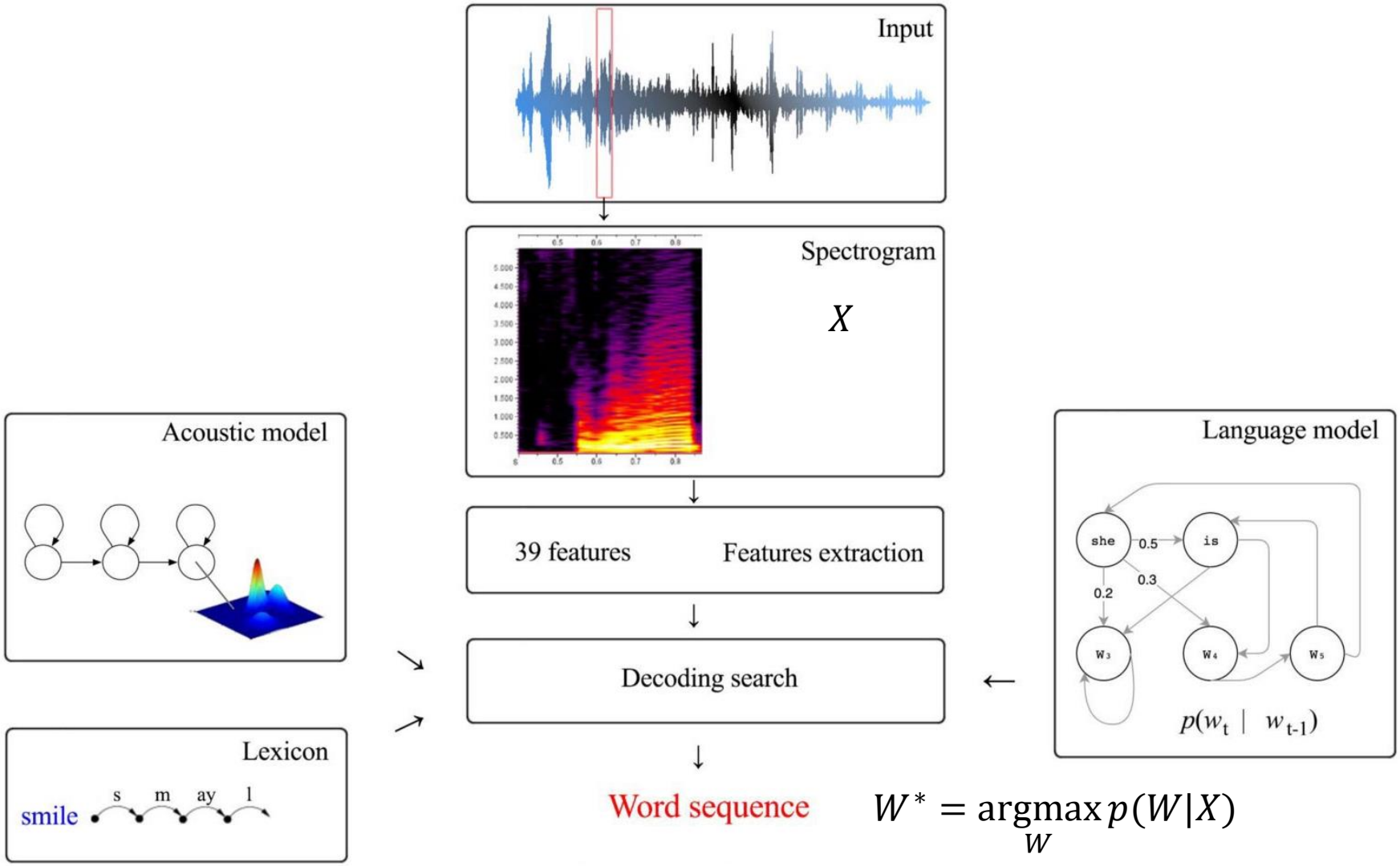
Deep Learning for Speech Recognition

Vassilis Tsiaras

November 2023

Speech Recognition

- Speech recognition can be viewed as finding the best sequence of words (W) given the audio features (X).



Speech Recognition

- The HMM-based speech recognition can be divided into three parts, each of which is independent of each other and plays a different role:
  a) acoustic,
  b) pronunciation and
  c) language model.
- The acoustic model is used to model the mapping between speech input and feature sequence (typically a phoneme or sub-phoneme sequence).
- The pronunciation model, which is typically constructed by professional human linguists, is to achieve a mapping between phonemes (or sub-phonemes) to graphemes.
- The language model maps the character sequence to fluent final transcription.

Speech Recognition

- Let X denote the sequence of audio features, S denote the sequence of HMM states and W denote the sequence of words.

$$W^* = \underset{W}{\operatorname{argmax}}\, p(W|X) = \underset{W}{\operatorname{argmax}} \frac{p(W,X)}{p(X)} = \underset{W}{\operatorname{argmax}}\, p(W,X) =$$

$$\underset{W}{\operatorname{argmax}} \sum_S p(W,S,X) = \underset{W}{\operatorname{argmax}} \sum_S p(X|W,S)p(W,S) =$$

$$\underset{W}{\operatorname{argmax}} \sum_S p(X|W,S)p(S|W)p(W) \approx$$

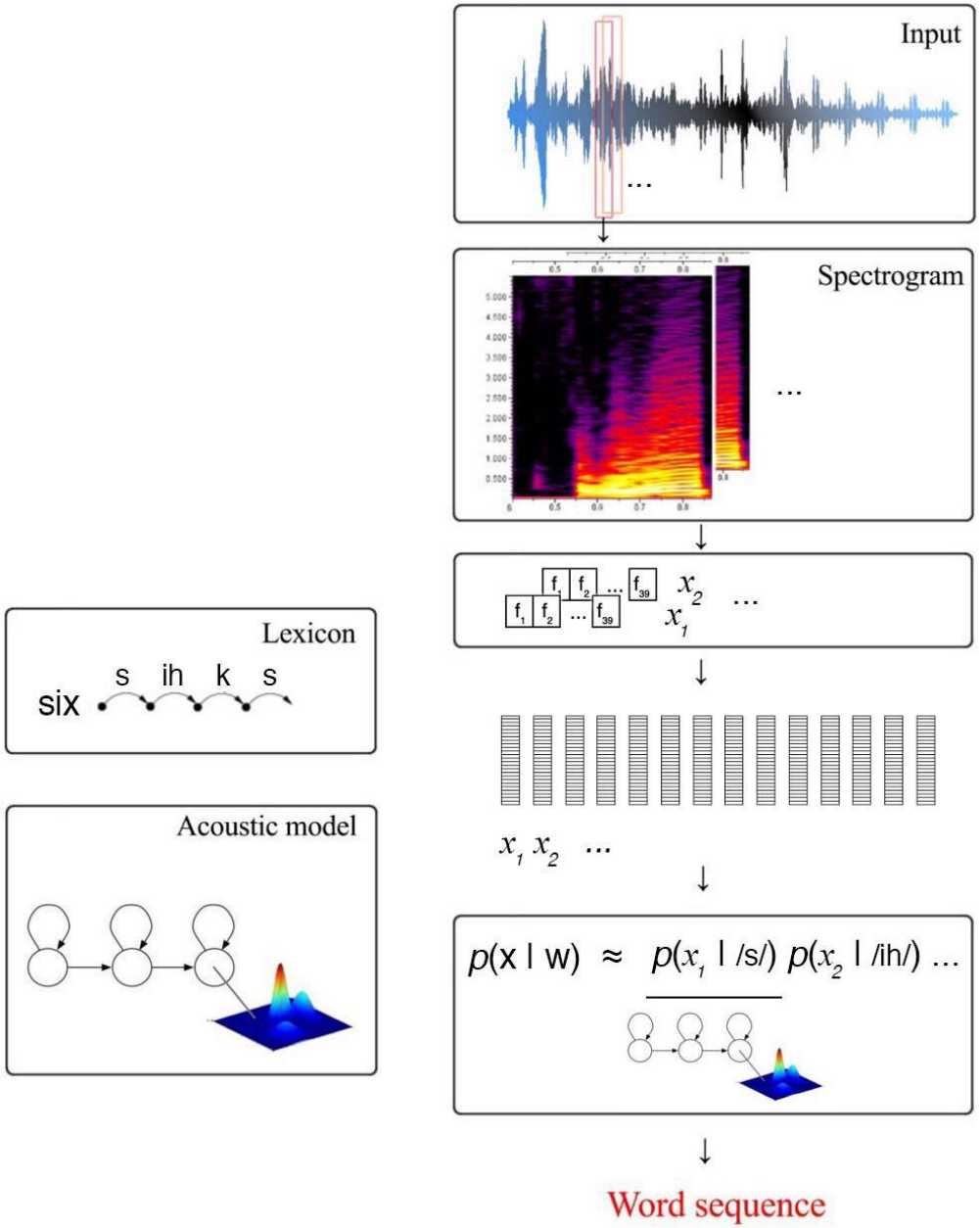
$$\underset{W}{\operatorname{argmax}} \sum_S p(X|S)p(S|W)p(W)$$

- These three factors $p(X|S)$, $p(S|W)$, $p(W)$ in the above equation correspond to acoustic model, pronunciation model and language model.

