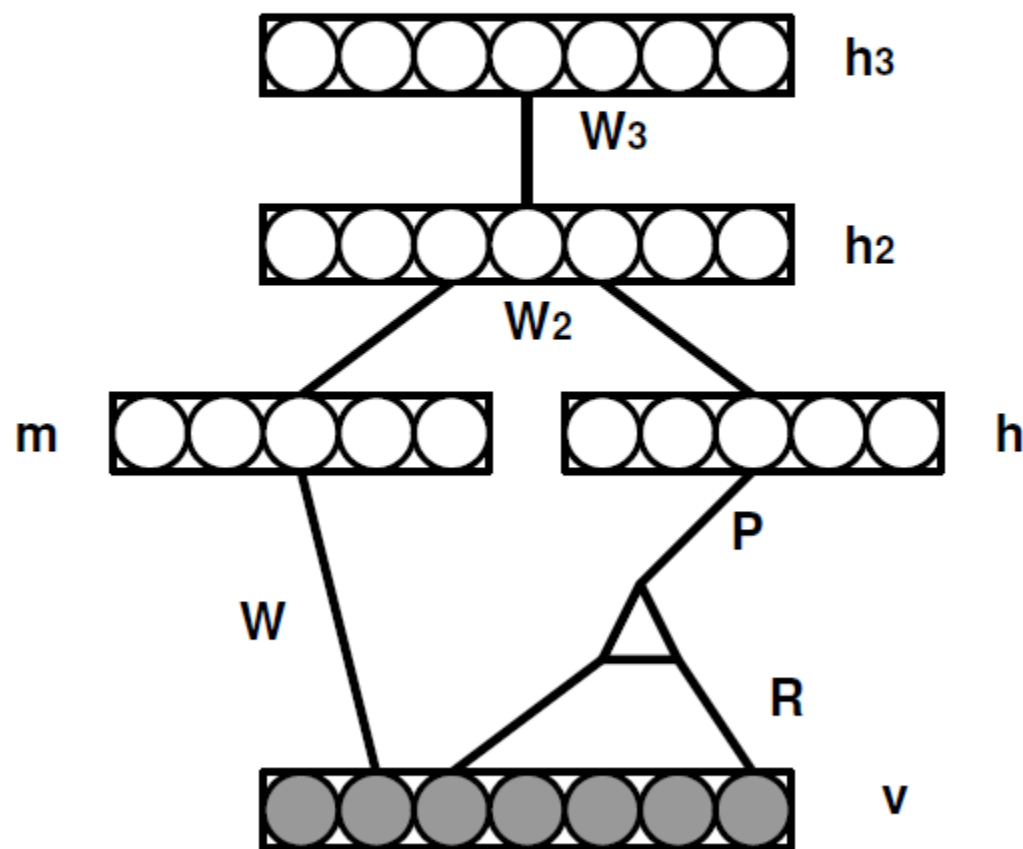
- Pre-training RBMs followed by fine-tuning with back propagation



Phone recognition using mcRBM

(Dahl et al., 2010)

- Mean-covariance RBM + DBN



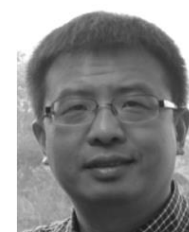
Mean-covariance RBM

$$E(\mathbf{v}, \mathbf{h}) = -\mathbf{d}^T \mathbf{h} - (\mathbf{v}^T \mathbf{R})^2 \mathbf{P} \mathbf{h},$$

Phone recognition results

(Dahl et al., 2010)

Method	PER
Stochastic Segmental Models	36.0%
Conditional Random Field	34.8%
Large-Margin GMM	33.0%
CD-HMM	27.3%
Augmented conditional Random Fields	26.6%
Recurrent Neural Nets	26.1%
Bayesian Triphone HMM	25.6%
Monophone HTMs	24.8%
Heterogeneous Classifiers	24.4%
Deep Belief Networks(DBNs)	23.0%
Triphone HMMs discriminatively trained w/ BMML	22.7%
Deep Belief Networks with mcRBM feature extraction	20.5%



[Ossama Abdel-Hamid](#), [Abdel-rahman Mohamed](#), [Hui Jiang](#), [Li Deng](#), [Gerald Penn](#), [Dong Yu](#)

Convolutional Neural Networks for Speech Recognition

IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING
VOL. 22, NO. 10, OCTOBER 2014

Ming-Han Yang

Outline

- Introduction
- Review : Deep Neural Networks
- Convolution Neural Networks and their use in ASR
- Experiments
- Conclusion

ASR tasks

- The aim of automatic speech recognition (ASR) is the transcription of human speech into spoken words.
- It is a very challenging task because human speech signals are highly variable due to various speaker attributes, different speaking styles, uncertain environmental noises, and so on.
- ASR, moreover, needs to map variable-length speech signals into variable-length sequences of words or phonetic symbols.

Hidden Markov Model

- It is well known that hidden Markov models (HMMs) have been very successful in handling variable length sequences as well as modeling the temporal behavior of speech signals using a sequence of states, each of which is associated with a particular probability distribution of observations.

Gaussian Mixture Models

- Gaussian mixture models (GMMs) have been, until very recently, regarded as the most powerful model for estimating the probabilistic distribution of speech signals associated with each of these HMM states.
- Meanwhile, the generative training methods of GMM-HMMs have been well developed for ASR based on the popular expectation maximization (EM) algorithm.
- In addition, a plethora of discriminative training methods, as reviewed in [1], [2], [3], are typically employed to further improve HMMs to yield the state-of-the-art ASR systems.

[1] H. Jiang, "Discriminative training for automatic speech recognition: A survey,"

[2] X. He, L. Deng, and W. Chou, "Discriminative learning in sequential pattern recognition—A unifying review for optimization-oriented speech recognition,"

[3] L. Deng and X. Li, "Machine learning paradigms for speech recognition: An overview,"

Artificial Neural Networks

- Very recently, HMM models that use artificial neural networks (ANNs) instead of GMMs have witnessed a significant resurgence of research interest [4], [5], [6], [7], [8], [9].
- Initially on the TIMIT phone recognition task with monophone HMMs for MFCC features, and shortly thereafter on several large vocabulary ASR tasks with triphone HMM models;

[4] G. E. Dahl, M. Ranzato, A. Mohamed, and G. E. Hinton, "Phone recognition with the mean-covariance restricted Boltzmann machine,"

[5] A. Mohamed, T. Sainath, G. Dahl, B. Ramabhadran, G. Hinton, and M. Picheny, "Deep belief networks using discriminative features for phone recognition,"

[6] D. Yu, L. Deng, and G. Dahl, "Roles of pre-training and fine-tuning in context-dependent DBN-HMMs for real-world speech recognition,"

[7] G. Dahl, D. Yu, L. Deng, and A. Acero, "Large vocabulary continuous speech recognition with context-dependent DBN-HMMs,"

[8] F. Seide, G. Li, X. Chen, and D. Yu, "Feature engineering in context-dependent deep neural networks for conversational speech transcription,"

[9] N. Morgan, "Deep and wide: Multiple layers in automatic speech recognition,"

Artificial Neural Networks

- In retrospect, the performance improvements of these recent attempts have been ascribed to their use of “deep” learning, a reference both to the number of hidden layers in the neural network as well as to the abstractness and, by some accounts, psychological plausibility of representations obtained in the layers furthest removed from the input, which hearkens back to the appeal of ANNs to cognitive scientists thirty years ago.
- A great many other design decisions have been made in these alternative ANN-based models to which significant improvements might have been attributed.

Artificial Neural Networks

- Even without deep learning, ANNs are powerful discriminative models that can directly represent arbitrary classification surfaces in the feature space without any assumptions about the data's structure.
- GMMs, by contrast, assume that each data sample is generated from one hidden expert (i.e., a Gaussian) and a weighted sum of those Gaussian components is used to model the entire feature space.
- ANNs have been used for speech recognition for more than two decades.

Artificial Neural Networks

- Early trials worked on static and limited speech inputs where a fixed-sized buffer was used to hold enough information to classify a word in an isolated speech recognition scheme
- They have been used in continuous speech recognition as feature extractors, in both the TANDEM approach and in so-called bottleneck feature methods , and also as nonlinear predictors to aid the recognition of speech units [25], [26].
- Their first successful application to continuous speech recognition, however, was in a manner that almost exactly parallels the use of GMMs now, i.e., as sources of HMM state posterior probabilities, given a fixed number of feature frames.

Artificial Neural Networks

- How do the recent ANN-HMM hybrids differ from earlier approaches? They are simply much larger.
- Advances in computing hardware over the last twenty years have played a significant role in the advance of ANN-based approaches to acoustic modeling because training ANNs with so many hidden units on so many hours of speech data has only recently become feasible.
- The recent trend towards ANN-HMM hybrids began with using restricted Boltzmann machines (RBMs), which can take (temporally) subsequent context into account.

