Deep Learning for Automatic Speech Recognition – Part II

Xiaodong Cui

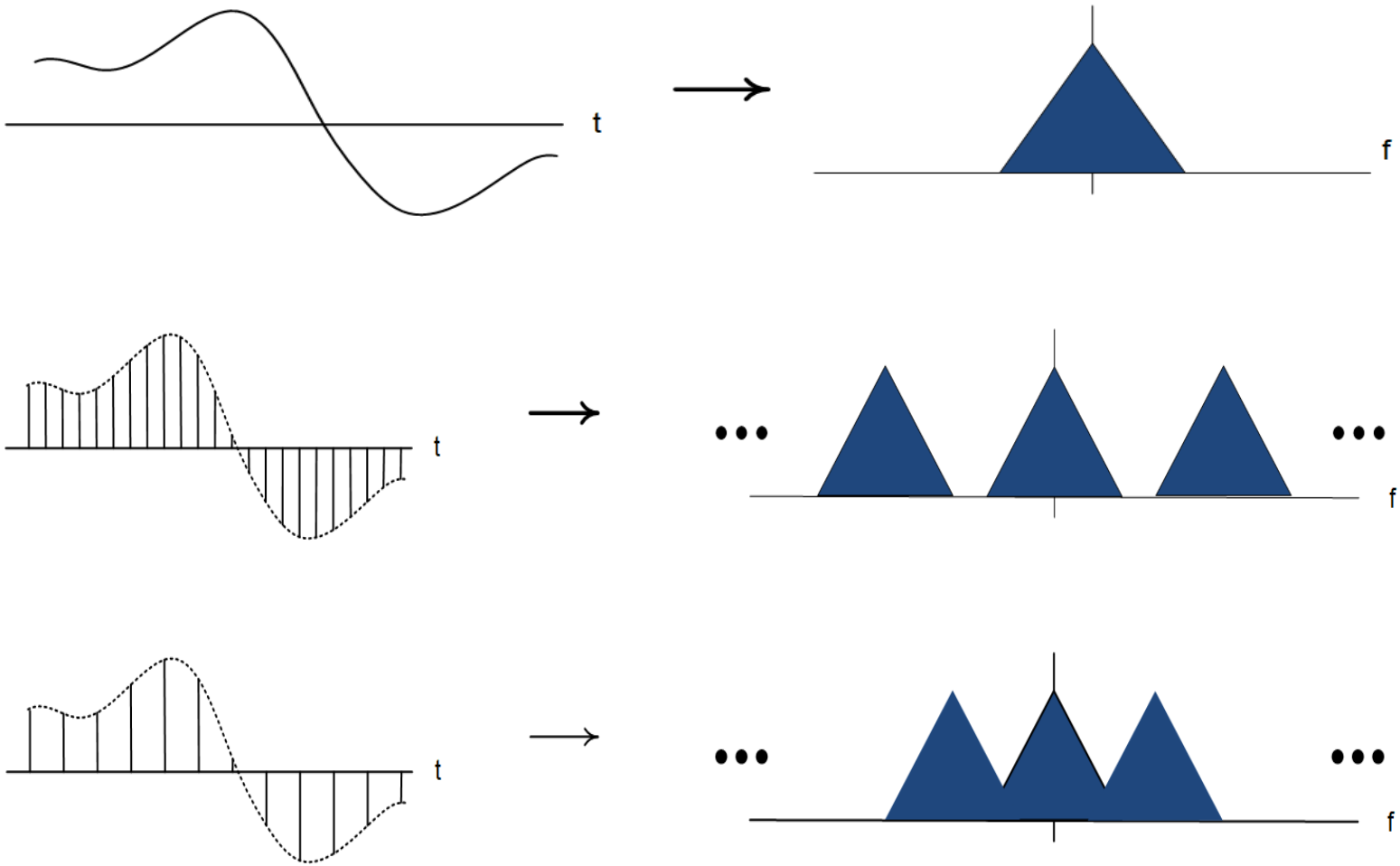
IBM T. J. Watson Research Center
Yorktown Heights, NY 10598

Fall, 2018

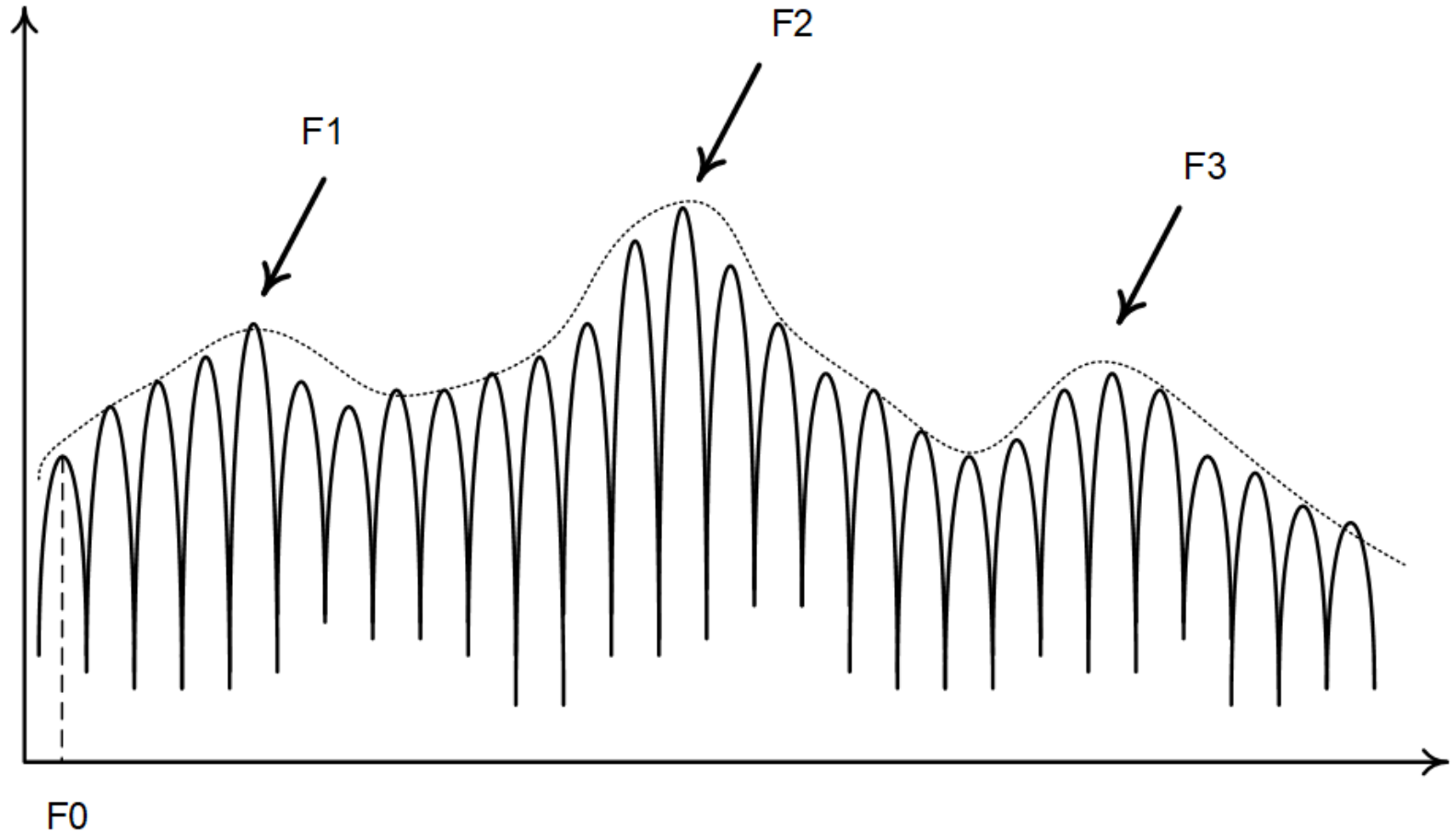
Outline

- A brief revisit of sampling, pitch/formant and MFCC
- DNN-HMM (hybrid) acoustic modeling
- Advanced acoustic modeling
  - Convolutional neural networks (CNNs)
  - Recurrent neural networks (RNNs) and long short-term memory (LSTM) networks
  - Convolutional long short-term memory (CLSTM) networks