Pronunciation Model

- Pronunciation model p(S|W): this is also called the dictionary or lexicon.
- Its role is to achieve the connection between acoustic sequence and language sequence.
- The dictionary includes various levels of mapping, such as pronunciation to phone, phone to trip-hone.
- The dictionary is not only used to achieve structural mapping, but also to map the probability calculation relationship.

Pronunciation Model



the word “six”

phones: /s/ /ih/ /k/ /s/

Language Model

- Models likelihood of word given previous word(s)
- Language model is used to
  - Guide the search algorithm (predict next word given history)
  - Disambiguate between phrases which are acoustically similar
    - Great wine vs Grey twine
- A language model calculates the likelihood of a sequence of words.

$$W^* = \arg\max_W P(X|W) \; P(W)$$

acoustic model language model

- In a bigram (a.k.a. 2-gram) language model, the current word depends on the last word only.

$$P(W_n \mid W_1, W_2, \ldots, W_{n-1}) \approx P(W_n \mid W_{n-1})$$

- For example,

$P(W)$ = P("Simplicity is the ultimate sophistication")

$\approx P$("Simplicity") P("is" | "Simplicity") P("the" | "is") P("ultimate" | "the") P("sophistication" | "ultimate")

Language Model

- To compute P("zero"|"two"), we search the corpus (say from Wall Street Journal corpus that contains 23M words) and calculate

$$P(\text{"zero"}|\text{"two"}) = \frac{\text{count ("two zero")}}{\text{count ("two")}}$$

- If the language model depends on the last 2 words, it is called trigram.

$$P(w_1, \ldots, w_L) = \prod_{i=1}^{L+1} P(w_i|w_{i-2}w_{i-1})$$

$$P(w_i|w_{i-2}w_{i-1}) = \frac{c(w_{i-2}w_{i-1}w_i)}{c(w_{i-2}w_{i-1})}$$ count in the corpus

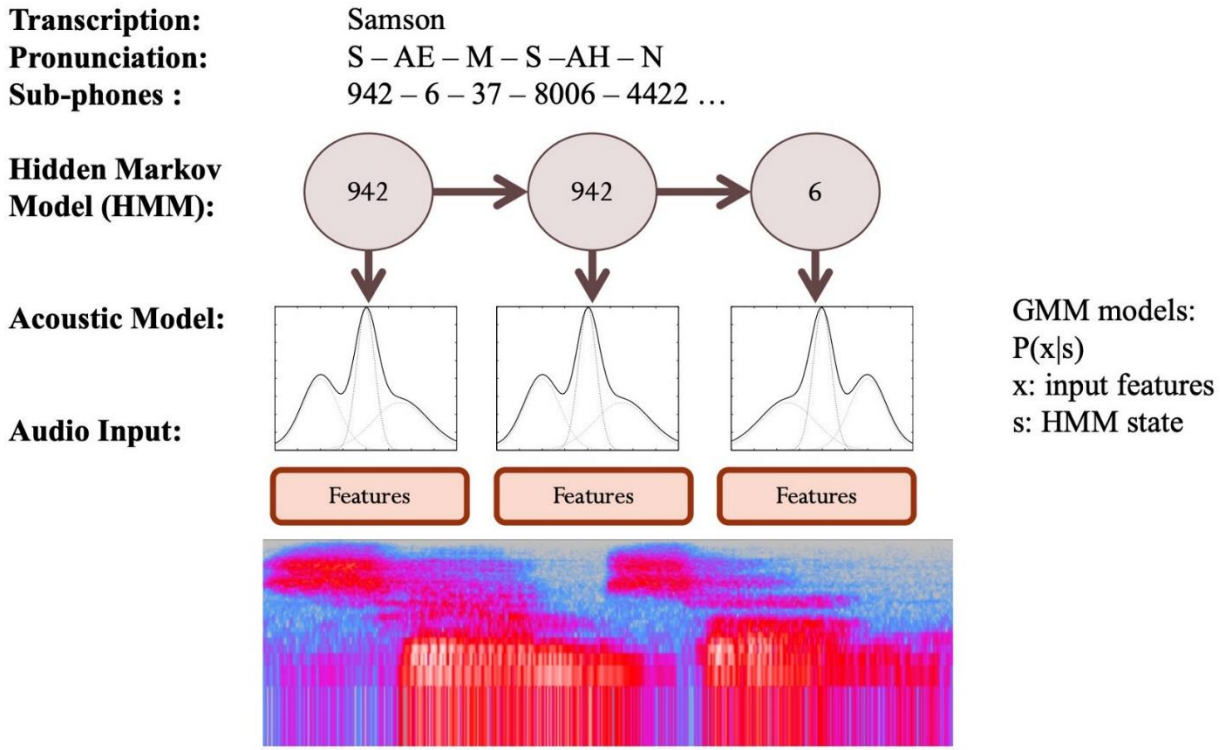
- n-gram depends on the last n-1 words.

$$P(w_N|w_1, w_2, \cdots, w_{N-1})$$

Acoustic model

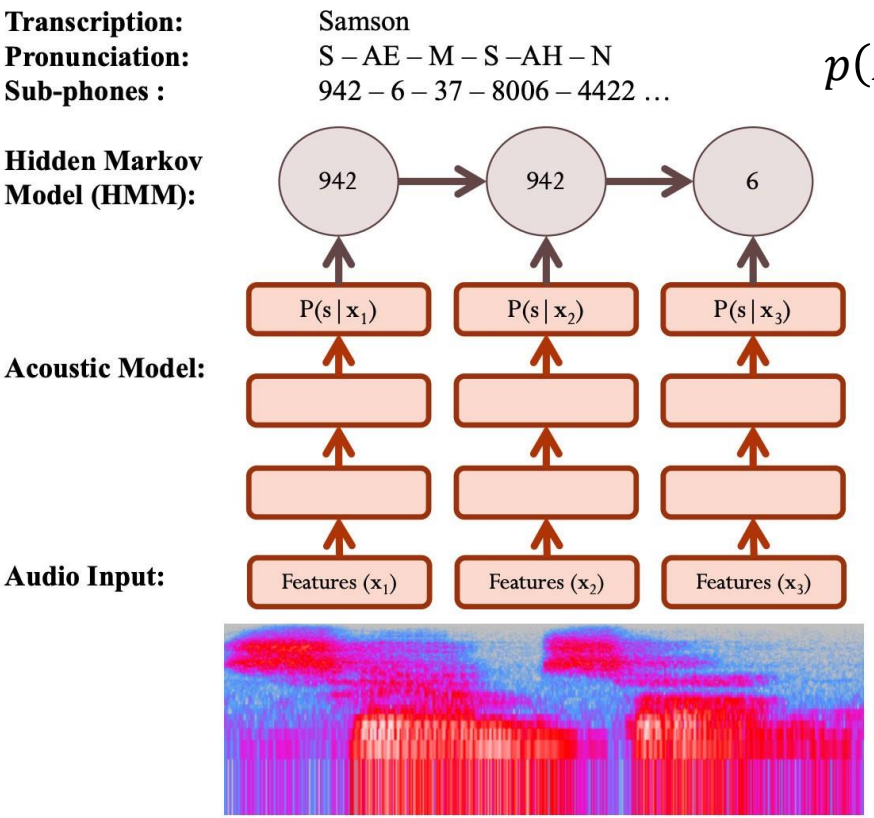
- Acoustic model P(X|S): It indicates the probability of observing X from hidden sequence S. According to the probability chain rule and the observation independence hypothesis in HMM (observations at any time depend only on the hidden state at that time), P(X|S) can be decomposed into the following form:

$$p(X|S) = \prod_{t=1}^{T} p(x_t|x_1, \dots, x_{t-1}, S) \approx \prod_{t=1}^{T} p(x_t|s_t) \propto \prod_{t=1}^{T} \frac{p(s_t|x_t)}{p(s_t)}$$