ANN vs GMM

- Comparatively recent advances in learning through minimizing “contrastive divergence” enable us to approximate learning with RBMs.
- Compared to conventional GMM-HMMs, ANNs can easily leverage highly correlated feature inputs, such as those found in much wider temporal contexts of acoustic frames, typically 9-15 frames.
- Hybrid ANN-HMMs also now often directly use log mel-frequency spectral coefficients without a decorrelating discrete cosine transform, DCTs being largely an artifact of the decorrelated mel-frequency cepstral coefficients (MFCCs) that were popular with GMMs.
- All of these factors have had a significant impact upon performance.

Artificial Neural Networks

- This historical deconstruction is important because the premise of the present paper is that very wide input contexts and domain-appropriate representational invariance are so important to the recent success of neural-network-based acoustic models that an ANN-HMM architecture embodying these advantages can in principle outperform other ANN architectures of potentially unlimited depth for at least some tasks.
- We present just such a novel architecture below, which is based upon convolutional neural networks (CNNs) [31].

CNN

- CNNs are among the oldest deep neural-network architectures , and have enjoyed great popularity as a means for handwriting recognition.
- A modification of CNNs will be presented here, called limited weight sharing, however, which to some extent impairs their ability to be stacked unboundedly deep.
- We moreover illustrate the application of CNNs to ASR in detail, and provide additional experimental results on how different CNN configurations may affect final ASR performance.

Why HMM? Not TDNN?

- CNNs have been applied to acoustic modeling before, notably by [33] and [34], in which convolution was applied over windows of acoustic frames that overlap in time in order to learn more stable acoustic features for classes such as phone, speaker and gender.
- Weight sharing over time is actually a much older idea that dates back to the so-called time-delay neural networks (TDNNs) of the late 1980s, but TDNNs had emerged initially as a competitor with HMMs for modeling time-variation in a “pure” neural-network-based approach.
- That purity may be of some value to the aforementioned cognitive scientists, but it is less so to engineers.

Why HMM? Not TDNN?

- As far as modeling time variations is concerned, HMMs do relatively well at this task; convolutional methods, i.e., those that use neural networks endowed with weight sharing, local connectivity and pooling (properties that will be defined below), are probably overkill, in spite of the initially positive results of [35].
- We will continue to use HMMs in our model for handling variation along the time axis, but then apply convolution on the frequency axis of the spectrogram.

CNN

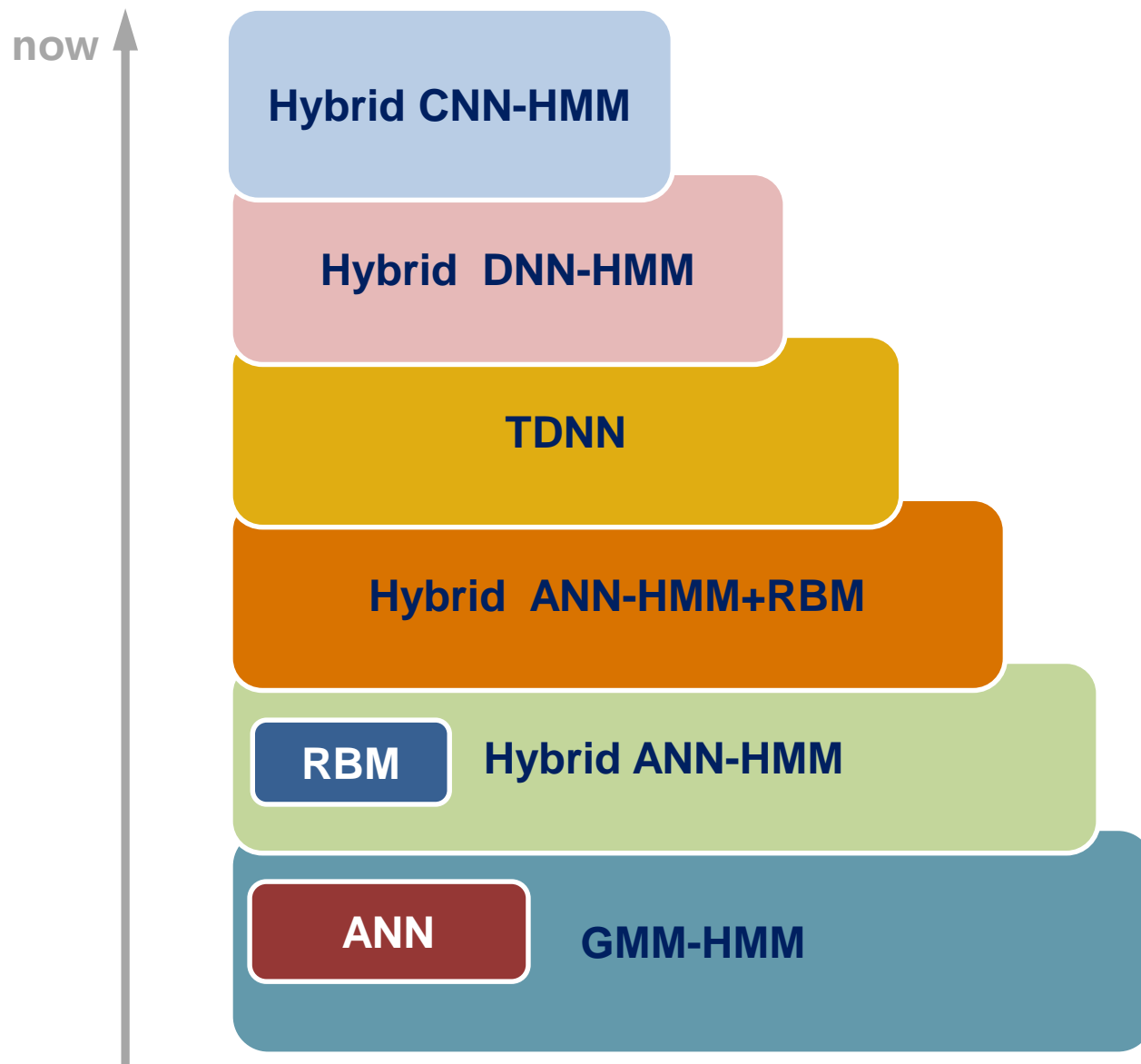
- This endows the learned acoustic features with a tolerance to small shifts in frequency, such as those that may arise from differing vocal tract lengths, and has led to a significant improvement over DNNs of similar complexity on TIMIT speaker-independent phone recognition, with a relative phone error rate reduction of about 8.5%.
- Learning invariant representations over frequency (or time) are notoriously more difficult for standard DNNs.
- Deep architectures have considerable merit. They enable a model to handle many types of variability in the speech signal.

CNN

- The work of [29], [36] shows that the feature representations used in the upper hidden layers of DNNs are indeed more invariant to small perturbations in the input, regardless of their putative deep structural insight or abstraction, and in a manner that leads to better model generalization and improved recognition performance, especially under speaker and environmental variations.
- The more crucial question we have undertaken to answer is whether even better performance might be attainable if some representational knowledge that arises from a careful study of the empirical domain can be used to explicitly handle the variations in question. 1

Vocal tract length normalization

- Vocal tract length normalization (VTLN) is another very good example of this.
- VTLN warps the frequency axis based on a single learnable warping factor to normalize speaker variations in the speech signals, and has been shown [41], [16] to further improve the performance of DNN-HMM hybrid models when applied to the input features.
- More recently, the deep architecture taking the form of recurrent neural networks, even with unstacked single-layer variants, have been reported with very competitive error rates [42].



Review : Deep Neural Networks

- A deep neural network (DNN) refers to a feedforward neural network with more than one hidden layer.

$$o_i^{(l)} = \underbrace{\sigma}_{\text{sigmoid() or tanh()}} \left(\sum_j o_j^{(l-1)} w_{j,i}^{(l)} + \underbrace{w_{0,i}^{(l)}}_{\text{bias}} \right)$$

- For a multi-class classification problem, the posterior probability of each class can be estimated using an output softmax layer.