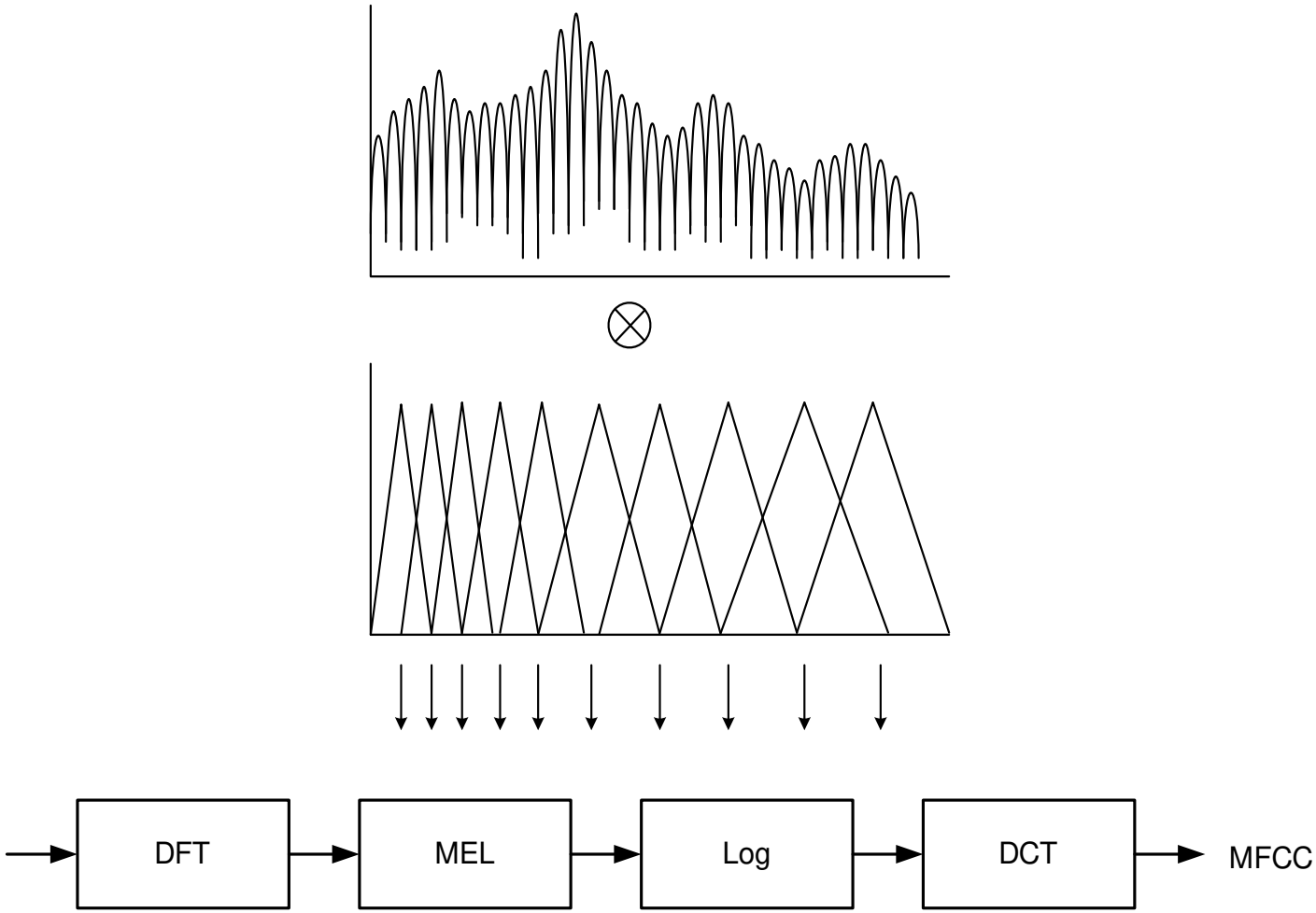
Nyquist Sampling



Pitch and Formants

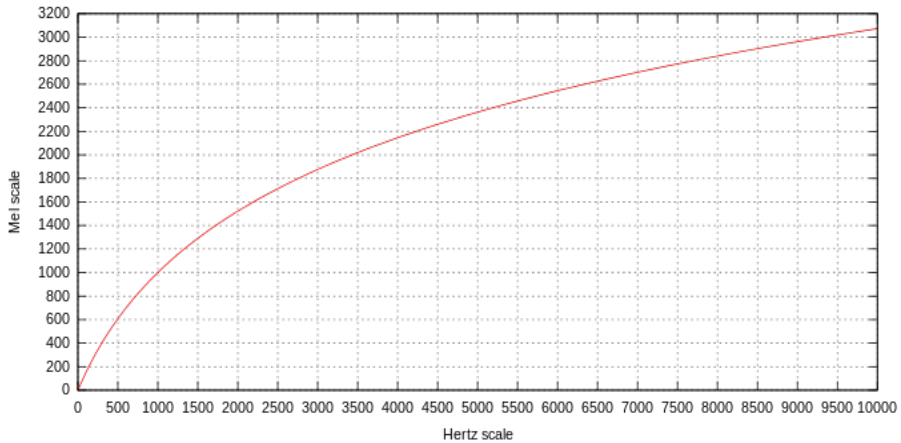


Mel-Frequency Filter Bank and MFCC



Mel-Frequency Filter Bank and MFCC

$$m = 2595 \log_{10}\left(1 + \frac{f}{100}\right)$$



- multi-resolution on frequency bins
- use DCT to replace IDFT for better dimension decorrelation and energy clustering
- typical order around 13 for speech recognition
- increased order for speaker recognition/verification (typically 19-22)

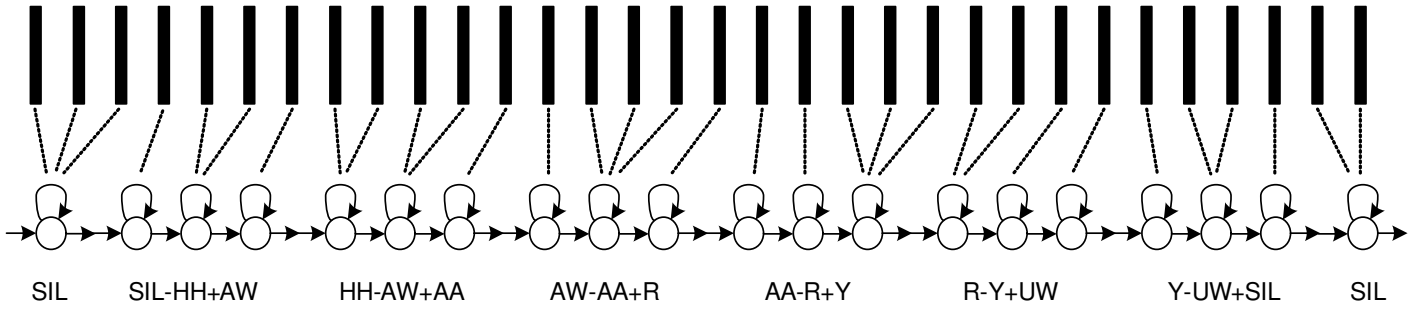
Input Speech Features for DNNs

- Commonly-used hand-crafted features
  - MFCCs
  - Mel-frequency filter bank (FBank)
  - Speaker adapted features (FMLLR)
  - Appended speaker embedding vectors (i-vectors)
- Let DNNs learn the features
  - Power spectra
  - Raw audio signals

Two Streams of DNN Acoustic Models

- Hybrid DNNs
  - Commonly referred to as DNN-HMM or CD-DNN-HMM
  - Use GMM-HMM alignments as labels
  - Use dictionary and language model for decoding.
- End-to-end (E2E) DNNs (will be addressed in next lecture)
  - Directly deal with sequence-to-sequence mapping problem with unequal consequence lengths
  - Do not need alignments, dictionary and language model **in principle**.
  - Two E2E architectures:
    - Connectionist Temporal Classification (CTC)
    - Encoder-Decoder Attention models

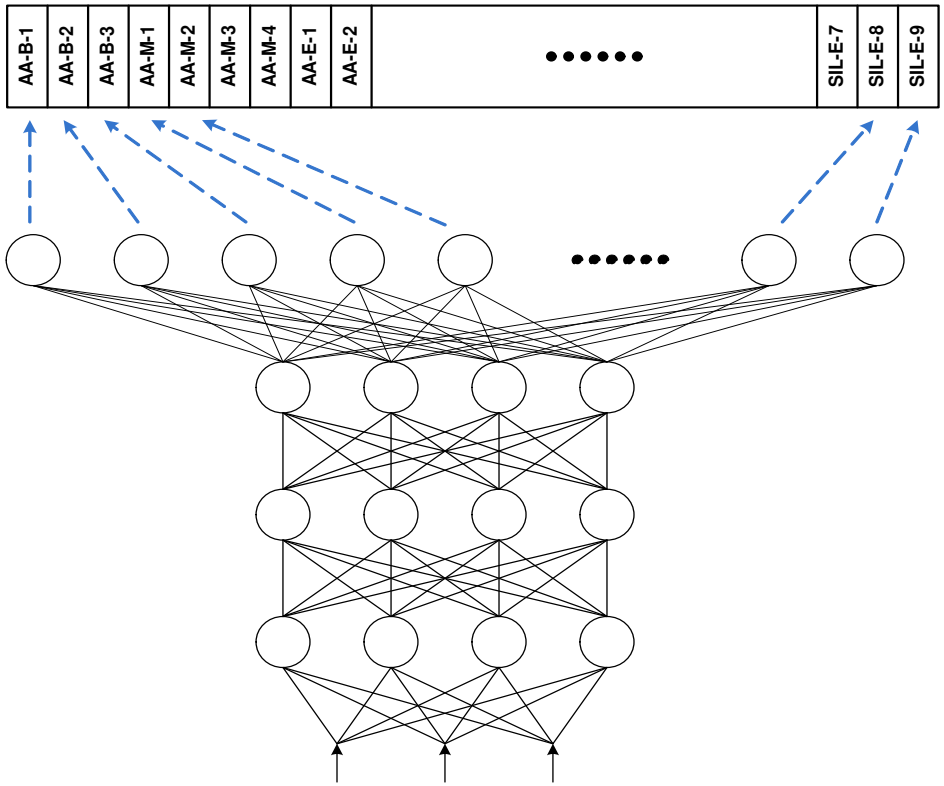
GMM-HMM: Forced Alignment



"How are you"

- Given the text label, how to find the best underlying state sequence?
  - same as decoding except the label is known
  - Viterbi algorithm (dynamic programming)
- Often referred to as Viterbi alignments in speech community
- Widely used in deep learning ASR to generate the target labels for the training data

Context-Dependent DNNs

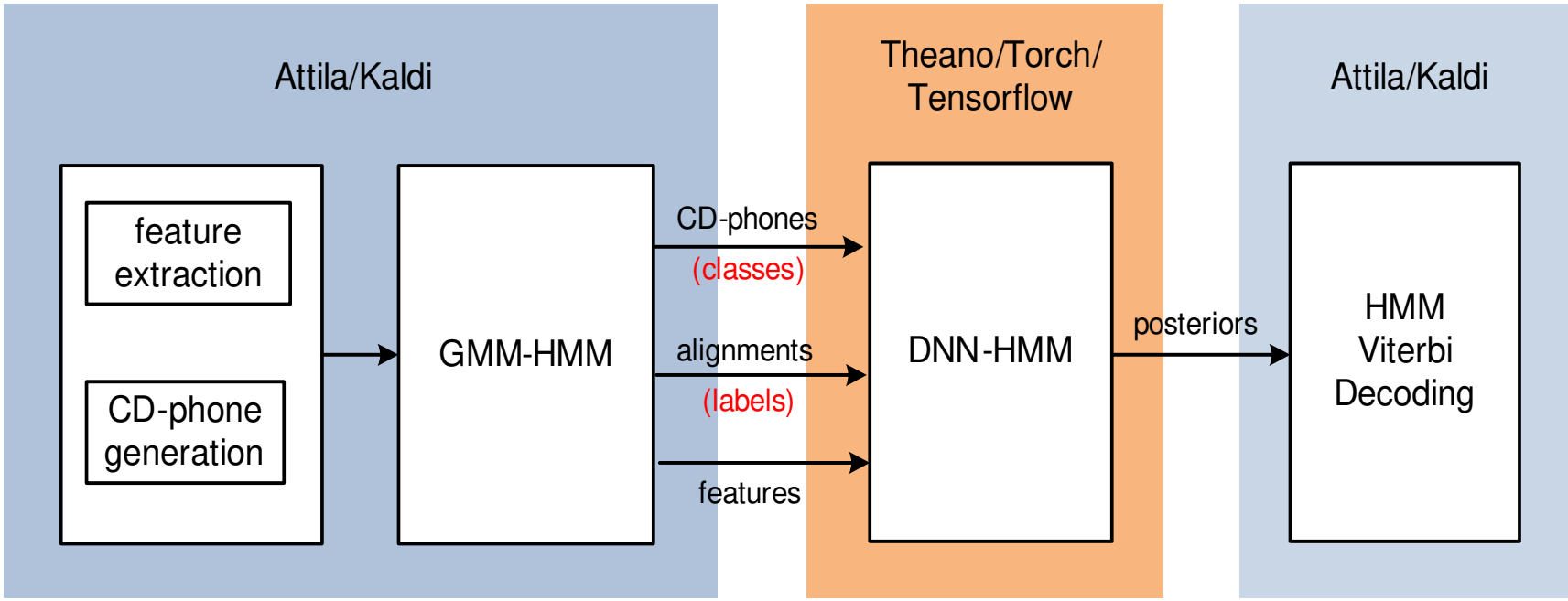


$$P(o|s) = \frac{p(s|o)p(o)}{p(s)} \propto \frac{p(s|o)}{p(s)}$$

DNN-HMMs: An Typical Training Recipe

- Preparation
  - 40-dim FBank features with ± 4 adjacent frames (input dim $=$ 40x9)
  - use an existing model to generate alignments which are then converted to 1-hot targets from each frame
  - create training and validation data sets
  - estimate priors $p(s)$ of CD states
- Training
  - set DNN configuration (multiple hidden layers, softmax output layers and cross-entropy loss function)
  - initialization
  - optimization based on back-prop using SGD on the training set while monitoring loss on the validation set
- Test
  - push input features from test utterances through the DNN acoustic model to get their posteriors
  - convert posteriors to likelihoods
  - Viterbi decoding on the decoding network
  - measure the word error rate