Acoustic model

- In the acoustic model, $p(x_t|s_t)$ is the observation probability, which is generally represented by GMM. The posterior probability distribution of hidden state $p(s_t|x_t)$ can be calculated by DNN method. These two different calculations of $p(X|S)$ result into two different models, namely HMM-GMM and HMM-DNN. The role of DNN is to calculate the posterior probability of the HMM state, which may be transformed into likelihoods, replacing the conventional GMM observation probability.



$$p(X|S) \approx \prod_{t=1}^{T} p(x_t|s_t) \propto \prod_{t=1}^{T} \frac{p(s_t|x_t)}{p(s_t)}$$

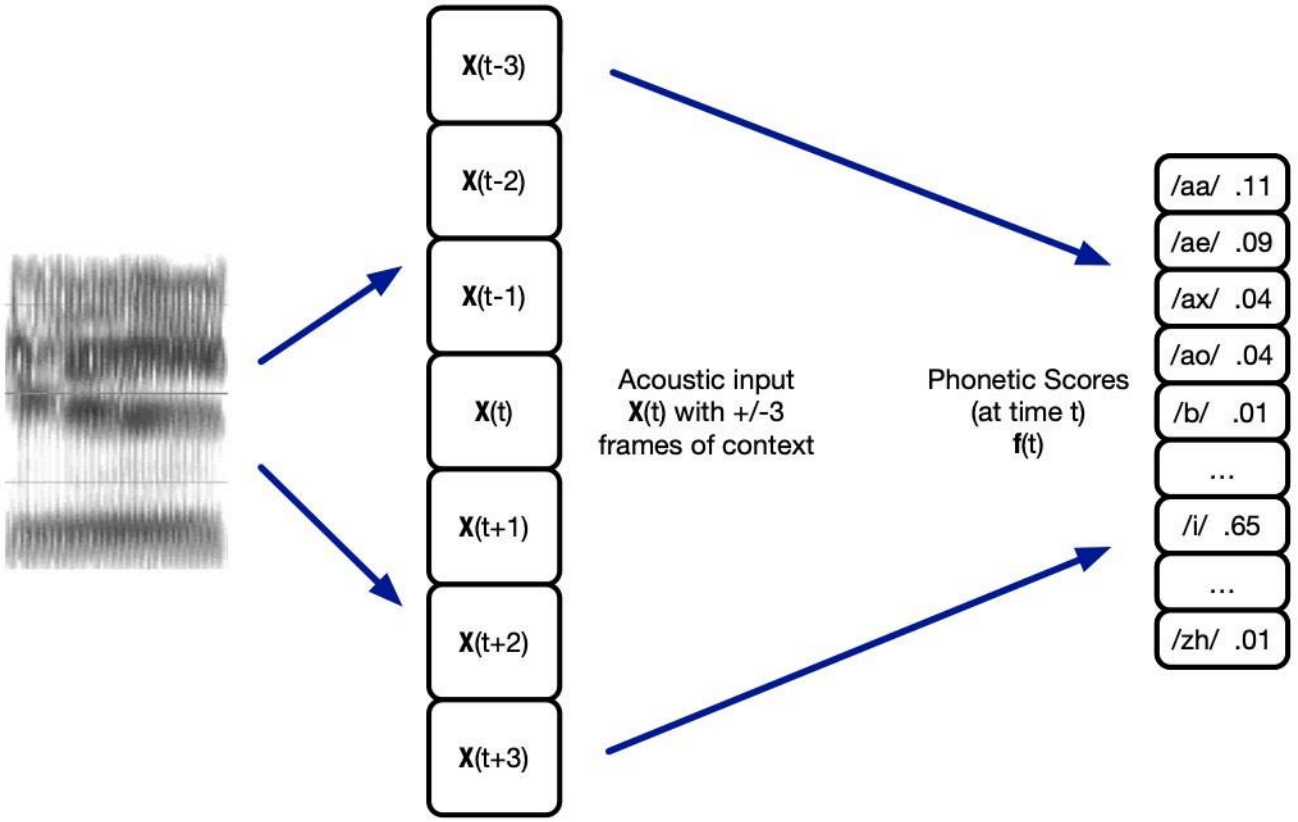
Use a DNN to approximate:
P(s|x)

Apply Bayes' Rule:
P(x|s) = P(s|x) * P(x) / P(s)

DNN * Constant / State prior

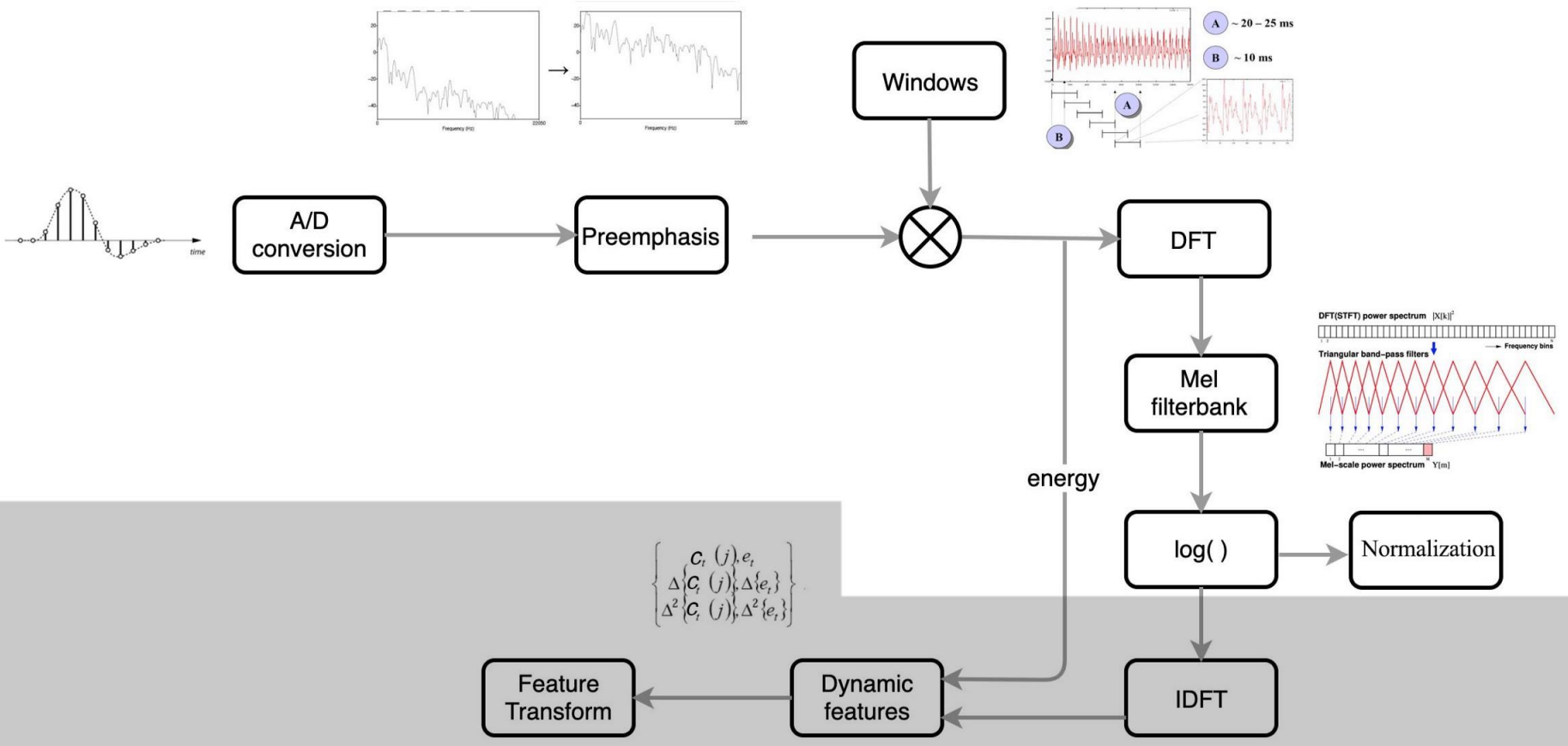
Acoustic model

- Given the features in an audio frame, we can use a deep network to predict p(s|x) and apply Bayes' Theorem to estimate $p(x|s) \propto p(s|x) / p(s)$.
- In addition, we can include the neighbour frames as input. This creates a better phone context for better predictions.