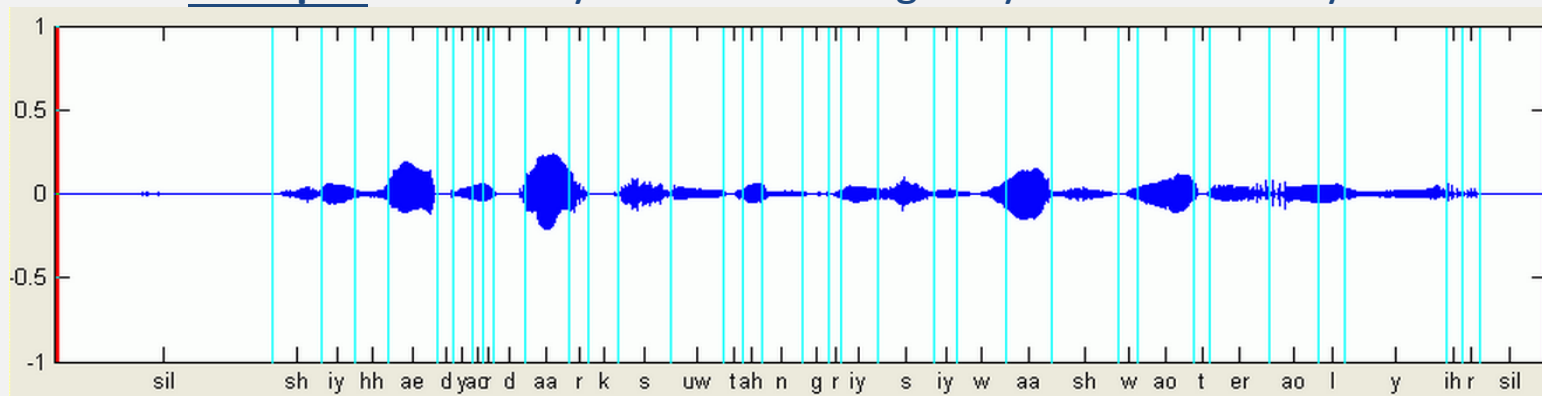
$$y_i = \frac{\exp(o_i^{(L)})}{\sum_j \exp(o_j^{(L)})}$$

where $o_i^{(L)}$ is computed as $o_i^{(L)} = \mathbf{o}^{(L-1)} \cdot \mathbf{w}_i^{(L)}$

Review : DNN-HMM model

- In the hybrid DNN-HMM model, the DNN replaces the GMMs to compute the HMM state observation likelihoods.
- In the training stage, forced alignment is first performed to generate a reference state label for every frame.

example : She had your dark suit in greasy wash water all year.



Picture: <http://mirlab.org/jang/mir/speechAssessme>

- These labels are used in supervised training to minimize the cross-entropy function.

$$Q(\{\mathbf{W}^{(l)}\}) = - \sum_i d_i \log y_i$$

Review : DNN-HMM model

- If we use the stochastic gradient descent algorithm to minimize the objective function, for each training sample or mini-batch, each weight matrix update can be computed as:

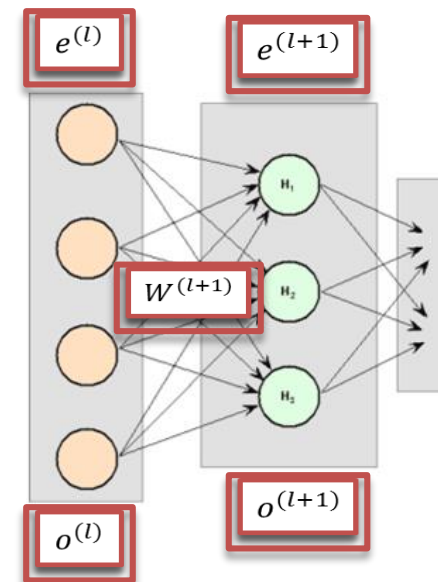
$$\Delta \mathbf{W}^{(l)} = \underbrace{\epsilon}_{\text{learning rate}} \cdot \left(\mathbf{o}^{(l-1)} \right)' \underbrace{\mathbf{e}^{(l)}}_{\text{DNN output phoneme states}} \quad (l = 1, 2, \dots, L)$$

training data phoneme states

DNN output phoneme states

$$\mathbf{e}^{(L)} = \mathbf{d} - \mathbf{y}$$

$$\mathbf{e}^{(l)} = \left(\mathbf{e}^{(l+1)} \left(\mathbf{W}^{(l+1)} \right)' \right) \bullet \underbrace{\mathbf{o}^{(l)} \bullet \left(1 - \mathbf{o}^{(l)} \right)}_{\text{Derivative of sigmoid}} \quad (l = L - 1, \dots, 2, 1)$$



Review : DNN pre-training

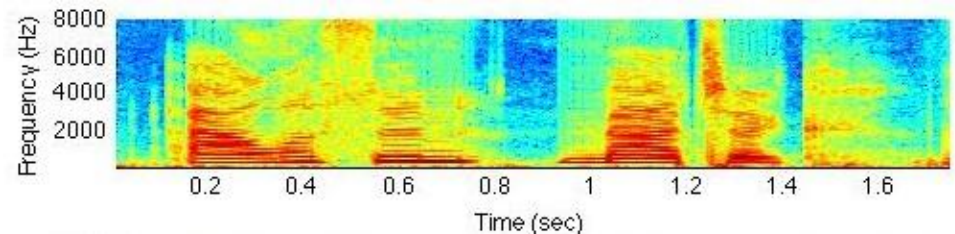
- Initializes all weight , especially when the amount of training data is limited and when no constraints are imposed on the DNN weights.
- One popular method to pre-train DNNs uses the restricted Boltzmann machine (RBM) as a building block.
- After learning, all RBM weights can be used as a good initialization for one DNN layer.



Convolution Neural Networks and their use in ASR

Organization of the Input Data to the CNN

- CNN introduces a special network structure, which consists of alternating so-called convolution and pooling layers.
- CNNs run a small window over the input image at both training and testing time, the weights of the network that looks through this window can learn from various features of the input data regardless of their absolute position within the input.
- The input “image can loosely be thought of as a spectrogram, with static, delta and delta-delta features (i.e., first and second temporal derivatives) serving in the roles of red, green and blue



How to organize speech feature vectors into feature maps

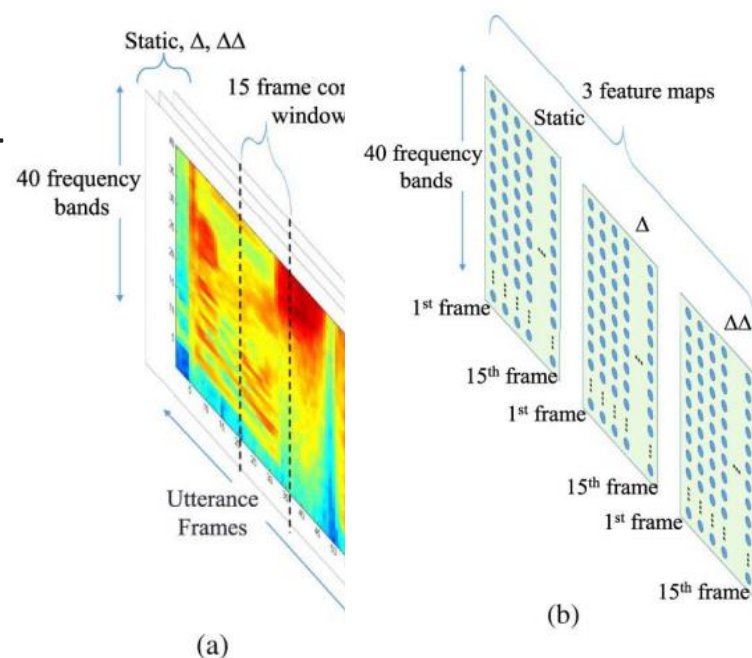
- We need to use inputs that **preserve locality** in both axes of frequency and time.
- **Time** : presents no immediate problem from the standpoint of locality. Like other DNNs for speech, a single window of input to the CNN will consist of a wide amount of context (9–15 frames).
- **Frequency** : the conventional use of MFCCs does present a major problem because the discrete cosine transform projects the spectral energies into a new basis that may not maintain locality.

How to organize speech feature vectors into feature maps

- In this paper, we shall use the log-energy computed directly from the **mel-frequency spectral coefficients** (i.e., with no DCT), which we will denote as MFSC features.
- **MFSC features will be used to represent each speech frame**, along with their deltas and delta-deltas (in order to describe the acoustic energy distribution in each of several different frequency bands.)
- There exist several different alternatives to organizing these MFSC features into maps for the CNN.

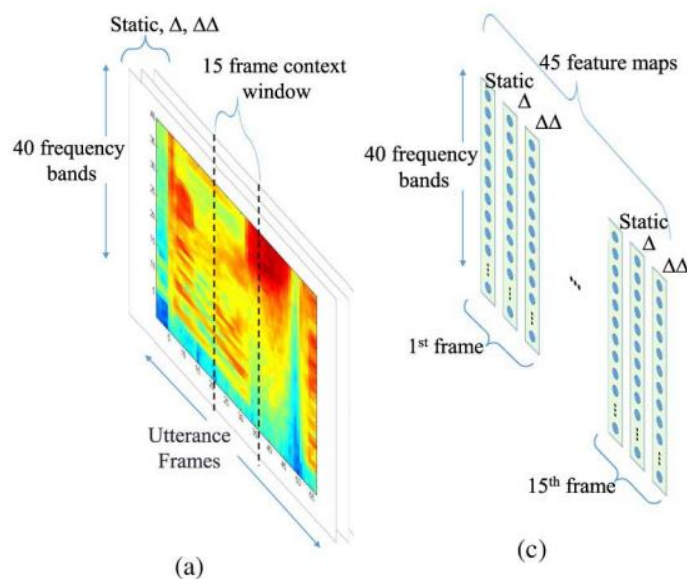
Organizing MFSC features : Fig.1(b)

- They can be arranged as three 2-D feature maps, each of which represents MFSC features (static, delta and delta-delta) distributed along both **frequency** (using the frequency band index) and **time** (using the frame number within each context window).
- In this case, a two-dimensional convolution is performed to normalize both frequency and temporal variations simultaneously.