Training A Hybrid DNN-HMM System

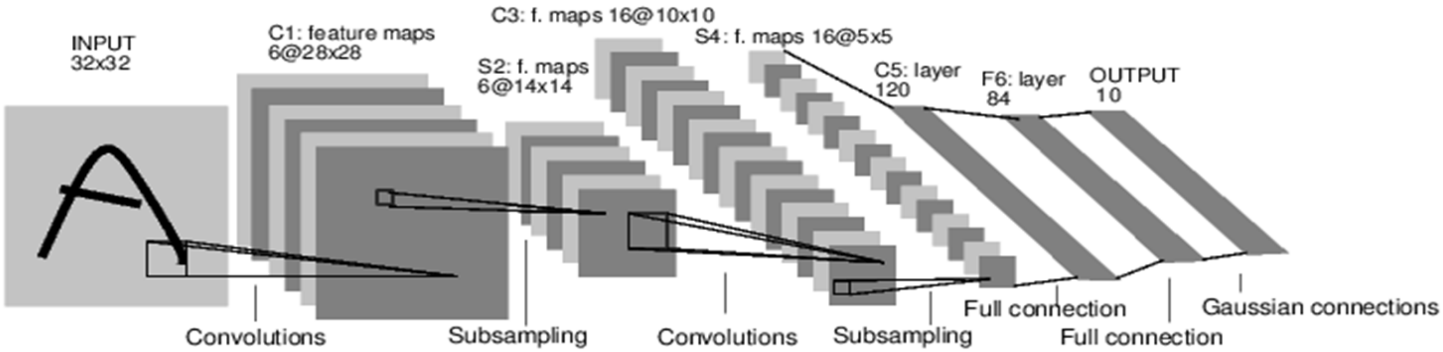


Convolutional Neural Networks (CNNs)

- Bio-inspired feed-forward neural networks*
  - individual cortical neurons in visual cortex respond to stimuli in a restricted region of space known as the receptive field.
  - receptive fields of different neurons partially overlap such that they tile the visual field.
  - response of an individual neuron to stimuli within its receptive field can be approximated mathematically by a convolution operation.
- LeNet-5
  - Y. LeCun, L. Bottou, Y. Bengio, P. Haffner (1998). "Gradient-based learning applied to document recognition," Proc. of the IEEE.
  - A 7-layer CNN that outperformed other techniques on a standard handwritten digit recognition task.
- Advantages:
  - Local (sparse) connectivity
  - Weight sharing
  - translation-invariant

*adapted from wikipedia

Convolutional Neural Networks (CNNs)



- Convolutional layers with nonlinearity

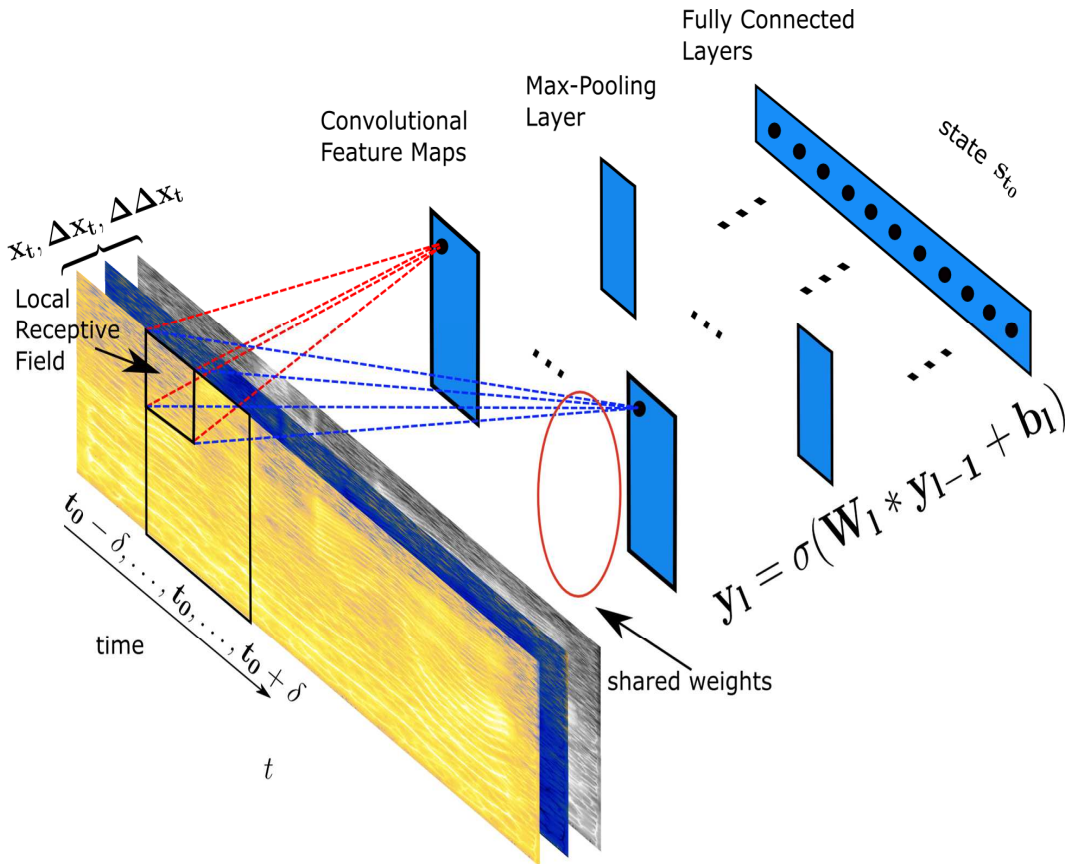
$$z_{ij}^k = \sigma((W^k * x)_{ij} + b^k)$$

$$W(m,n) * x(m,n) = \sum_{u=-\infty}^{+\infty} \sum_{v=-\infty}^{+\infty} x(u,v) W(u-m, v-n)$$

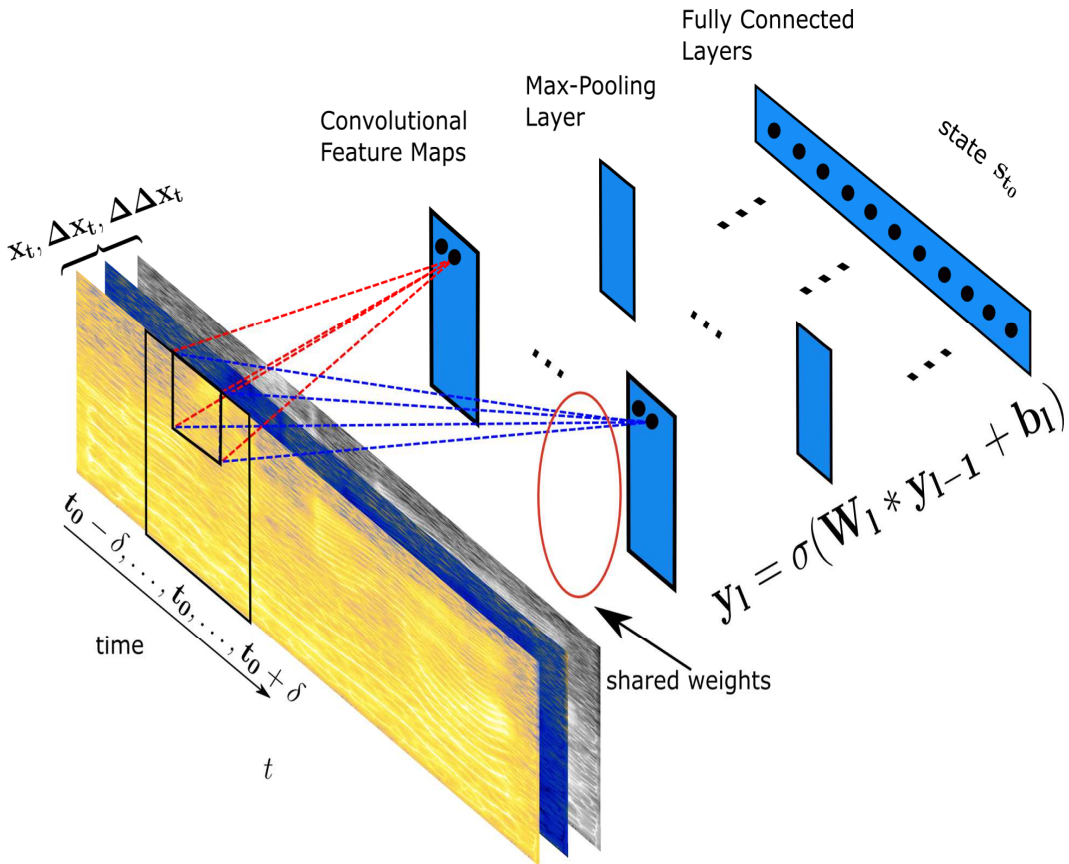
$$W(m,n) \neq 0 \text{ only for } 0 \leq m \leq M,\ 0 \leq n \leq N$$

- Pooling layers
  - subsampling
  - typically max-pooling or average-pooling
- Fully connected layers

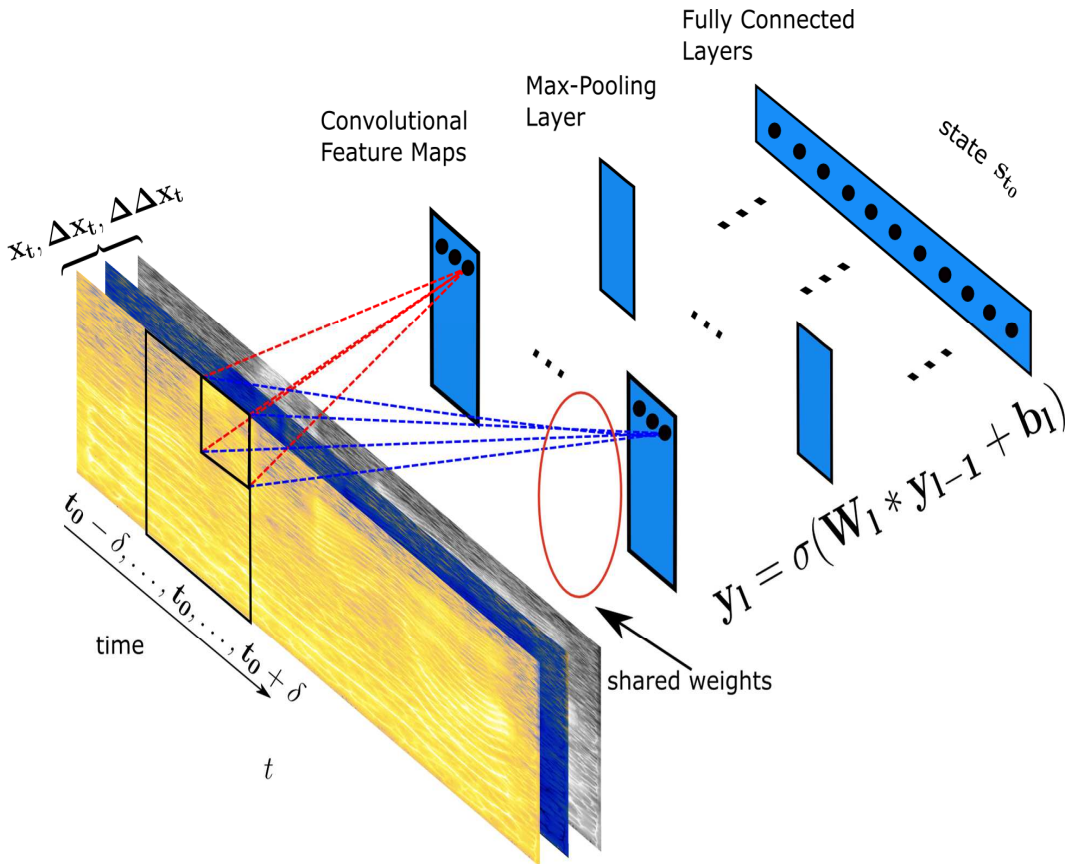
Convolutional Neural Networks (CNNs)