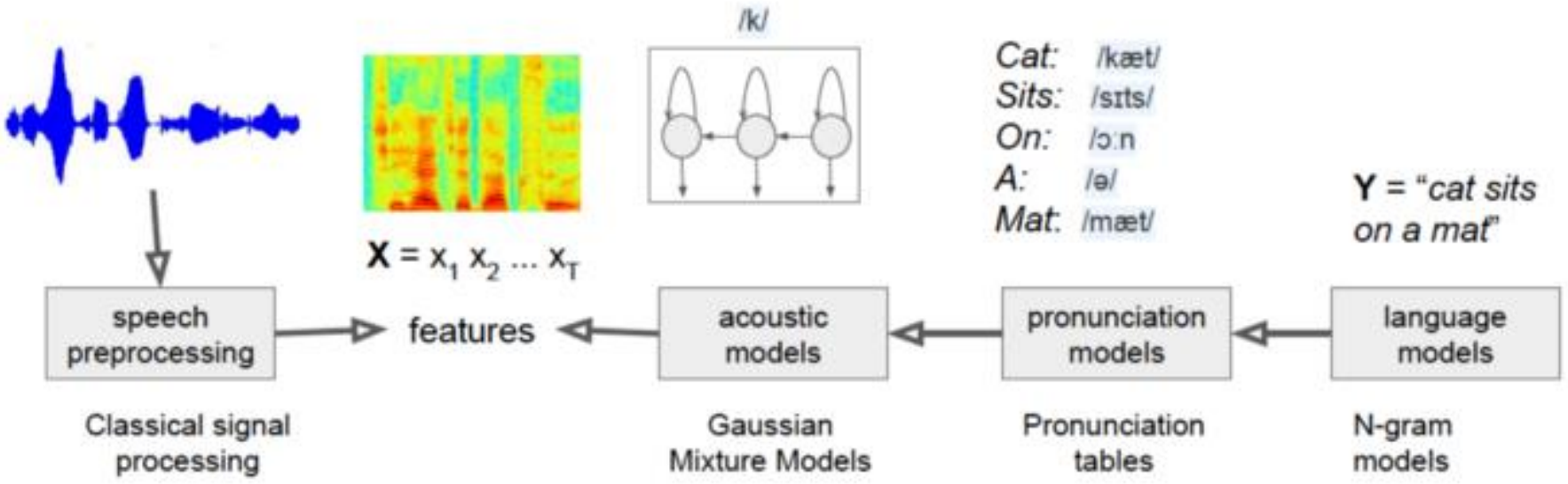
Feature extraction

- ML speech recognition, we extract MFCC features from the audio frames. It includes steps like an inverse discrete Fourier transform to make features less correlated with each other. Also, we only take the lower 12 coefficients.
- DL requires no or less pre-processing. It can handle a much larger number of input features and there is no particular need to un-correlate them first.



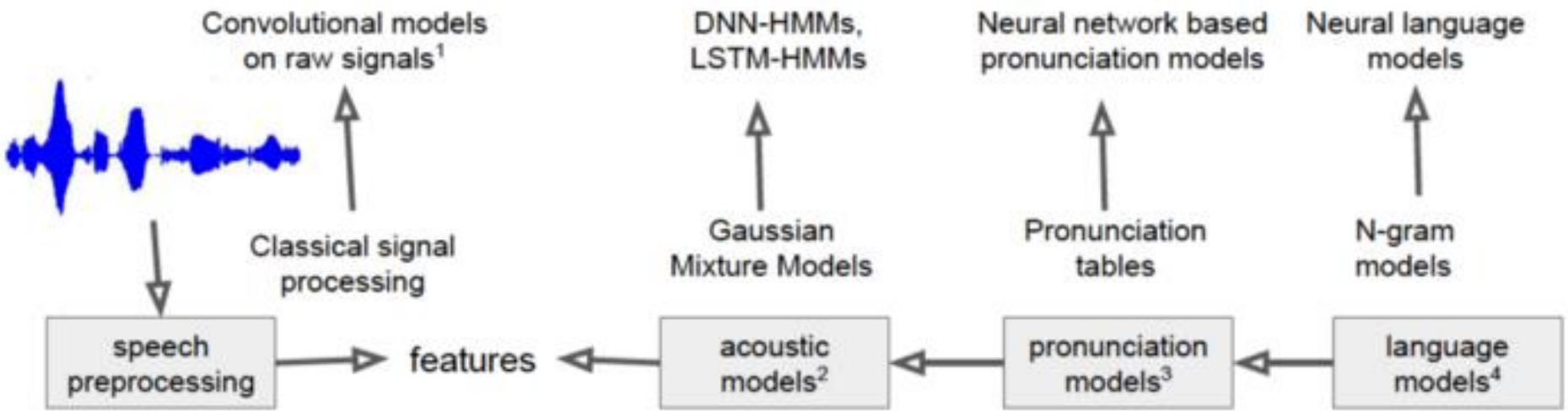
Classical Speech Recognition

- In the past, language models were typically N-gram models, which are essentially tables describing the probabilities of token sequences.
- The pronunciation models were simple lookup tables with probabilities associated with pronunciations.
- Acoustic models are built using Gaussian Mixture Models with very specific architectures associated with them.
- The speech processing was pre-defined.



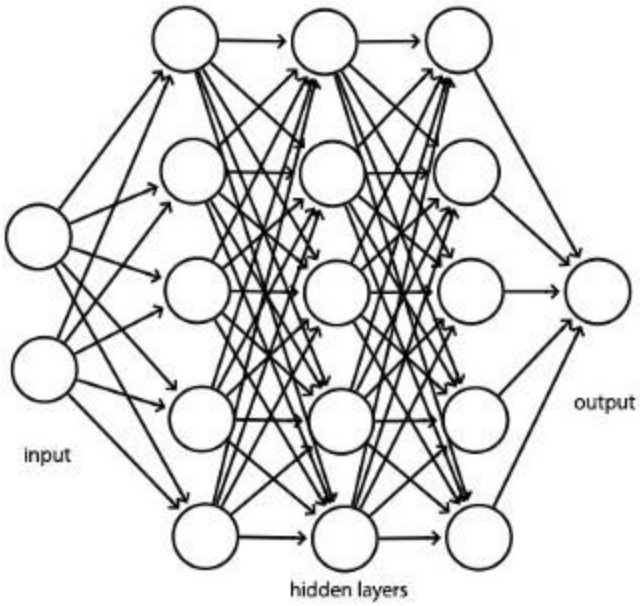
Neural Network Speech Recognition

- Instead of the N-gram language models, we can build neural language models and feed them into a speech recognition system.
- A neural network can infer pronunciation for a new sequence of characters that has never seen before.
- For acoustic models, we can build deep neural networks to get much better classification accuracy scores for the current frame.
- Even the speech pre-processing steps were found to be replaceable with convolutional neural networks on raw speech signals.



NN Acoustic models

- What neural networks are used as acoustic models?
- Fully connected network



- The input is the Mel filter bank. Some ASR FC models contain 3–8 hidden layers with 2048 hidden units in each layer.
- This model can predict the distribution of the context-dependent states (say 9304 CD triphones) from the audio frames.