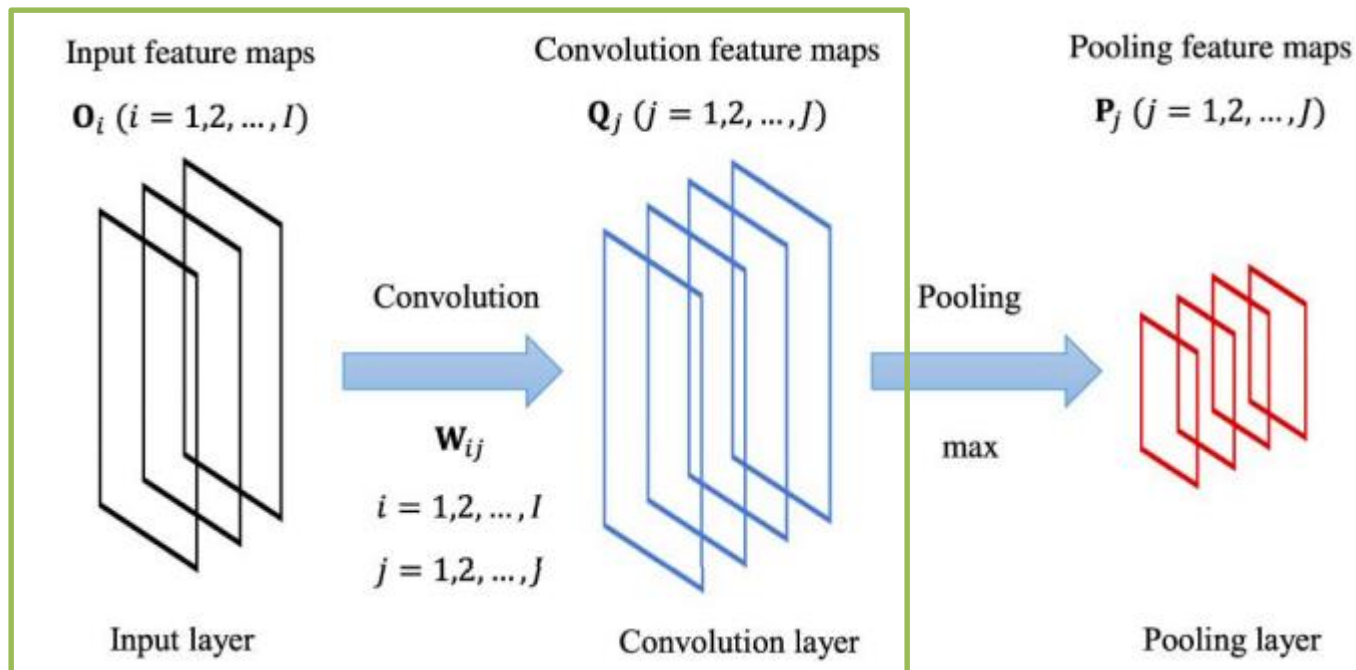


Organizing MFSC features : Fig.1(c)

- Alternatively, we may only consider normalizing **frequency** variations.
- For example, if the context window contains 15 frames and 40 filter banks are used for each frame, we will construct 45 (i.e., 15 times 3) 1-D feature maps, with each map having 40 dimensions



Convolution ply



$$q_{j,m} = \sigma \left(\sum_{i=1}^I \sum_{n=1}^F o_{i,n+m-1} w_{i,j,n} + w_{0,j} \right), \quad (j = 1, \dots, J)$$

Convolution ply

- Both O_i and $W_{i,j}$ are vectors if one dimensional feature maps are used, and are matrices if two dimensional feature maps are used.

$$Q_j = \sigma \left(\sum_{i=1}^I O_i * \mathbf{w}_{i,j} \right) \quad (j = 1, \dots, J)$$

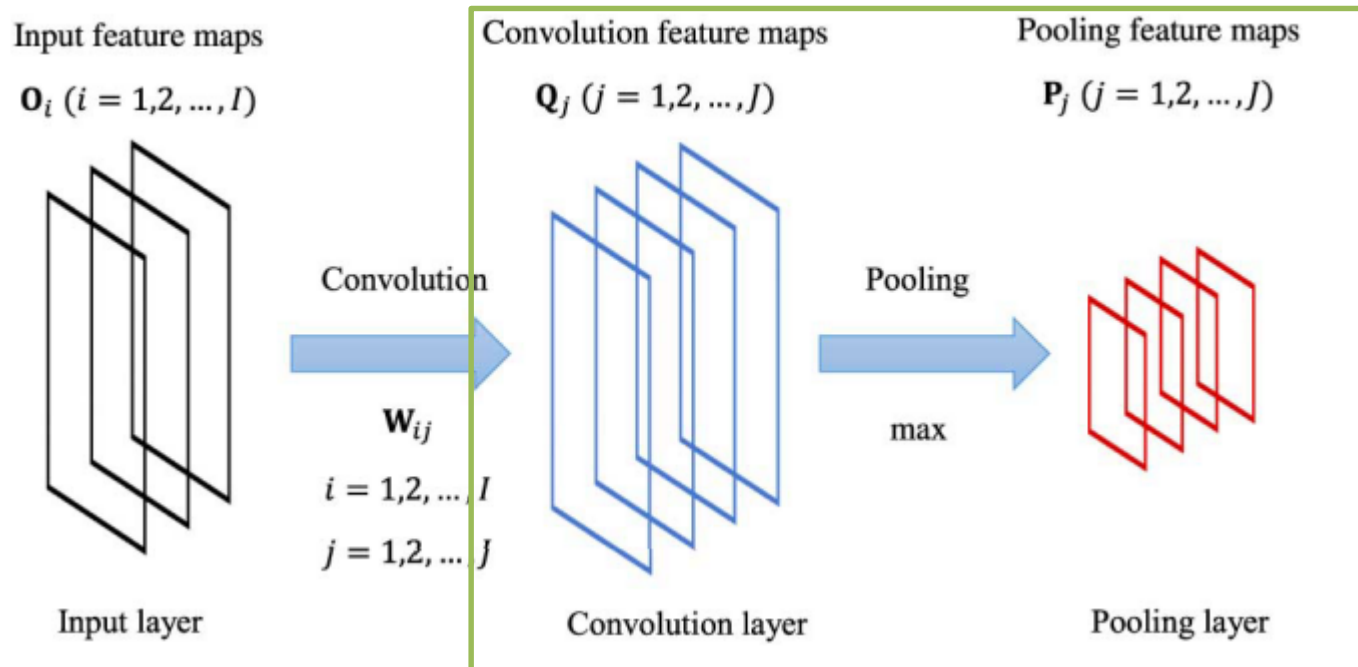
$W_{i,j}$: each local matrix O_i : the i-th input feature map

- Note that, in this presentation, the number of feature maps in the convolution ply directly determines the number of local weight matrices that are used in the above convolutional mapping.

Convolution ply

- The convolution operation itself produces lower-dimensional data—each dimension decreases by filter size minus one.
- But we can pad the input with dummy values (both dummy time frames and dummy frequency bands) to preserve the size of the feature maps.
- All units in the same feature map share the same weights but receive input from different locations of the lower layer.