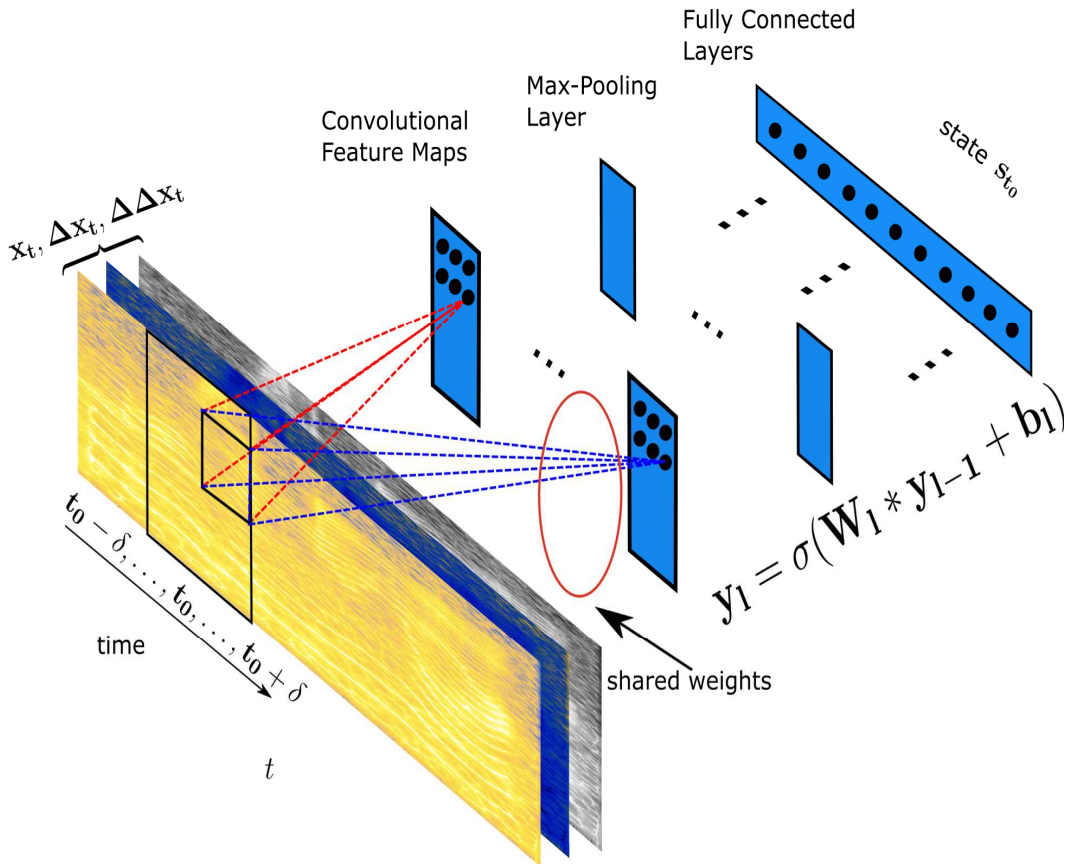
Convolutional Neural Networks (CNNs)



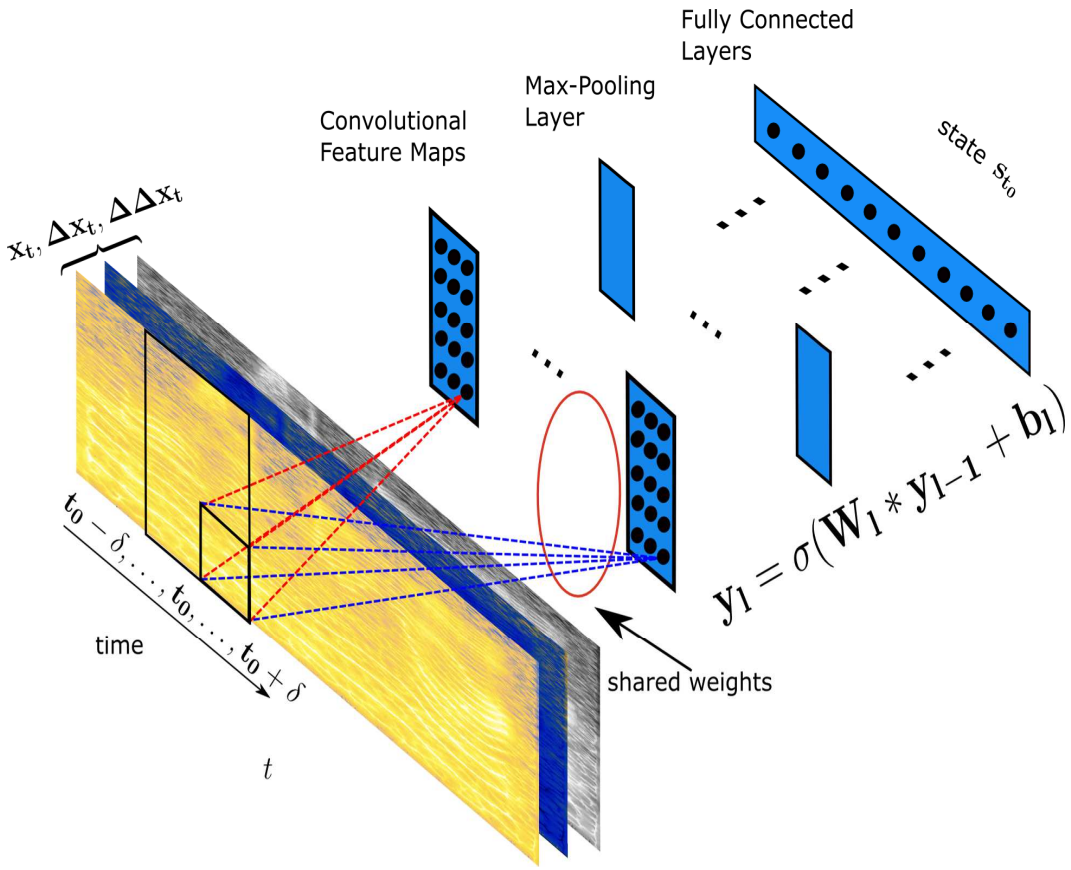
Convolutional Neural Networks (CNNs)



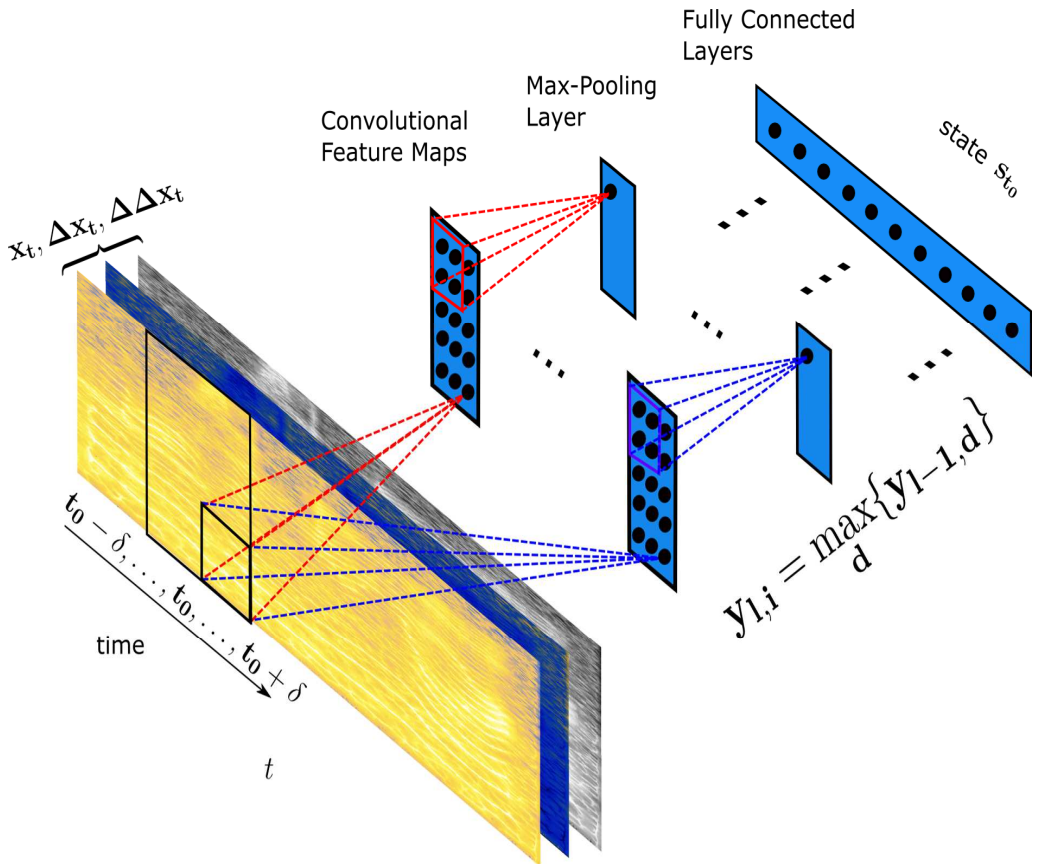
Convolutional Neural Networks (CNNs)



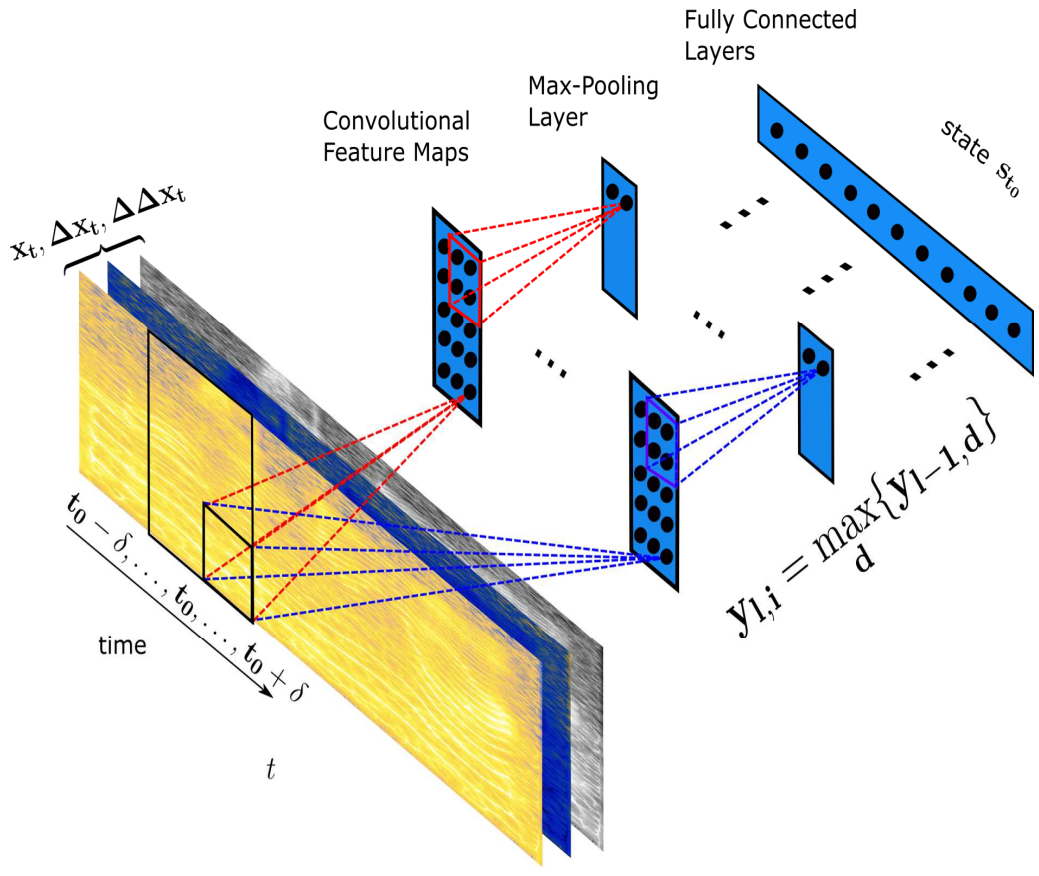
Convolutional Neural Networks (CNNs)