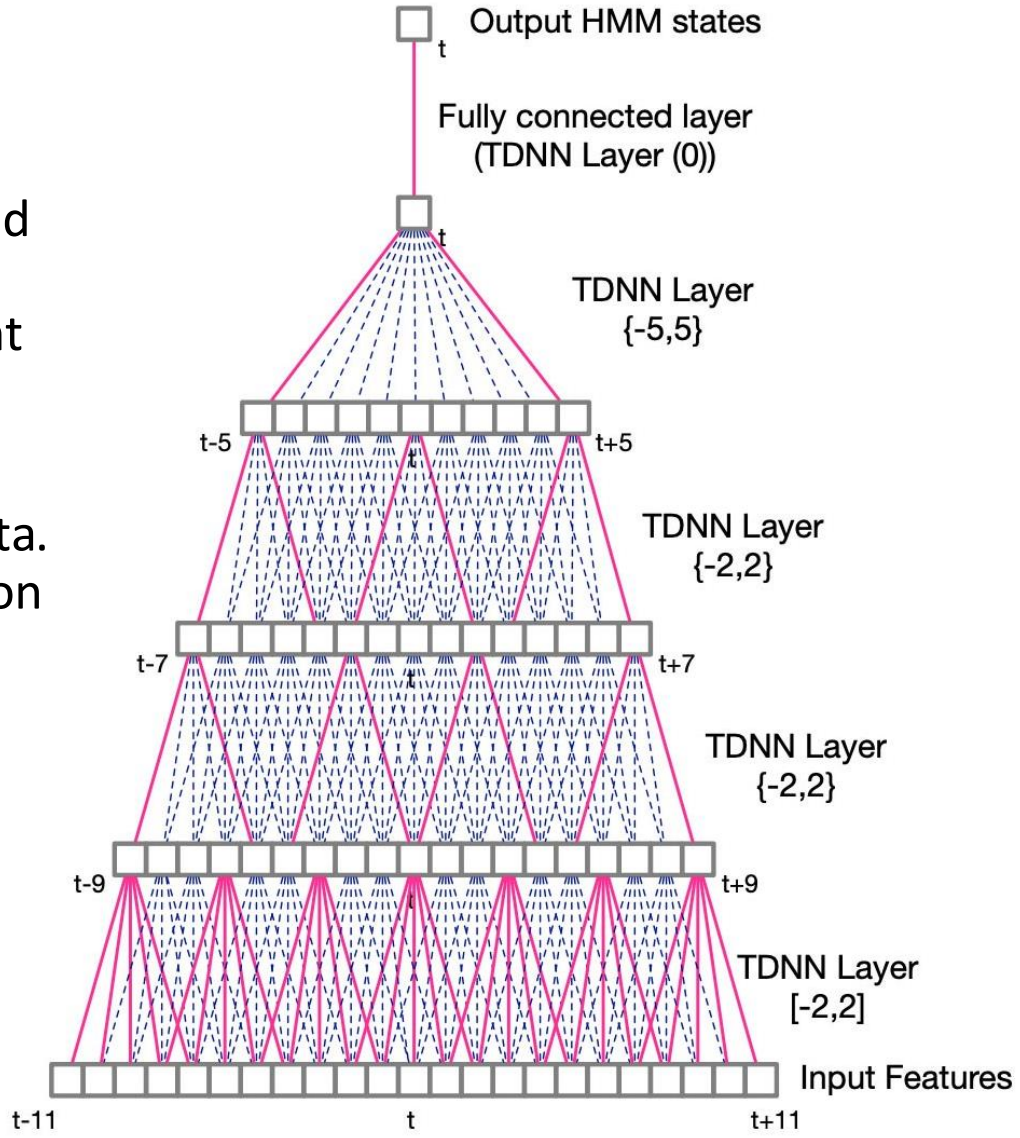
NN Acoustic models

- Fully connected networks vs GMM-HMM

TASK	HOURS OF TRAINING DATA	DNN-HMM	GMM-HMM WITH SAME DATA	GMM-HMM WITH MORE DATA
SWITCHBOARD (TEST SET 1)	309	18.5	27.4	18.6 (2,000 H)
SWITCHBOARD (TEST SET 2)	309	16.1	23.6	17.1 (2,000 H)
ENGLISH BROADCAST NEWS	50	17.5	18.8	
BING VOICE SEARCH (SENTENCE ERROR RATES)	24	30.4	36.2	
GOOGLE VOICE INPUT	5,870	12.3		16.0 (>> 5,870 H)
YOUTUBE	1,400	47.6	52.3	

NN Acoustic models

- CNN/TDNN
- FC networks are computationally intense.
- CNN takes advantage of locality and discovers local information hierarchically. CNN is more efficient if the information has a strong spatial relationship.
- Audio speech is time-sequence data. Instead of applying a 2D convolution filter, we use a 1-D filter to extract features across multiple frames in time. This is the Time-delay neural networks (TDNN).



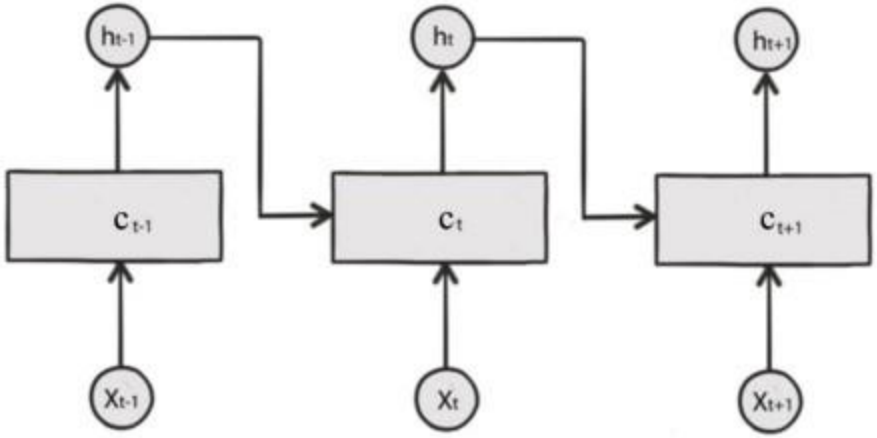
NN Acoustic models

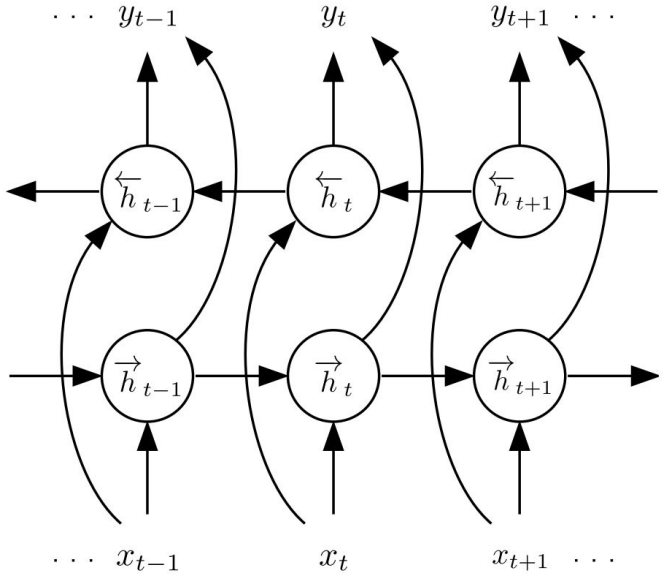
- Fully connected networks (DNN) vs TDNN

Database	Size	WER		Rel. Change
		DNN	TDNN	
Res. Management	3h hrs	2.27	2.30	-1.3
Wall Street Journal	80 hrs	6.57	6.22	5.3
TedLIUM	118 hrs	19.3	17.9	7.2
Switchboard	300 hrs	15.5	14.0	9.6
Librispeech	960 hrs	5.19	4.83	6.9
Fisher English	1800 hrs	22.24	21.03	5.4

NN Acoustic models

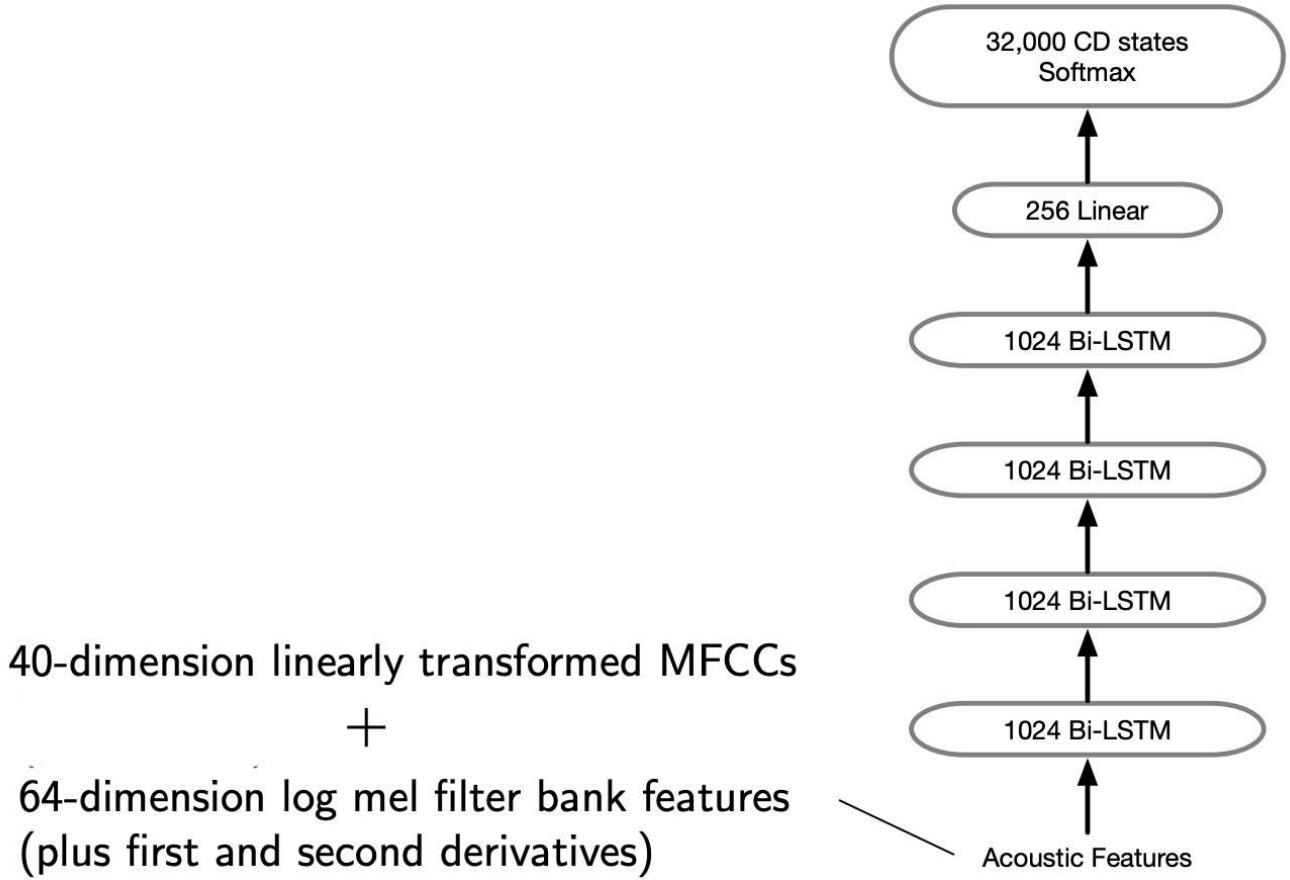
- RNN (LSTM & GRU)
- RNNs are designed for time-sequence data. Each cell contains a cell state c and outputs h. The cell state is determined by the current input and the previous states including the previous cell state and the previous output.