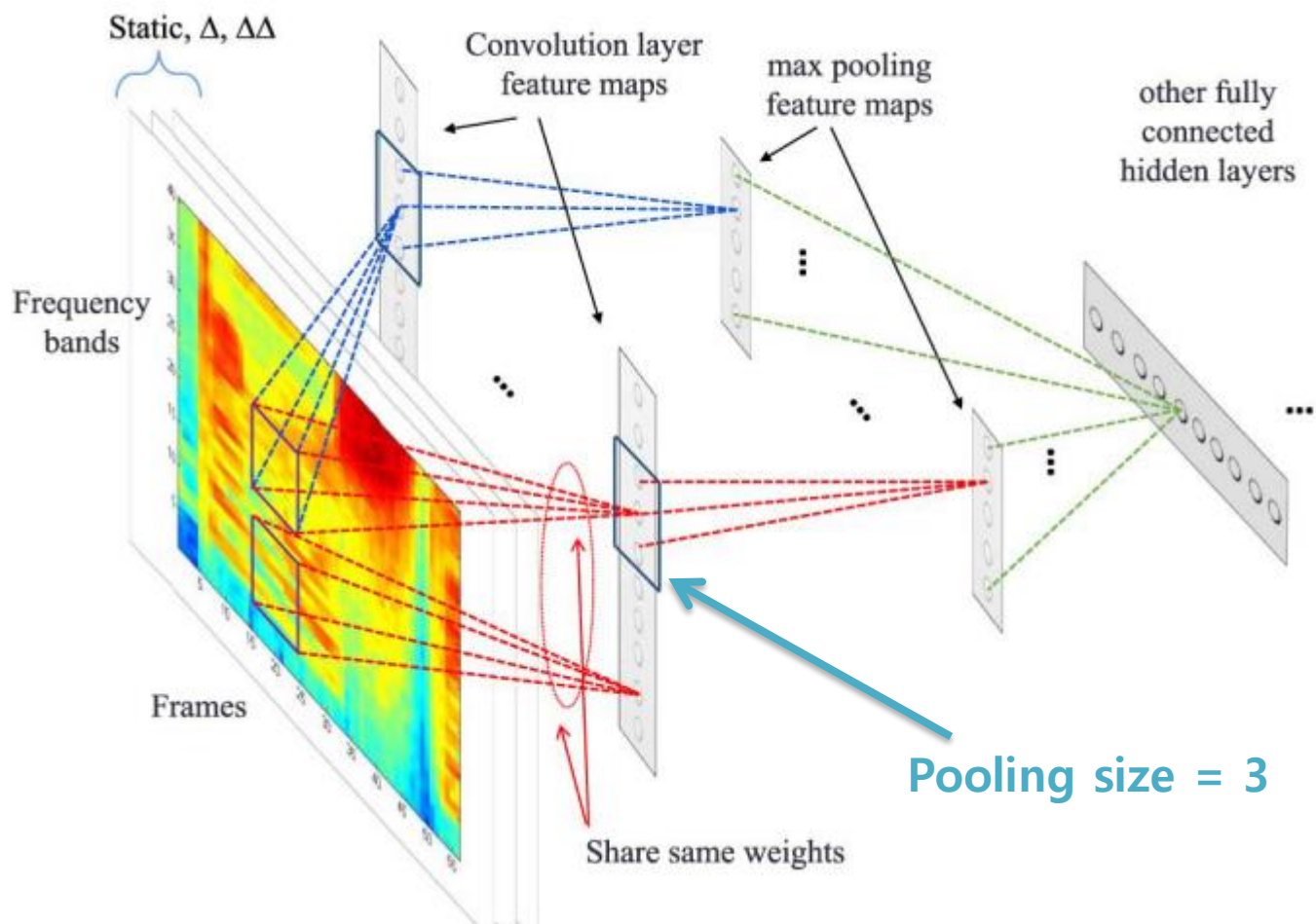
Pooling ply



- It has the same number of feature maps as the number of feature maps in its convolution ply, but each map is smaller.
- The purpose of the pooling ply is to reduce the resolution of feature maps.

Pooling ply

- It is usually a simple function such as maximization or averaging.
- Max pooling : $p_{i,m} = \max_{n=1}^G q_{i,(m-1) \times s + n}$ ✓ Better performance
- Mean pooling : $p_{i,m} = r \sum_{n=1}^G q_{i,(m-1) \times s + n}$
- **G : pooling size** ← 影像辨識中, 通常不會重疊
- **s : shift size,** determines the overlap of adjacent pooling windows ←
- r : a scaling factor that can be learned



Learning weight in CNN

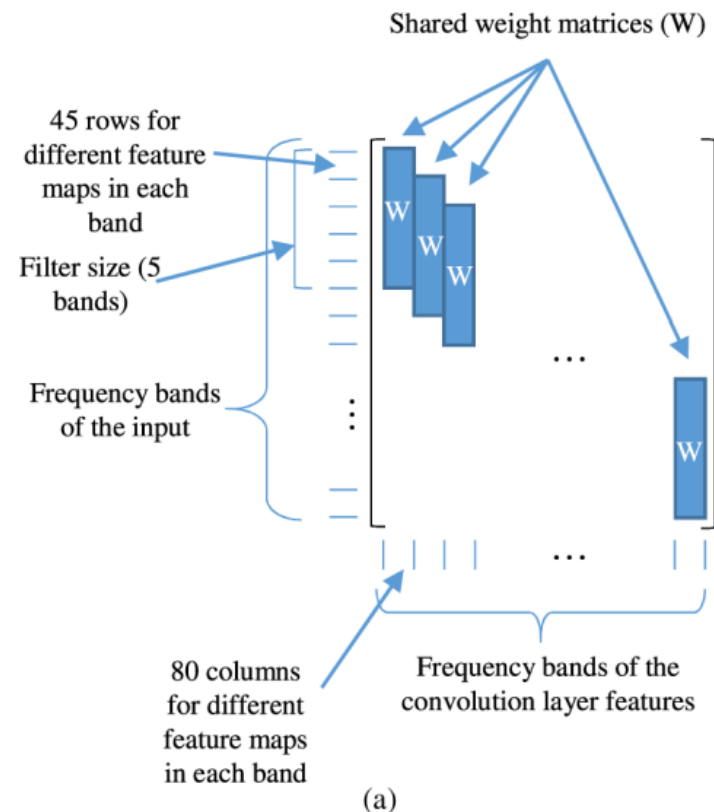
- All weights in the convolution ply can be learned using the same error back-propagation algorithm, but some special modifications are needed to take care of sparse connections and weight sharing.
- In order to illustrate the learning algorithm for CNN layers, let us first represent the convolution operation in eq. (9) in the same mathematical form as the fully connected ANN layer.

$$Q_j = \sigma \left(\sum_{i=1}^I O_i * \mathbf{w}_{i,j} \right) \quad (j = 1, \dots, J)$$


Learning weight in CNN

- When one-dimensional feature maps are used, the convolution operations can be represented as a simple matrix multiplication by introducing a large sparse weight matrix \widehat{W} .

$$\mathbf{W} = \begin{bmatrix} w_{1,1,1} & w_{1,2,1} & \dots & w_{1,J,1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{I,1,1} & w_{I,2,1} & \dots & w_{I,J,1} \\ \vdots & \vdots & \ddots & \vdots \\ w_{I,1,2} & w_{I,2,2} & \dots & w_{I,J,2} \\ \vdots & \vdots & \ddots & \vdots \\ w_{I,1,F} & w_{I,2,F} & \dots & w_{I,J,F} \end{bmatrix}$$



Learning weight in CNN

- The input and the convolution feature maps are also vectorized as row vectors \hat{o} and \hat{q} .
- One single row vector \hat{o} is created from all of the input feature maps O_i ($i = 1, \dots, I$) as follows: $\hat{o} = [\mathbf{v}_1 | \mathbf{v}_2 | \dots | \mathbf{v}_M]$

- Therefore, the convolution ply outputs computed in eq. (9) can be equivalently expressed as a weight vector:

$$Q_j = \sigma \left(\sum_{i=1}^I O_i * \mathbf{w}_{i,j} \right) \quad (j = 1, \dots, J) \longrightarrow \hat{\mathbf{q}} = \sigma(\hat{o} \hat{\mathbf{W}})$$

Learning weight in CNN

- The update for \hat{W} is similarly calculated as:

$$\Delta \hat{W} = \epsilon \cdot \hat{o}' e.$$

- The difference is that for the shared weights here, we sum them in their updates according to:

$$\Delta w_{i,j,n} = \sum_m \Delta \hat{W}_{i+(m+n-2) \times I, j+(m-1) \times J}$$

- Biases can be handled by adding one row to the matrix \hat{W} to hold the bias values replicated among all convolution ply bands and adding one element with a value of one to the vector \hat{o}

$$\begin{aligned} e^{(L)} &= d - y \\ e^{(l)} &= \left(e^{(l+1)} \left(W^{(l+1)} \right)' \right) \bullet o^{(l)} \bullet \left(1 - o^{(l)} \right) \\ (l &= L - 1, \dots, 2, 1) \end{aligned}$$

Learning weight in CNN

- Since the pooling ply has no weights, no learning is needed here. However, **the error signals should be back-propagated to lower plies through the pooling function.**
- In the case of **max-pooling**, the error signal is passed backwards only to the most active (largest) unit among each group of pooled units.
- That is, the error signal reaching the lower convolution ply can

be computed as:
$$e_{i,n}^{\text{low}} = \sum_m e_{i,m} \cdot \delta(u_{i,m} + (m-1) \times s - n)$$

$$\delta(x) = \begin{cases} 1, & \text{if } x = 0 \\ 0, & \text{otherwise} \end{cases}$$

- $u_{i,m}$ is the index of the unit with the maximum value among the pooled units and is defined as :

$$u_{i,m} = \underset{n=1}{\operatorname{argmax}}^G q_{i,(m-1) \times s + n}$$

Pretraining CNN Layers

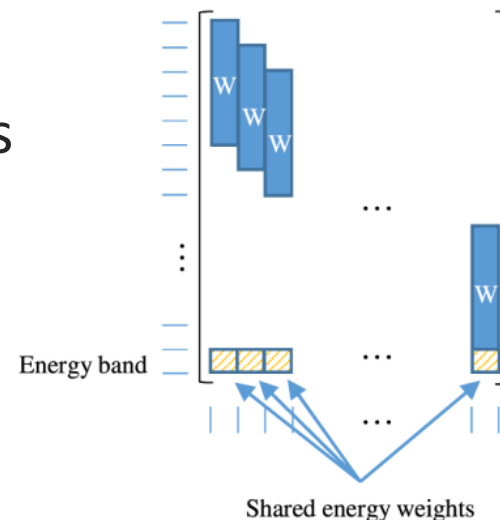
- RBM-based pretraining improves DNN performance especially when the training set is small.
- For convolutional structure, a convolutional RBM (CRBM) has been proposed in

[2008 Bengio et al.] Empirical Evaluation of Convolutional RBMs for V

- Similar to RBMs, the training of the CRBM aims to maximize the likelihood function of the full training data according to an approximate contrastive divergence (CD) algorithm.