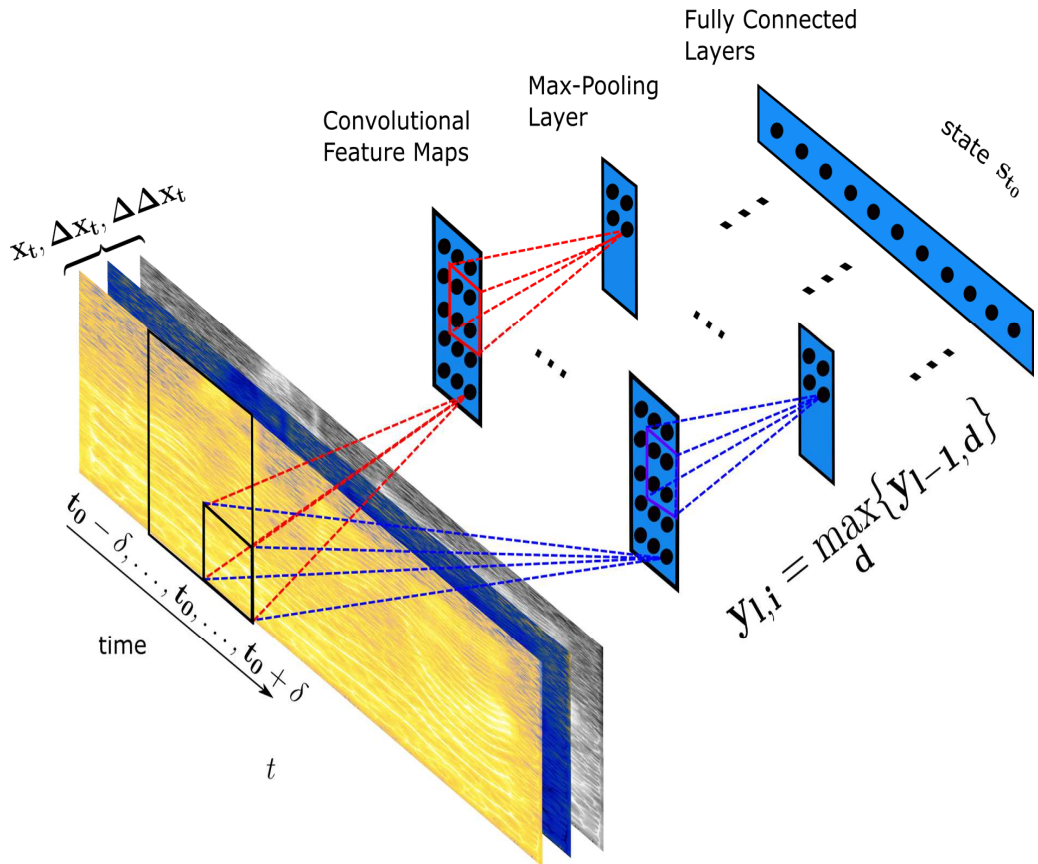
Convolutional Neural Networks (CNNs)



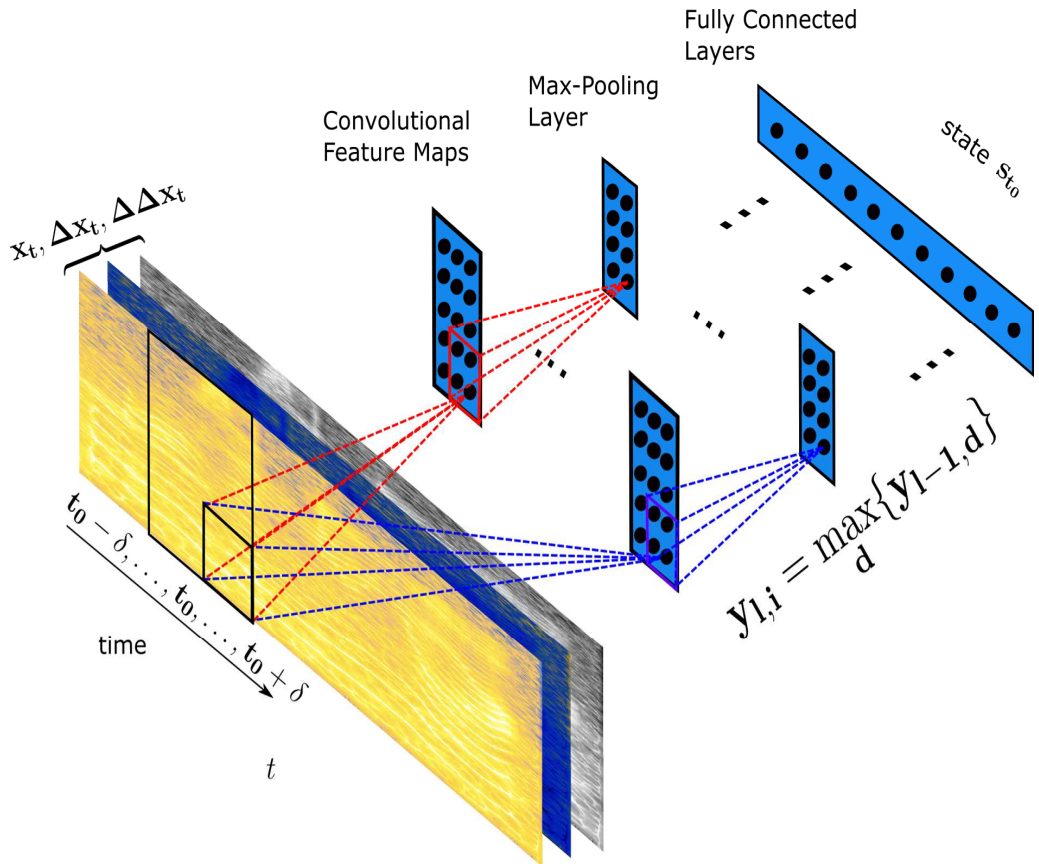
Convolutional Neural Networks (CNNs)



Convolutional Neural Networks (CNNs)



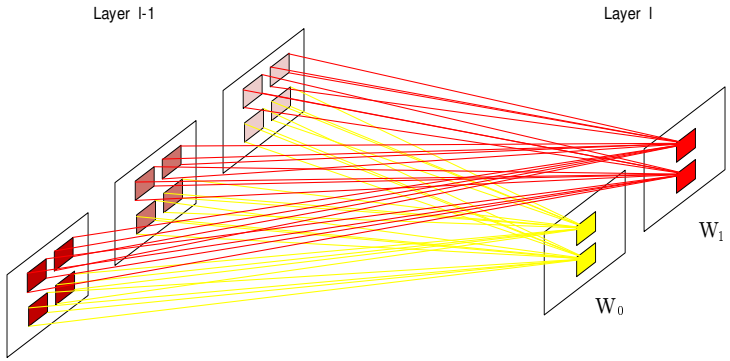
Convolutional Neural Networks (CNNs)



A Brief Summary of Some CNN Terminology

- feature map
- local receptive field (filter or kernel)
- padding
- stride
- convolution
- pooling
  - max-pooling
  - average-pooling
  - pooling along different axes
  - resulting different resolutions

Tensor Representation



- weights W_{mn}^{pq} connecting each unit of the p-th feature map at layer $l-1$ and the q-th feature map of layer l with the local receptive filter $w(m,n)$.
- number of parameters in the weights that connect the two convolutional layers is (ignore biases here)

$$|W_{mn}^{pq}| = M \times N \times P \times Q$$

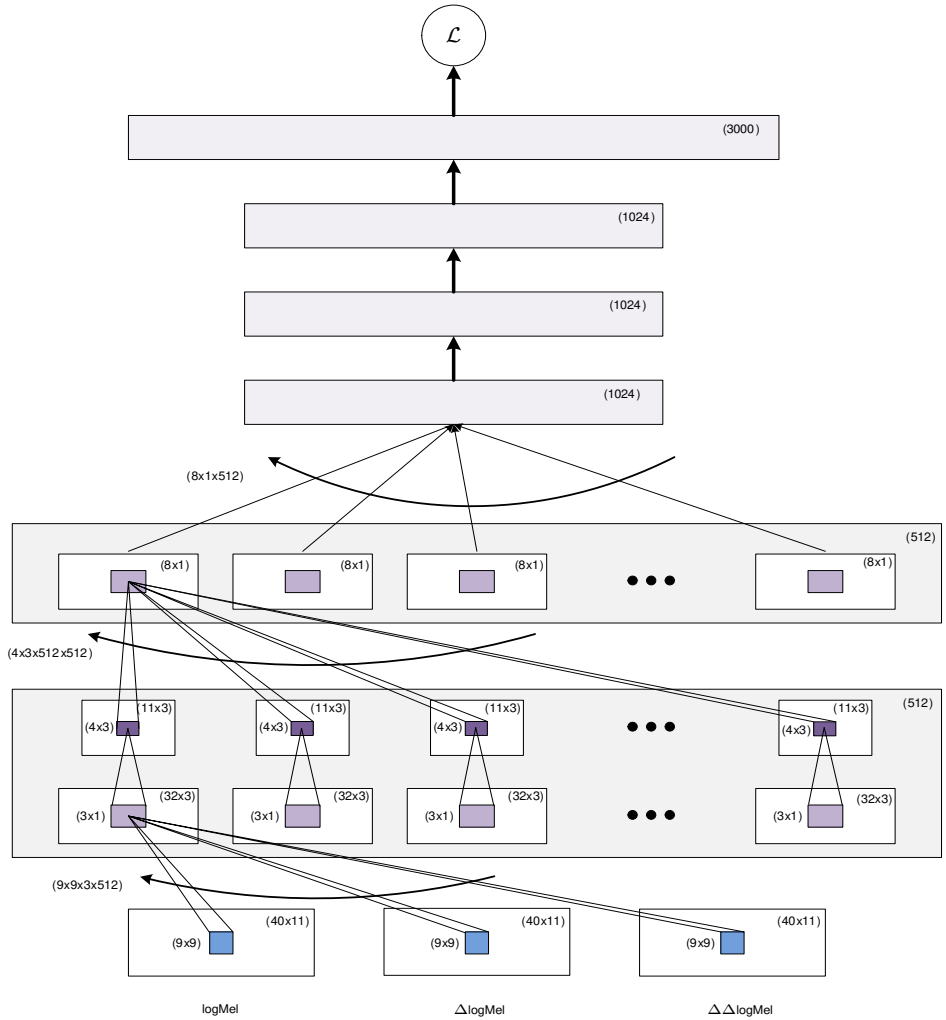
where M and N are the dimensions of the local receptive filters and P and Q are the numbers of feature maps in layer $l-1$ and layer l.

- relation of input and output dimensionality

$$O = \frac{W - K + 2P}{S} + 1$$

where W is the input dimensionality, O the output dimensionality, K the filter size, P the padding and S the stride.