NN Acoustic models

- RNN (LSTM & GRU)
- Classifying context dependent (CD) states.

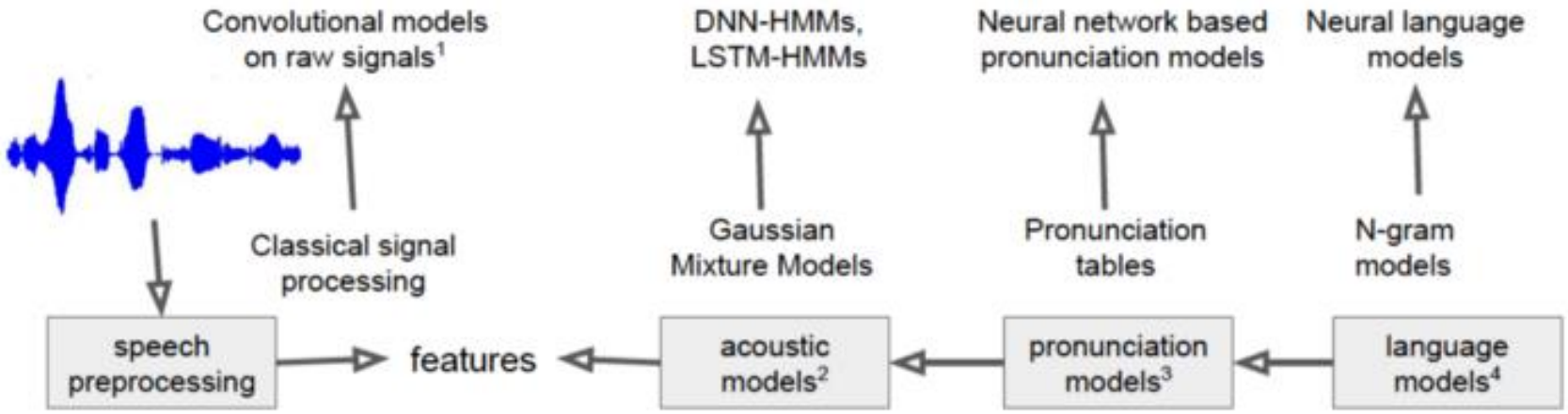


NN Acoustic models

- Comparison of different acoustic models

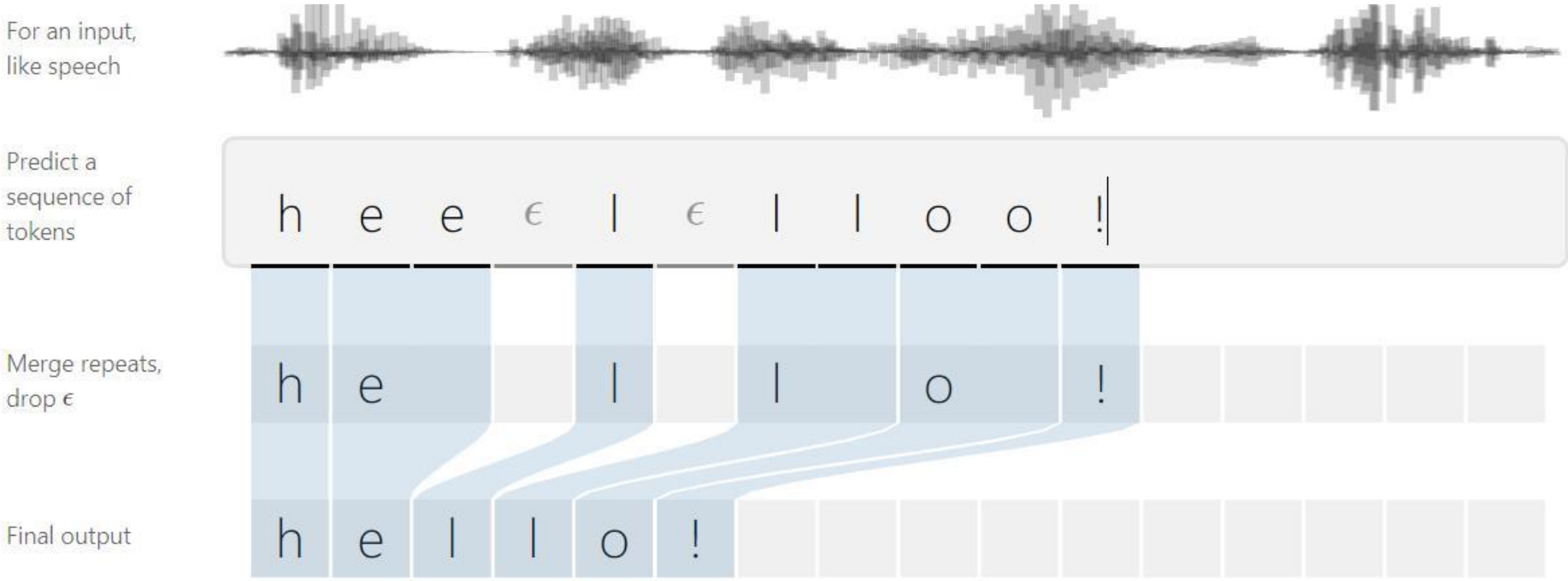
Network Architecture	Test Set WER/% Switchboard	CallHome
GMM (ML)	21.2	36.4
GMM (BMMI)	18.6	33.0
DNN (7x2048) / CE	14.2	25.7
DNN (7x2048) / MMI	12.9	24.6
TDNN (6x1024) / CE	12.5	
TDNN (6x576) / LF-MMI	9.2	17.3
LSTM (4x1024)	8.0	14.3
LSTM (6x1024)	7.7	14.0
LSTM-6 + feat fusion	7.2	12.7

End-to-end Speech Recognition



- In the above figure, there are neural networks in each component, but they're trained independently with different objectives.
- Train the entire model as one big component itself.
- These so-called end-to-end models encompass more and more components in the pipeline. The 2 most popular ones are:
  1. Connectionist Temporal Classification (CTC)
  2. Sequence-To-Sequence (Seq-2-Seq)

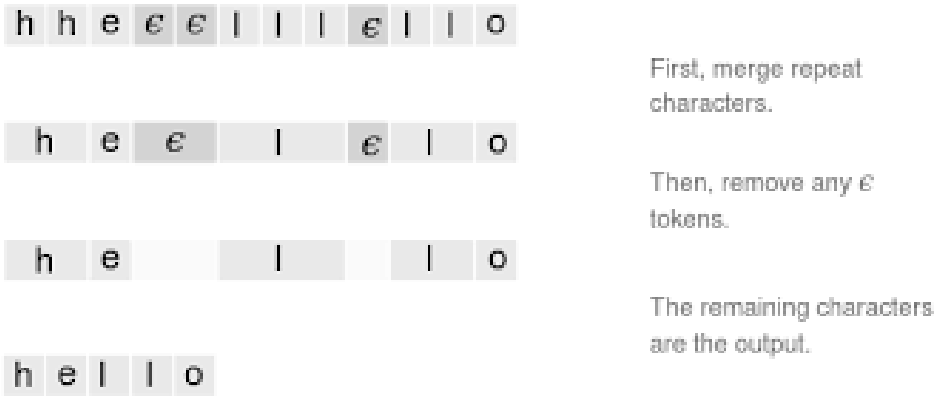
Connectionist Temporal Classification



- Input sequence of frames of audio features $X = [x_1, x_2 \dots, x_T]$
- Output sequence of phonemes $Y = [y_1, y_2, \dots, y_U]$
- $T > U$
- We do not have an alignment
- Loss function: maximize the probability $p(Y|X)$
- Inference: $Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y|X)$