Treatment of Energy Features

- In ASR, log-energy is usually calculated per frame and appended to other spectral features.
- In a CNN, it is not suitable to treat energy the same way as other filter bank energies since it is the sum of the energy in all frequency bands and so does not depend on frequency.
- Instead, the log-energy features should be appended as extra inputs to all convolution units
- Other non-localized features can be similarly treated.



The Overall CNN Architecture

- In this paper, we follow the hybrid ANN-HMM framework, where we use a **softmax output layer** on top of the topmost layer of the CNN to compute the posterior probabilities for all HMM states.
- These posteriors are used to estimate the likelihood of all HMM states per frame by dividing by the states' prior probabilities.
- Finally, the likelihoods of all HMM states are sent to a Viterbi decoder to recognize the continuous stream of speech units.

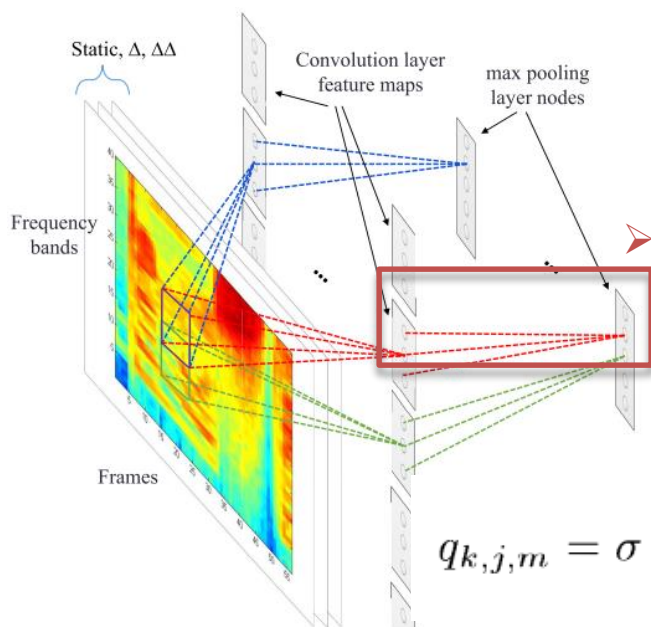
Benefits of CNNs for ASR

- The CNN has 3 key properties:
 - 1) **Locality** : allows more robustness against non-white noise where some bands are cleaner than the others.
 - 2) **Weight sharing** : can also improve model robustness and reduce overfitting as each weight is learned from multiple frequency bands in the input instead of just from one single location.
 - 3) **Pooling** : the same feature values computed at different locations are pooled together and represented by one value.

CNN with limited weight sharing for ASR

Limited Weight Sharing (LWS)

- The properties of the speech signal typically vary over different frequency bands, however.
- Using separate sets of weights for different frequency bands may be more suitable since it allows for detection of distinct feature patterns in different filter bands along the frequency axis.



➤ Only the convolution units that are attached to the same pooling unit share the same convolution weights.

$$q_{k,j,m} = \sigma \left(\sum_i \sum_{n=1}^F o_{i,(k-1) \times s + n + m - 1} \cdot w_{k,i,j,n} + w_{k,0,j} \right)$$

Limited Weight Sharing (LWS)

$$q_{k,j,m} = \sigma \left(\sum_i \sum_{n=1}^F o_{i,(k-1) \times s + n + m - 1} \cdot w_{k,i,j,n} + w_{k,0,j} \right)$$

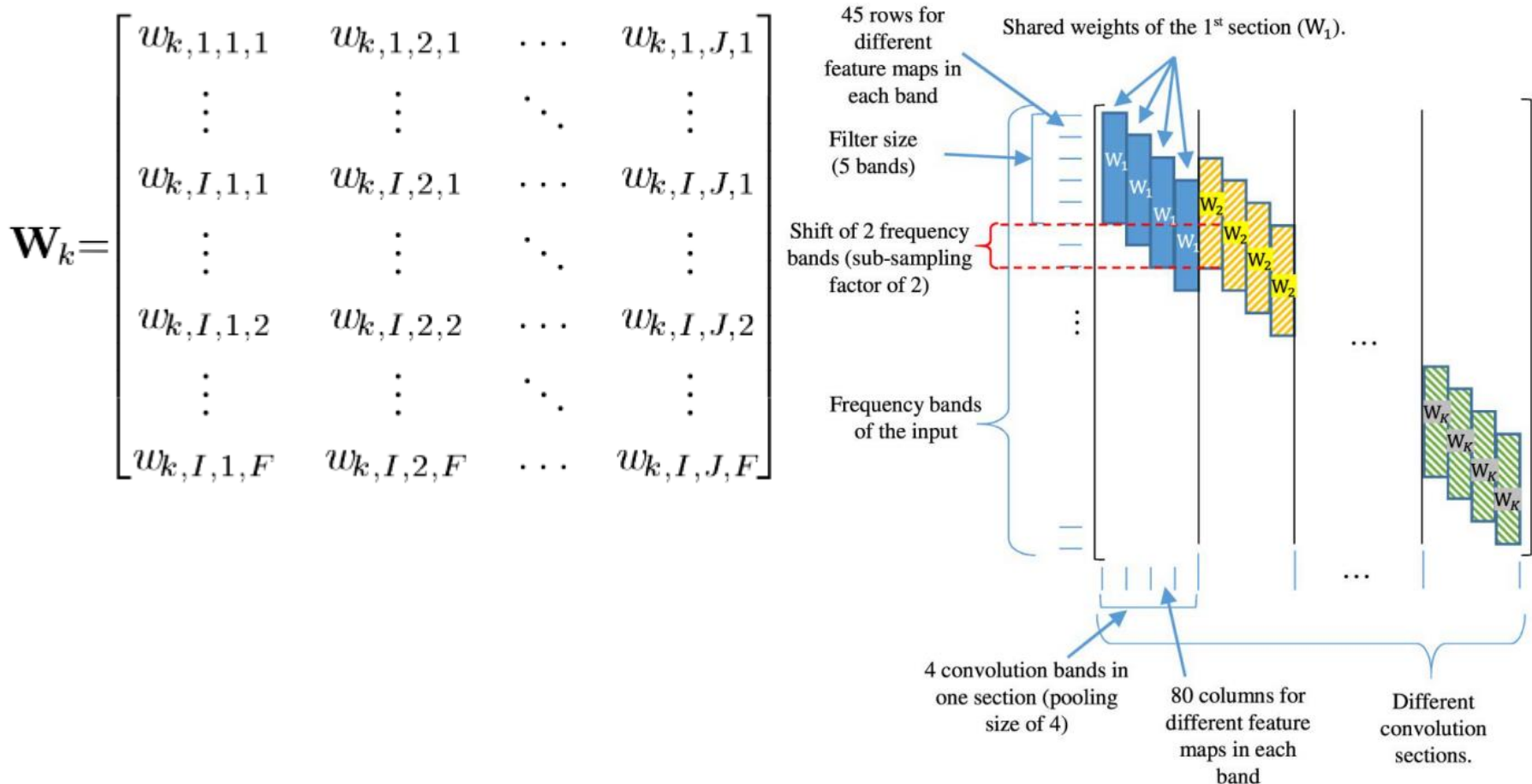
- $w_{k,i,j,n}$ denotes the n -th convolution weight, mapping from the i -th input feature map to the j -th convolution map in the k -th section

$$p_{k,j} = \max_{m=1}^G q_{k,j,m}.$$

- m ranges from 1 up to G (pooling size)

Limited Weight Sharing (LWS)

- The sparse matrix \hat{W} is constructed as below, where each is formed based on local weights, $w_{k,i,j,n}$, as follows:



Limited Weight Sharing (LWS)