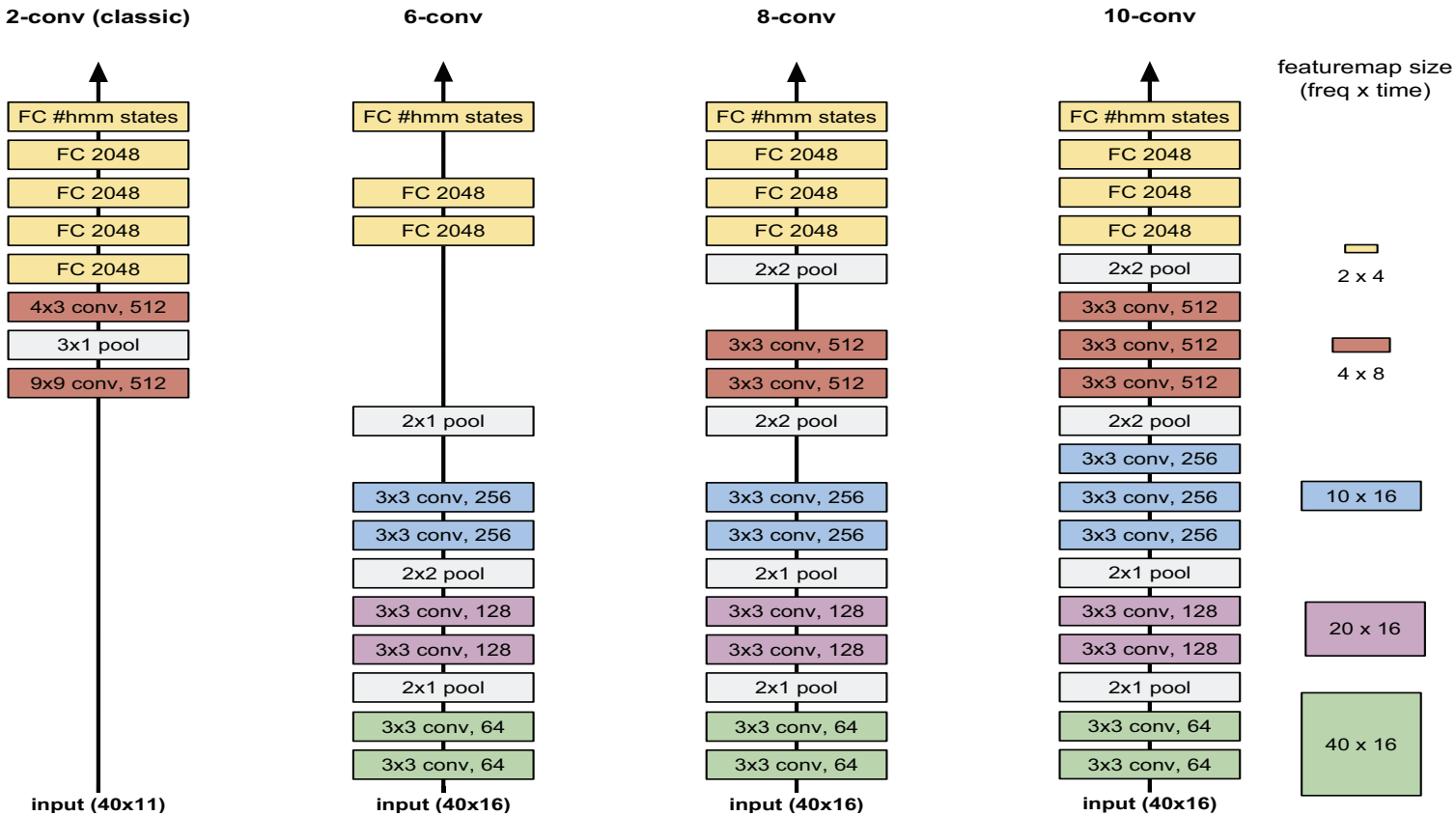
An Example of CNN Acoustic modeling



Notable Development In CNNs

- AlexNet
  - Alex Krizhevsky, Ilya Sutskever and Geoffrey E. Hinton, "ImageNet Classification with Deep Convolutional Networks," NIPS 2012.
  - 5 conv layers, 3 fully-connected layers, ReLU, max-pooling, dropout, data augmentation, 2-GPU implementation
- ZFNet
  - Matthew D. Zeiler and Rob Fergus, "Visualizing and understanding convolutional networks." ECCV, 2014.
  - similar architecture to AlexNet, reduced window size and stride step in first conv layer.
- VGGNet
  - Karen Simonyan and Andrew Zisserman, "Very deep convolutional neural networks for large-scale image recognition," ICLR 2015.
  - up to 19 conv layers – small 3x3 filters with stride 1, padding, 2x2 max-pooling layers with stride 2 every 2,3 or 4 conv layers
- ResNet
  - Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, "Deep Residual Learning for Image Recognition," CVPR 2016.
  - 152 layers with 3x3 filters
  - introduced residual learning by a direct by-pass identity link
- DenseNet
  - Gao Huang, Zhuang Liu, Laurens van der Maaten, Kilian Q. Weinberger, "Densely connected convolutional networks," CVPR 2017.
  - do not throw away anything you learn along the way

VGG Acoustic Modeling



T. Sercu, C. Puhrsch, B. Kingsbury and Y. LeCun, "Very Deep Multilingual Convolutional Neural Networks for LVCSR," ICASSP 2016.

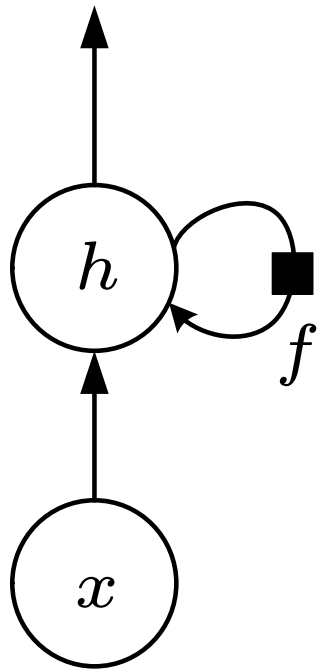
Recurrent Neural Networks (RNNs)

- Sequential data is ubiquitous – speech, video, text, stock index $\cdots$
- Sequence modeling is crucial – how to model the history?

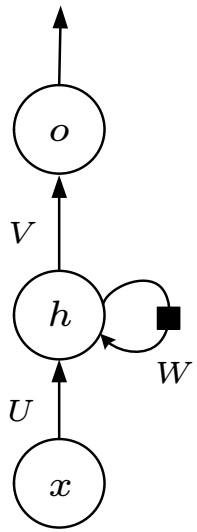
$$\boldsymbol{h}_t = g_t(\boldsymbol{x}_t, \boldsymbol{x}_{t-1}, \boldsymbol{x}_{t-2}, \cdots, \boldsymbol{x}_1)$$

- RNNs model sequences with shared parameters (functions)

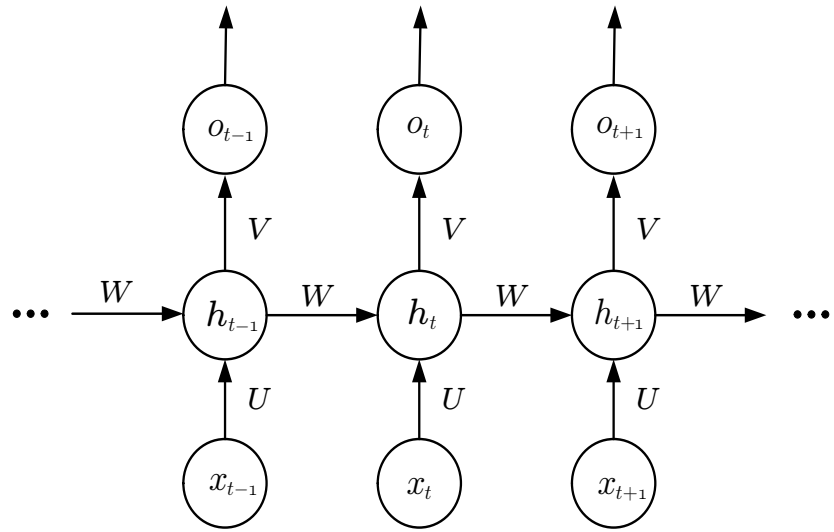
$$\boldsymbol{h}_t = f(\boldsymbol{h}_{t-1}, \boldsymbol{x}_t; \boldsymbol{\theta})$$



Compact and Unfolded Representations of RNNs



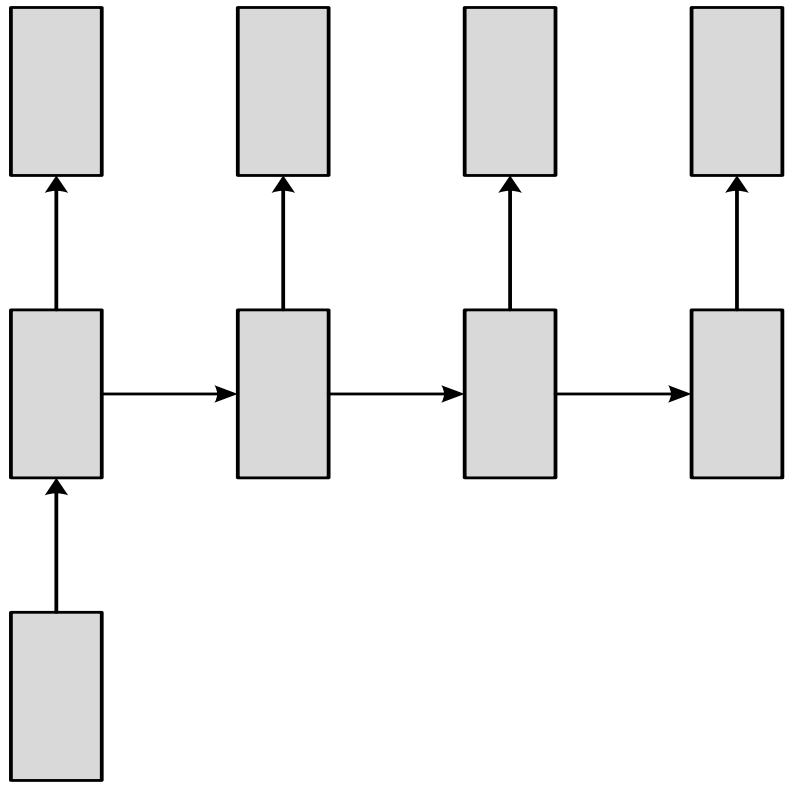
compact



unfolded

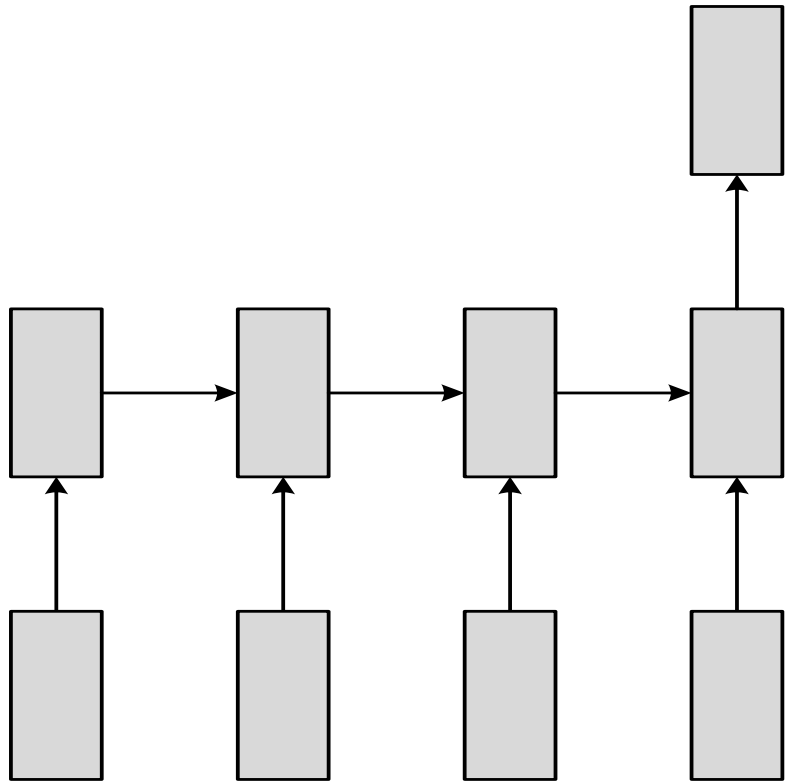
$$\boldsymbol{a}_t = \boldsymbol{b} + \mathbf{W}\boldsymbol{h}_{t-1} + \mathbf{U}\boldsymbol{x}_t$$
$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t)$$
$$\boldsymbol{o}_t = \boldsymbol{c} + \mathbf{V}\boldsymbol{h}_t$$
$$\hat{\boldsymbol{y}}_t = \text{softmax}(\boldsymbol{o}_t)$$