Connectionist Temporal Classification

- **Alignment**
- One way to align X and Y is to assign an output character to each input step and collapse repeats.
- Introduce a blank token ϵ
- The alignments allowed by CTC are the same length as the input. We allow any alignment which maps to Y after merging repeats and removing ϵ tokens:



Connectionist Temporal Classification

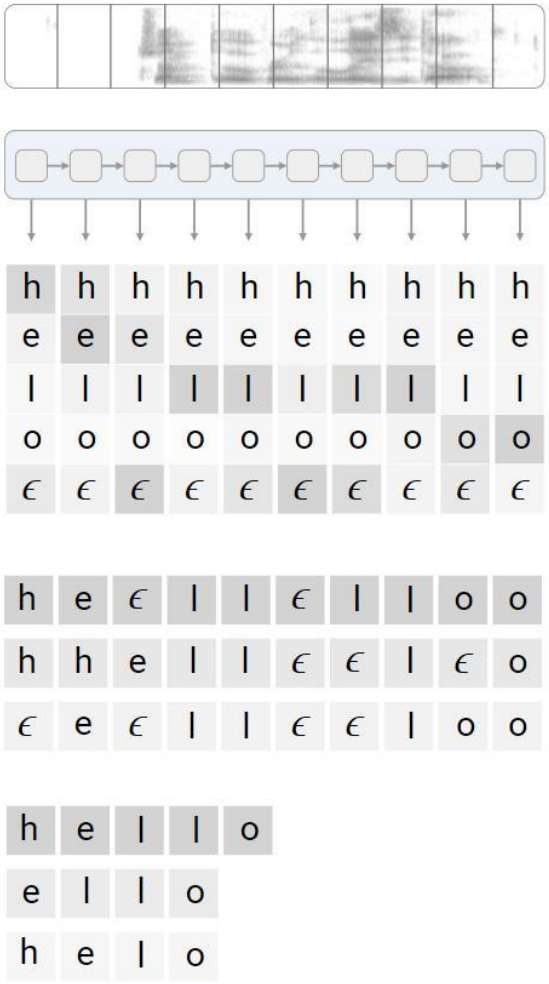
- **Loss**
- The CTC alignments give us a natural way to go from probabilities at each time-step to the probability of an output sequence.
- The CTC objective for a single (X, Y) pair is

$$p(Y|X) = \sum_{A \in A_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X)$$

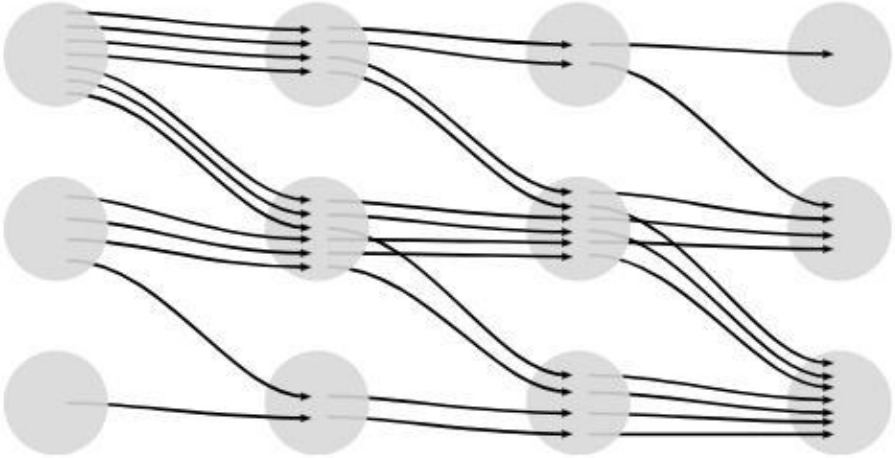
The CTC conditional probability

Marginalizes over the set of valid alignments

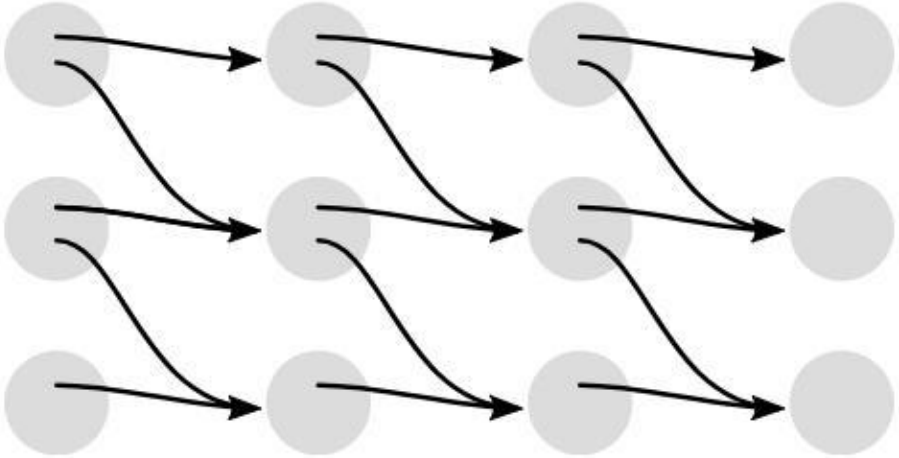
The probability of a single alignment at time t



Connectionist Temporal Classification



Summing over all alignments can be very expensive.



Dynamic programming merges alignments, so it's much faster.

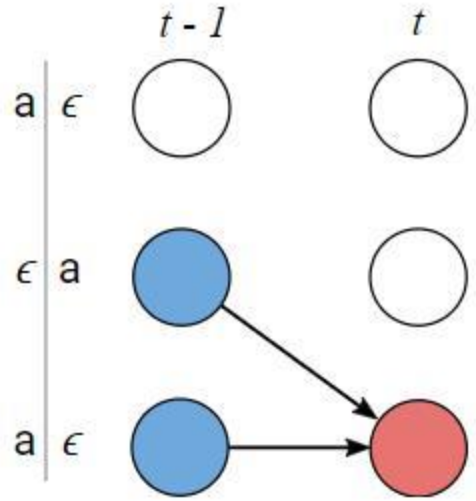
- Since we can have an ϵ before or after any token in Y, it's easier to describe the algorithm using a sequence which includes them. We'll work with the sequence $Z = [\epsilon, y_1, \epsilon, y_2, \dots, \epsilon, y_U, \epsilon]$
- Let $a_{s,t}$ be the CTC score of the subsequence $Z_{1:s}$ after t input steps. As we'll see, we'll compute the final CTC score, $p(Y|X)$, from the a's at the last time-step.

Connectionist Temporal Classification

- As long as we know the values of a at the previous time-step, we can compute $a_{s,t}$.
- There are two cases.
- Case 1:
- In this case, we can't jump over z_{s-1}, the previous token in Z. The first reason is that the previous token can be an element of Y, and we can't skip elements of Y. Since every element of Y in Z is followed by an ϵ, we can identify this when $z_s = \epsilon$. The second reason is that we must have an ϵ between repeat characters in Y. We can identify this when $z_s = z_{s-2}$.

$$a_{s,t} = (a_{s-1,t-1} + a_{s,t-1}) \cdot p_t(z_s|X)$$

Example of not allowed transitions

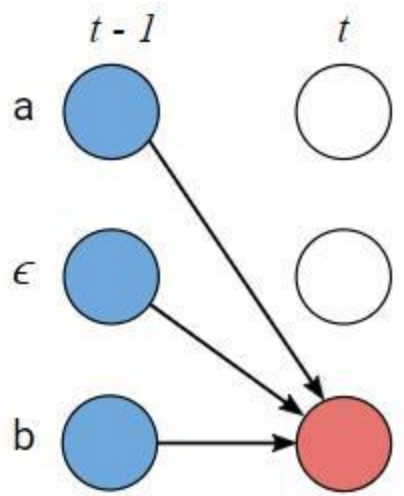


Connectionist Temporal Classification

- Case 2:
- In the second case, we're allowed to skip the previous token in Z. We have this case whenever z_{s-1} is an ϵ between unique characters. As a result there are three positions we could have come from at the previous step.