- The computed feature maps are organized as a large row vector by concatenating all values in each section as follows:

$$\hat{\mathbf{q}} = [\mathbf{v}_{1,1}] \cdots [\mathbf{v}_{1,G}] \cdots [\mathbf{v}_{K,1}] \cdots [\mathbf{v}_{K,G}]$$

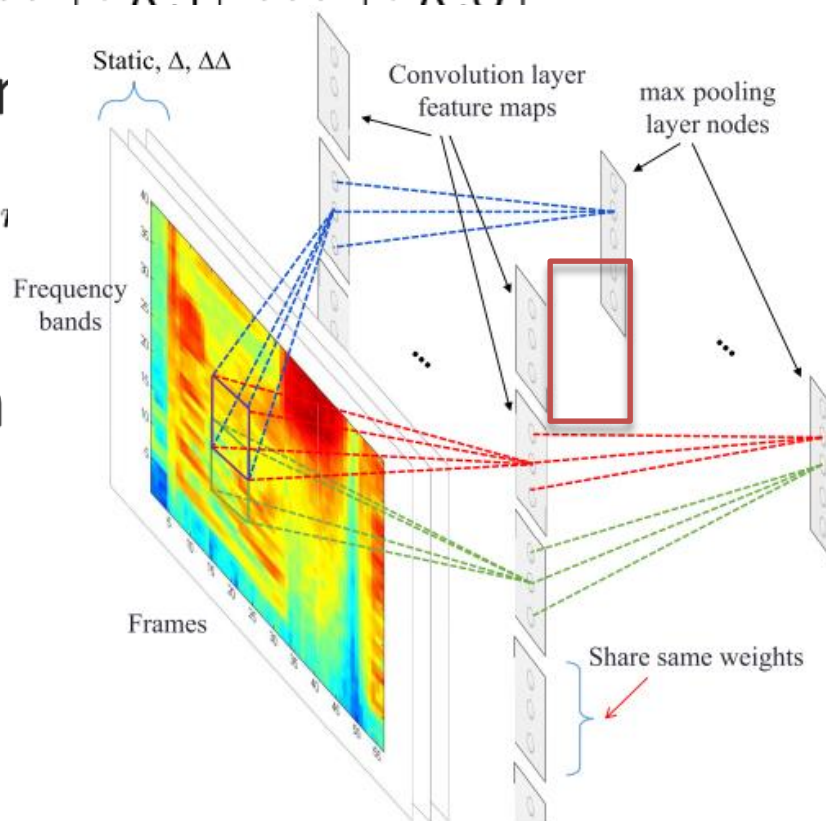
- K is the total number of section

$$\hat{\mathbf{v}}_{k,m} = [q_{k,1,m}, q_{k,2,m}, \dots, q_{k,I,m}]$$

- $\hat{\mathbf{v}}_{k,m}$ is a row vector containing m -th band of the k -th section a convolution ply.

$$\mathbf{e}_{k,i,n}^{\text{low}} = \mathbf{e}_{k,i} \cdot \delta(u_{k,i} - n)$$

$$u_{k,i} = \arg\max_{m=1}^G q_{k,i,m}$$



Pretraining of LWS-CNN

- The conditional probability of the activation for a hidden unit $h_{k,j,m}$, given the CRBM input, is defined as the following softmax function:
$$P(h_{k,j,m} = 1|\mathbf{v}) = \frac{\exp(I(h_{k,j,m}))}{\sum_{n=1}^p \exp(I(h_{k,j,n}))}$$
- $P(h_{k,j,m} = 1|v)$ is the sum of the weighted signal reaching unit $h_{k,j,m}$ from the input layer and is defined as:

$$I(h_{k,j,m}) = \sum_i \sum_{n=1}^f v_{i,(k-1) \times s + n + m - 1} w_{k,i,j,n} + w_{k,i,j,0}$$

- The conditional probability distribution of $v_{i,n}$, can be computed by the following Gaussian distribution:

$$P(v_{i,n}|\mathbf{h}) = \mathcal{N} \left(v_{i,n}; \sum_{j,(k,m) \in \mathbf{C}(i,n)} h_{k,j,m} w_{k,i,j,f(n,k,m)}, \sigma^2 \right)$$

Pretraining of LWS-CNN

- Based on the above two conditional probabilities, **all connection weights** of the above CRBM can be iteratively estimated by using the regular **contrastive divergence (CD) algorithm**.
- The outputs of the pooling ply are used as inputs to continuously pretrain the next layer as done in deep belief network training .

Experiments

Experiments

- The experiments of this section have been conducted on two speech recognition tasks to evaluate the effectiveness of CNNs in ASR:
 - 1) small-scale phone recognition in TIMIT
 - 2) large vocabulary voice search (VS) task

Speech Data and Analysis

- Speech is analyzed using a 25-ms Hamming window with a fixed 10-ms frame rate.
- Speech feature vectors are generated by Fourier-transform-based filter-bank analysis, which includes 40 log energy coefficients distributed on a mel scale, along with their first and second temporal derivatives.
- All speech data were normalized so that each vector dimension has a zero mean and unit variance.

TIMIT Phone Recognition Settings

- **Training set** : 462-speaker and removed all SA records.
- **Tuning** : Separate development set of 50 speakers
- **Test set** : 24-speaker , no overlap with the development set.
- **Target class labels** : 183 (3 states for each HMM of 61 phones)
- 61 phone classes were mapped to a set of 39 classes for final scoring after decoding.
- **Language model** : bigram
- **Training** : mono-phone HMM model

Effects of Varying CNN Parameters :

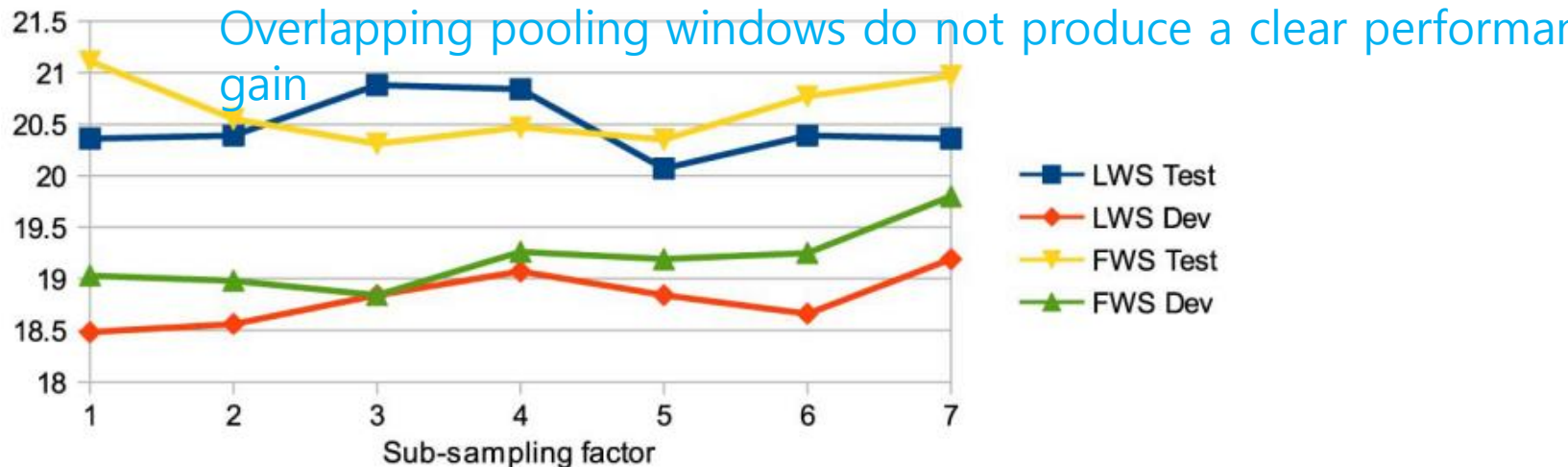
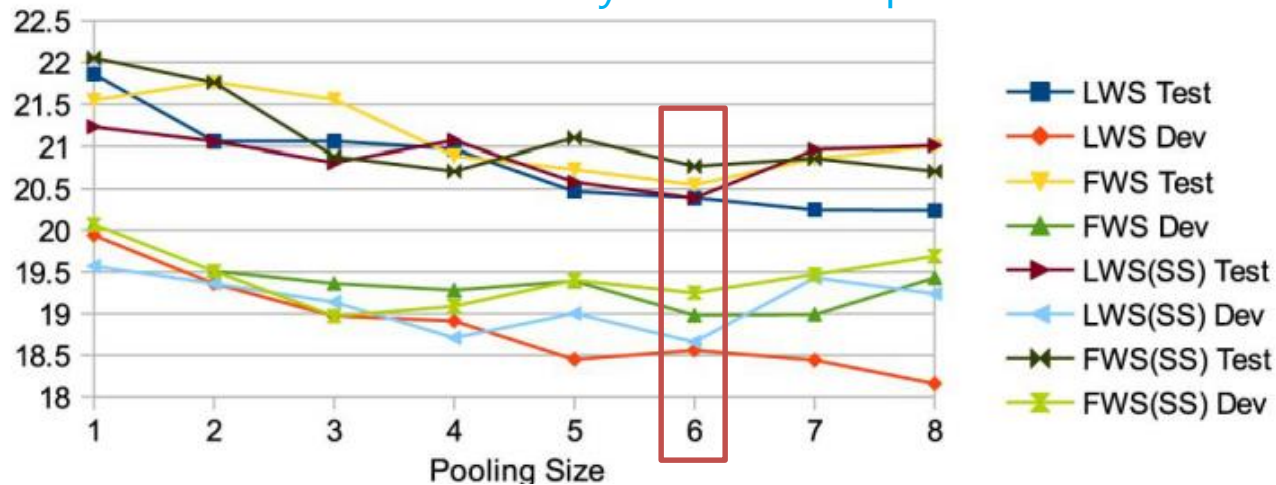
Settings