Some RNN Configurations



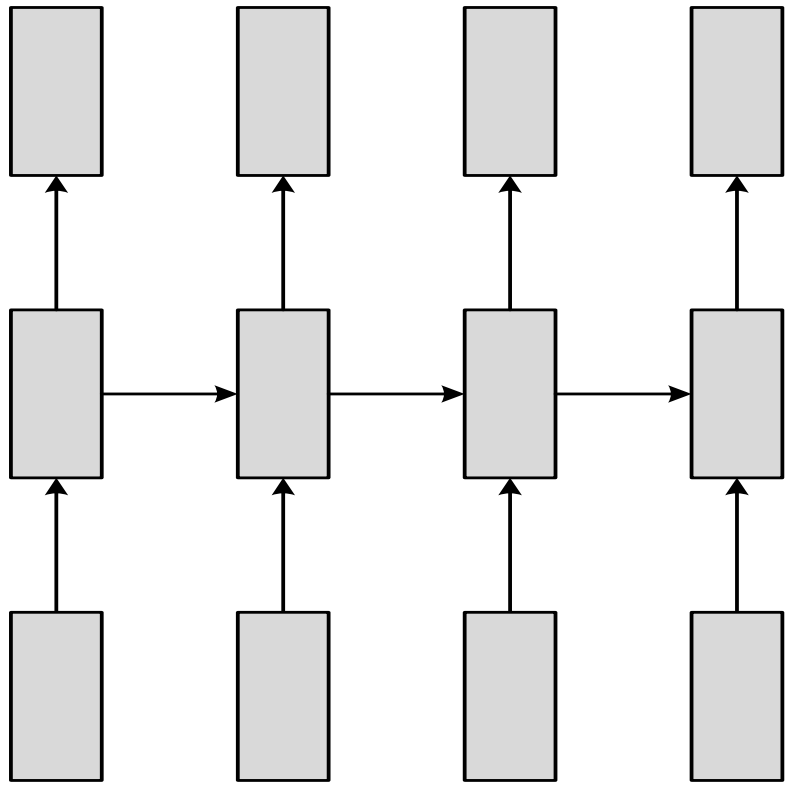
one-to-many mapping — image captioning.

Some RNN Configurations



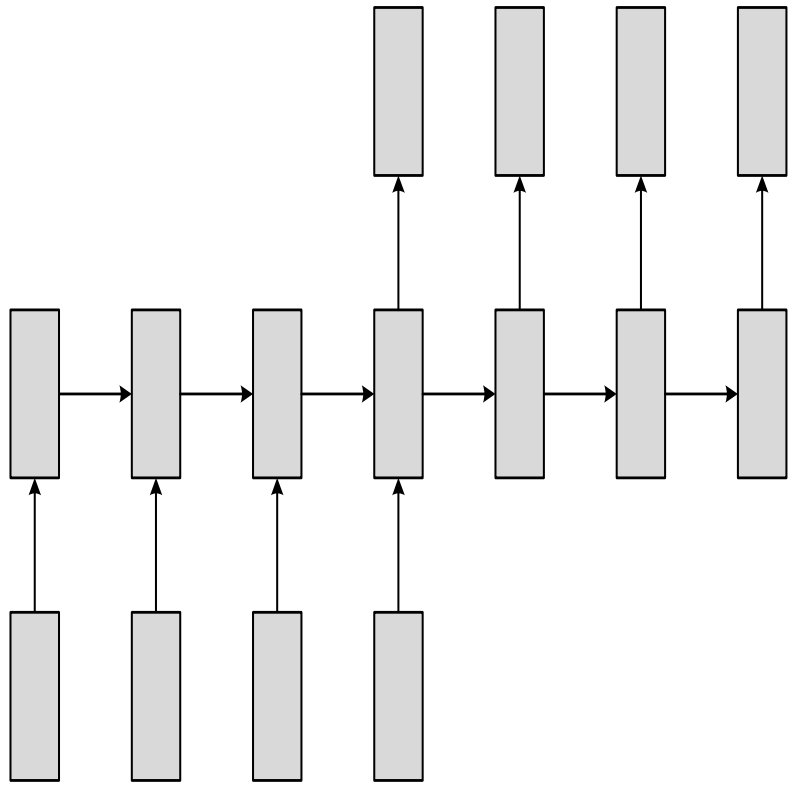
many-to-one mapping — sentiment analysis.

Some RNN Configurations



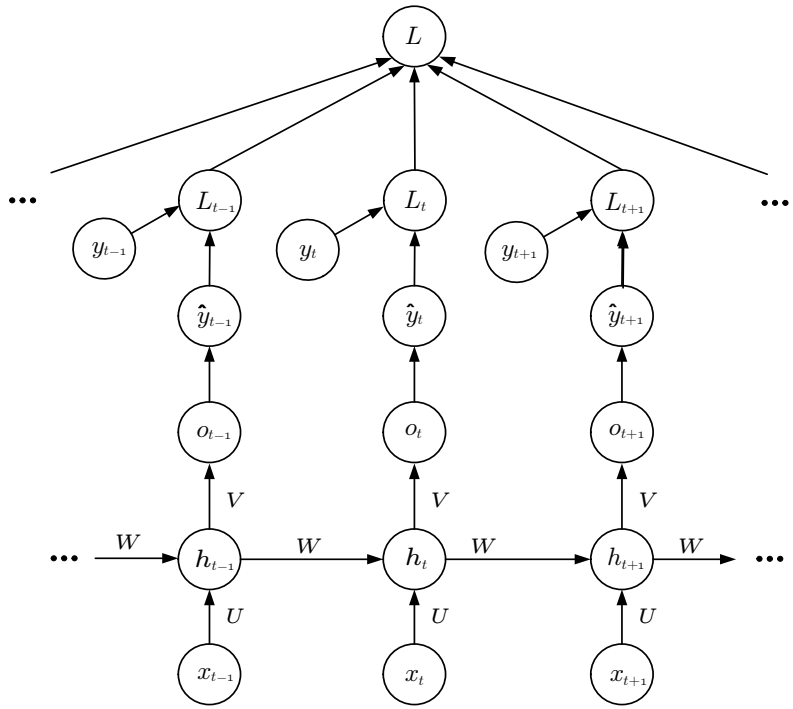
synchronized many-to-many mapping — speech recognition, video labeling.

Some RNN Configurations



asynchronized many-to-many mapping — speech recognition, machine translation.

Back-Propagation Through Time (BPTT)



Forward Propagation:

$$\boldsymbol{a}_t = \boldsymbol{b} + \mathbf{W}\boldsymbol{h}_{t-1} + \mathbf{U}\boldsymbol{x}_t$$

$$\boldsymbol{h}_t = \tanh(\boldsymbol{a}_t)$$

$$\boldsymbol{o}_t = \boldsymbol{c} + \mathbf{V}\boldsymbol{h}_t$$

$$\hat{\boldsymbol{y}}_t = \text{softmax}_K(\boldsymbol{o}_t)$$

$$\mathcal{L}_t = -\sum_{k=1}^{K} y_{t,k} \log \hat{y}_{t,k}$$

$$\mathcal{L} = \sum_{t=1}^{T} \mathcal{L}_t$$

Back-Propagation Through Time (BPTT)

Parameters to be optimized: $\mathbf{W}$, $\mathbf{U}$, $\mathbf{V}$, $\mathbf{b}$ and $\mathbf{c}$

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \mathbf{W}} & = \sum_t \sum_i \frac{\partial \boldsymbol{h}_t^i}{\partial \mathbf{W}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t^i} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{U}} & = \sum_t \sum_i \frac{\partial \boldsymbol{h}_t^i}{\partial \mathbf{U}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t^i} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{b}} & = \sum_t \frac{\partial \boldsymbol{h}_t}{\partial \mathbf{b}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{V}} & = \sum_t \sum_i \frac{\partial \boldsymbol{h}_t^i}{\partial \mathbf{V}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t^i} \\ \frac{\partial \mathcal{L}}{\partial \mathbf{c}} & = \sum_t \frac{\partial \boldsymbol{o}_t}{\partial \mathbf{c}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} \end{cases}$$

which involves gradients of two internal nodes $\boldsymbol{o}_t$ and $\boldsymbol{h}_t$.

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t}, \quad \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$$

Back-Propagation Through Time (BPTT)

$$\frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} = \frac{\partial \hat{\boldsymbol{y}}_t}{\partial \boldsymbol{o}_t} \frac{\partial \mathcal{L}_t}{\partial \hat{\boldsymbol{y}}_t} \frac{\partial \mathcal{L}}{\partial \mathcal{L}_t} = \hat{\boldsymbol{y}}_t - \boldsymbol{y}_t$$

evaluate the gradient of $\boldsymbol{h}_t$ backward starting from the last time step τ:

$$\begin{aligned}
\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_\tau} &= \frac{\partial \boldsymbol{o}_\tau}{\partial \boldsymbol{h}_\tau} \frac{\partial \mathcal{L}_\tau}{\partial \boldsymbol{o}_\tau} \frac{\partial \mathcal{L}}{\partial \mathcal{L}_\tau} = \mathbf{V}^\mathsf{T}(\hat{\boldsymbol{y}}_\tau - \boldsymbol{y}_\tau) \\
\frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} &= \frac{\partial \boldsymbol{o}_t}{\partial \boldsymbol{h}_t} \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} + \frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{h}_t} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}} \\
&= \mathbf{V}^\mathsf{T}(\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t) + \frac{\partial \boldsymbol{a}_{t+1}}{\partial \boldsymbol{h}_t} \frac{\partial \boldsymbol{h}_{t+1}}{\partial \boldsymbol{a}_{t+1}} \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}} \\
&= \mathbf{V}^\mathsf{T}(\hat{\boldsymbol{y}}_t - \boldsymbol{y}_t) + \mathbf{W}^\mathsf{T} \cdot \mathbf{diag}\left(1 - \boldsymbol{h}_{t+1}^2\right) \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_{t+1}}
\end{aligned}$$

Back-Propagation Through Time (BPTT)

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}} = \sum_t \textbf{diag}\left(1 - \boldsymbol{h}_t^2\right) \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \cdot \boldsymbol{h}_{t-1}^\mathsf{T}$$