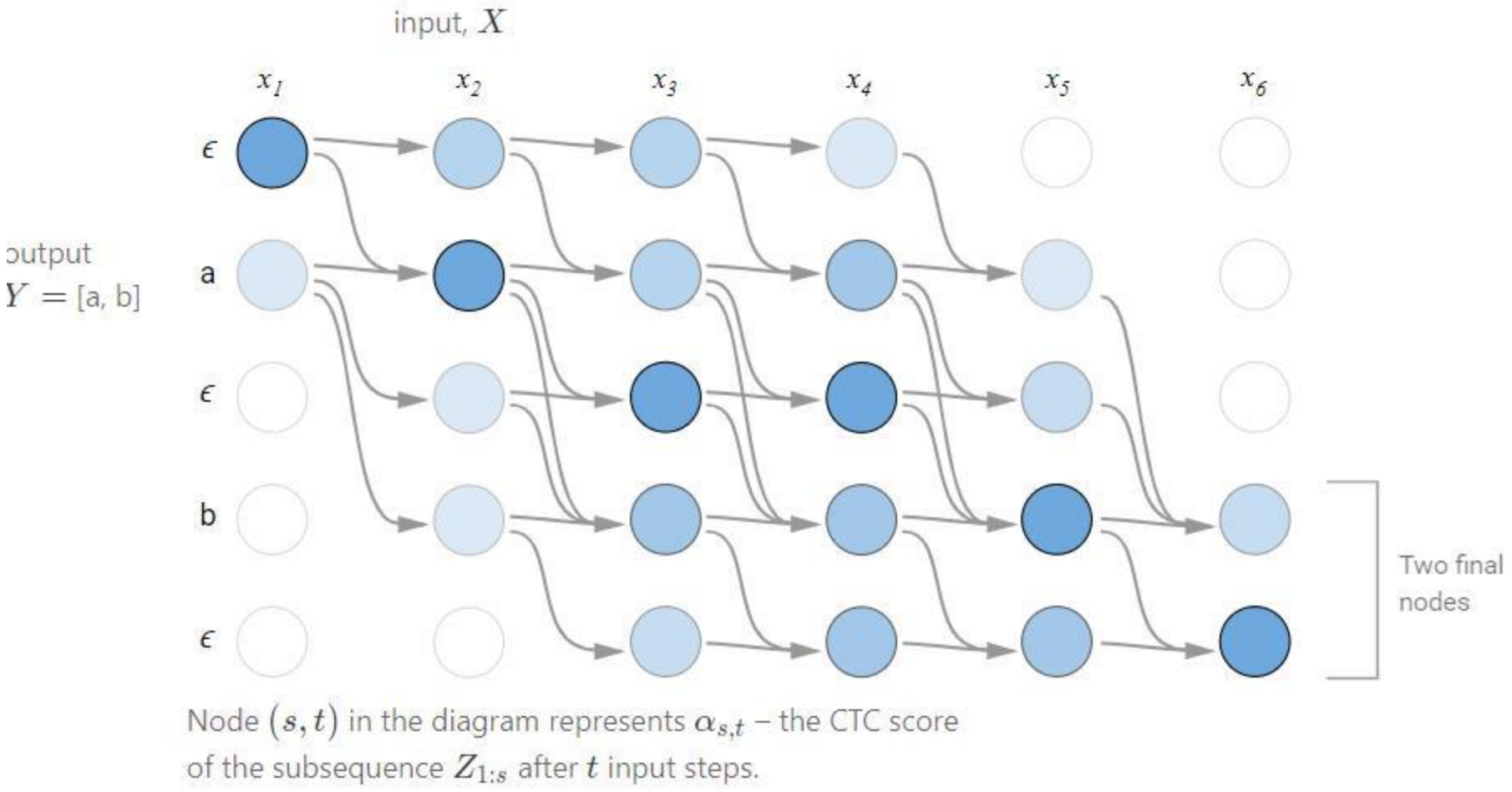
$$a_{s,t} = (a_{s-2,t-1} + a_{s-1,t-1} + a_{s,t-1}) \cdot p_t(z_s|X)$$

Example of allowed transitions



Connectionist Temporal Classification

- Example



Connectionist Temporal Classification

- **Loss**
- For a training set D, the model's parameter are tuned to minimize the negative log-likelihood

$$\sum_{(X,Y)\in D} -\log p(Y|X)$$

- **Inference**
- After we've trained the model, we'd like to use it to find a likely output for a given input. More precisely, we need to solve:

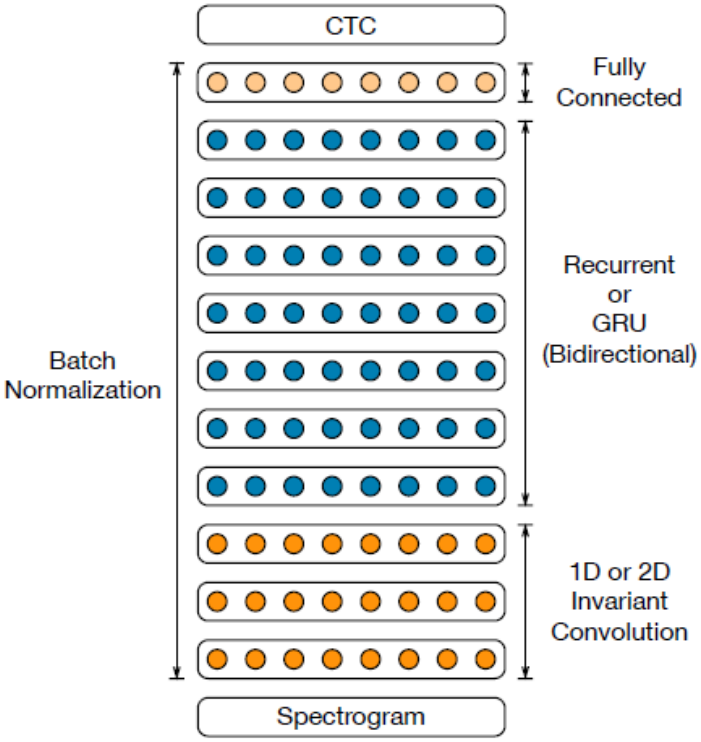
$$Y^* = \underset{Y}{\operatorname{argmax}}\, p(Y|X)$$

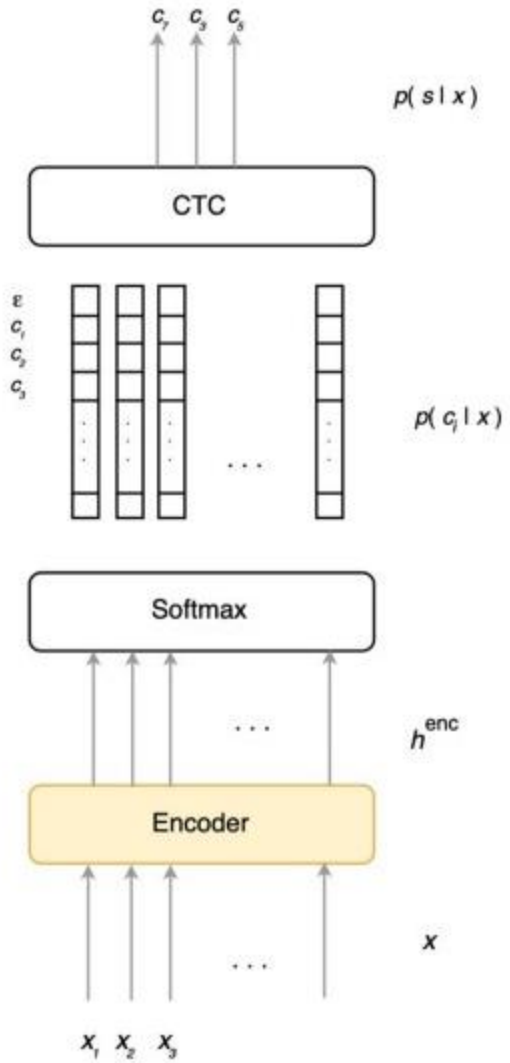
- Search for the best Y, among all possible sequences Y with length less than X.
- One heuristic is to take the most likely output at each time-step. This gives us the alignment with the highest probability:

$$A^* = \underset{A}{\operatorname{argmax}} \prod_{t=1}^{T} p_t(a_t|X)$$

- A modified beam search is a better heuristic

Deep Speech 2





Simple audio recognition

- This tutorial will show you how to build a basic speech recognition network that recognizes ten different words from 1 sec audio.
  - "down", "go", "left", "no", "right", "stop", "up" and "yes".