- In these experiments we used 1 convolution ply, 1 pooling ply and 2 fully connected hidden layers.
- Fully connected layer : 1000 units per layer
- **FWS**
 - ✓ Pooling size : 6
 - ✓ Shift size : 2
 - ✓ Filter size : 8
 - ✓ Feature maps : 150
- **LWS**
 - ✓ Feature maps : 80 per frequency band

Effects of Varying CNN Parameters

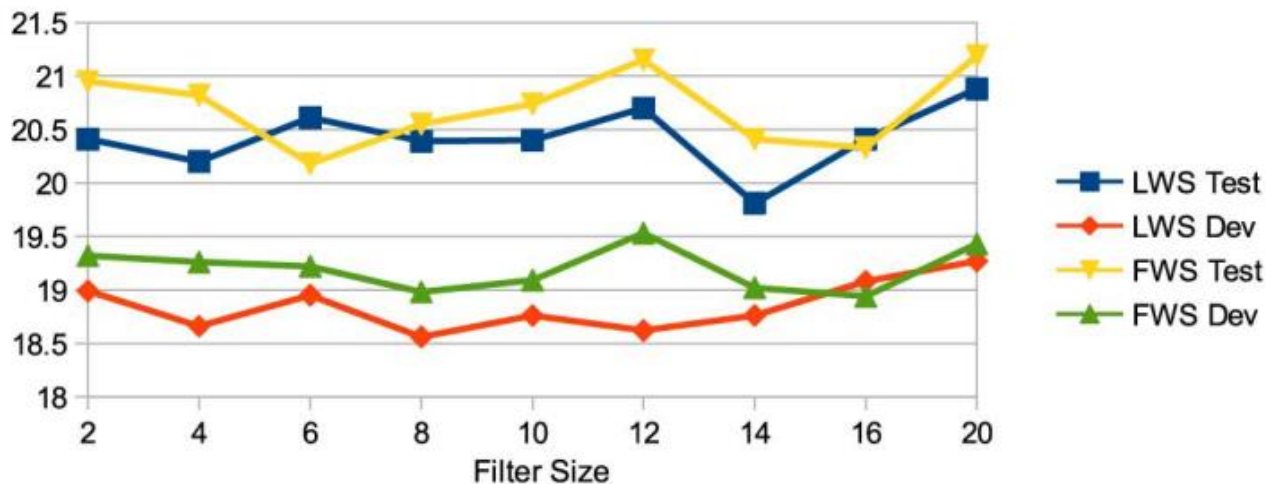
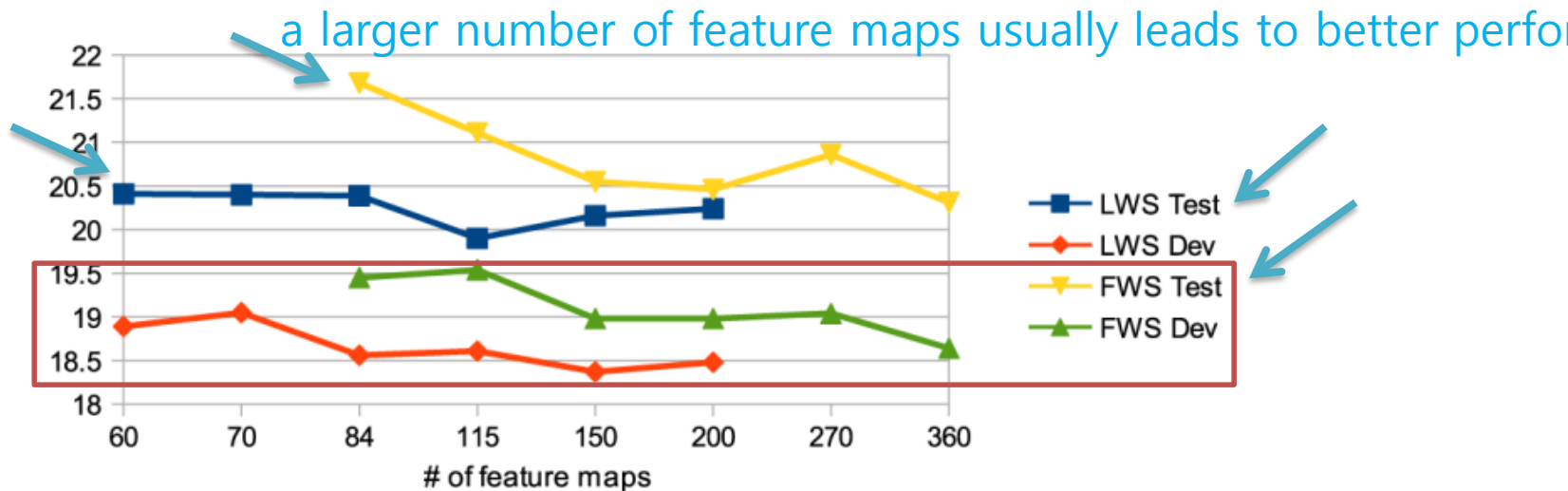
Phone Error Rate (PER in %)

LWS yields better performance with bigger pooling



Effects of Varying CNN Parameters

Phone Error Rate (PER in %)



Effects of Energy Features

Phone Error Rate (PER in %)

	No Energy	Energy
LWS	20.61%	20.39%
FWS	21.19%	20.55%

- The benefit of using energy features, producing a significant accuracy improvement, especially for FWS.
- While the energy features can be easily derived from other MFSC features, adding them as separate inputs to the convolution filters results in more discriminative power as it provides a way to compare the local frequency bands processed by the filter with the overall spectrum.

Effects of Pooling Functions

Phone Error Rate (PER in %)

	Average	Max
Development Set	19.63%	18.56%
Test Set	21.6%	20.39%

- Max-pooling function performs better than the average function with the LWS scheme.
- These results are consistent with what has been observed in image recognition applications.

Overall Performance in TIMIT task

- Baseline : DNN

ID	Network structure	Average PER	min-max PER	# param's	# op's
1	DNN {2000 + 2×1000}	22.02%	21.86-22.11%	6.9M	6.9M
2	DNN {2000 + 4×1000}	21.87%	21.68-21.98%	8.9M	8.9M
3	CNN {LWS(m:150 p:6 s:2 f:8) + 2×1000}	20.17%	19.92-20.41%	5.4M	10.7M
4	CNN {FWS(m:360 p:6 s:2 f:8) + 2×1000}	20.31%	20.16-20.58%	8.5M	13.6M
5	CNN {FWS(m:150 p:4 s:2 f:8) + FWS (m:300 p:2 s:2 f:6) + 2×1000}	20.23%	20.11-20.29%	4.5M	11.7M
6	CNN {FWS(m:150 p:4 s:2 f:8) + LWS (m:150 p:2 s:2 f:6) + 2×1000}	20.36%	19.91-20.61%	4.1M	7.5M

Large Vocabulary Speech Recognition Results

- 18 hours of voice search dataset
- conventional state-tied triphone HMM

Word Error Rate (WER in %)

	No PT	With PT
DNN	37.1%	35.4%
CNN	34.2%	33.4%

pretraining

- The results show that pretraining the CNN can improve its performance, although the effect of pretraining for the CNN is not as strong as that for the DNN.

Conclusion

- Our hybrid CNN-HMM approach delegates temporal variability to the HMM, while **convolving along the frequency axis** creates a degree of invariance to small frequency shifts, which normally occur in actual speech signals due to speaker differences.
- We have proposed a new, **limited weight sharing** scheme that can handle speech features in a better way than the full weight sharing
- Limited weight sharing leads to a much smaller number of units in the pooling ply.

- The use of energy information is very beneficial for the CNN in terms of recognition accuracy.
- The ASR performance was found to be sensitive to the pooling size, but insensitive to the overlap between pooling units.
- Pretraining of CNNs was found to yield [better performance in the large-vocabulary voice search experiment](#), but not in the phone recognition experiment.