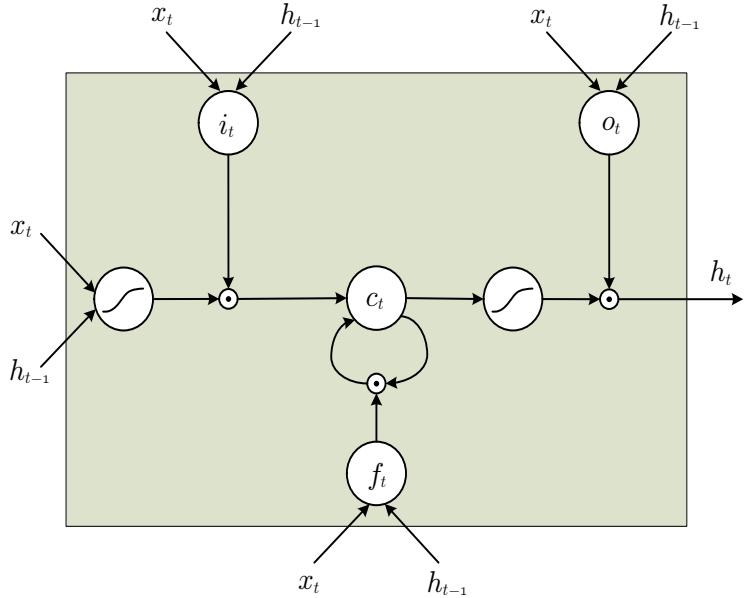
$$\frac{\partial \mathcal{L}}{\partial \mathbf{U}} = \sum_t \textbf{diag}\left(1 - \boldsymbol{h}_t^2\right) \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t} \cdot \boldsymbol{x}_t^\mathsf{T}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{b}} = \sum_t \textbf{diag}\left(1 - \boldsymbol{h}_t^2\right) \cdot \frac{\partial \mathcal{L}}{\partial \boldsymbol{h}_t}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{V}} = \sum_t \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t} \cdot \boldsymbol{h}_t^\mathsf{T}$$

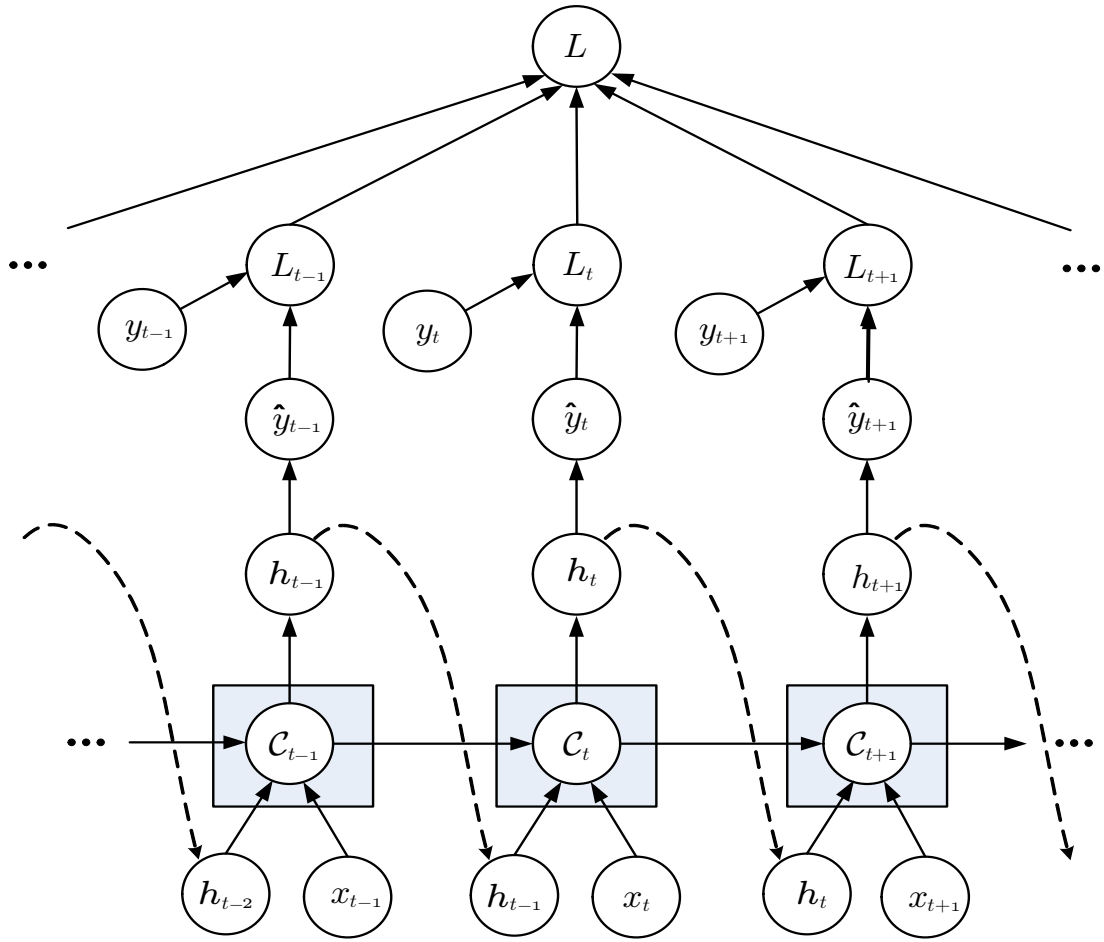
$$\frac{\partial \mathcal{L}}{\partial \mathbf{c}} = \sum_t \frac{\partial \mathcal{L}}{\partial \boldsymbol{o}_t}$$

An LSTM Memory Cell



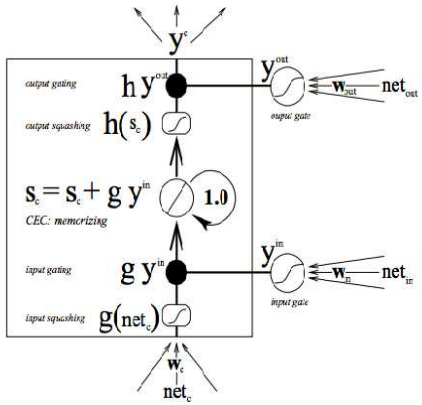
- input gate: $\boldsymbol{i}_t = \sigma(\mathbf{W}_{ix}\boldsymbol{x}_t + \mathbf{W}_{ih}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i)$
- forget gate: $\boldsymbol{f}_t = \sigma(\mathbf{W}_{fx}\boldsymbol{x}_t + \mathbf{W}_{fh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f)$
- cell candidate: $\boldsymbol{g}_t = \tanh(\mathbf{W}_{cx}\boldsymbol{x}_t + \mathbf{W}_{ch}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c)$
- cell: $\boldsymbol{c}_t = \boldsymbol{f}_t \odot \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \odot \boldsymbol{g}_t$
- output gate: $\boldsymbol{o}_t = \sigma(\mathbf{W}_{ox}\boldsymbol{x}_t + \mathbf{W}_{oh}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o)$
- hidden state output: $\boldsymbol{h}_t = \boldsymbol{o}_t \odot \tanh(\boldsymbol{c}_t)$

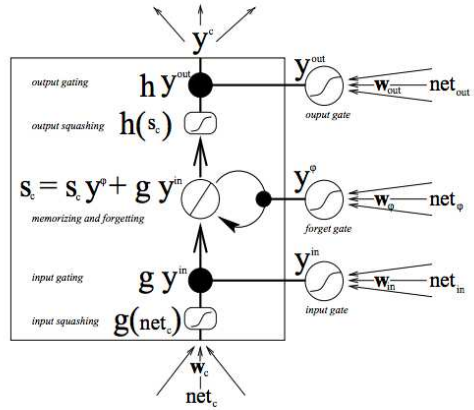
Unfolded LSTM Representation



Why LSTMs Work?

$$\frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_k} = \frac{\partial \boldsymbol{c}_t}{\partial \boldsymbol{c}_{t-1}} \frac{\partial \boldsymbol{c}_{t-1}}{\partial \boldsymbol{c}_{t-2}} \cdots \frac{\partial \boldsymbol{c}_{k+1}}{\partial \boldsymbol{c}_k} = \mathbf{diag}(\boldsymbol{f}_t \odot \boldsymbol{f}_{t-1} \odot \cdots \odot \boldsymbol{f}_{k+1})$$

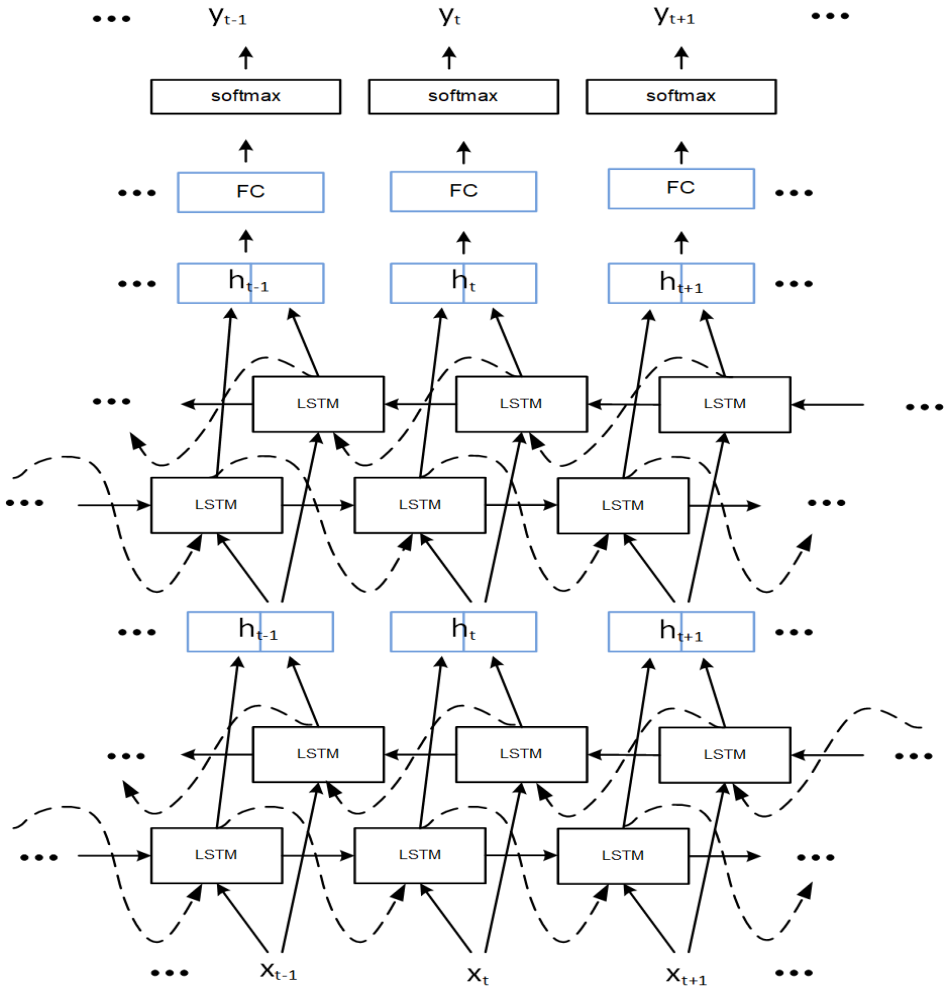




- Sepp Horchreiter and Jürgen Schmidhuber, "Long Short-Term Memory," Neural Computation 9, 1997
- a recurrent self-connection with weight 1.0 – constant error carousel (CEC)

- Felix A. Gers, Jürgen Schmidhuber and Fred Cummins. "Learning to Forget: Continual Prediction with LSTM." Neural Computation 12, 2000
- A forget gate added to learn to reset

A Bi-directional Multi-layer LSTM for ASR



A CLSTM for ASR