Deep Learning for Speech Recognition

YUAN-FU LIAO

NATIONAL TAIPEI UNIVERSITY OF TECHNOLOGY

Outline

History of Automatic Speech Recognition

Hidden Markov Model (HMM) based Automatic Speech Recognition

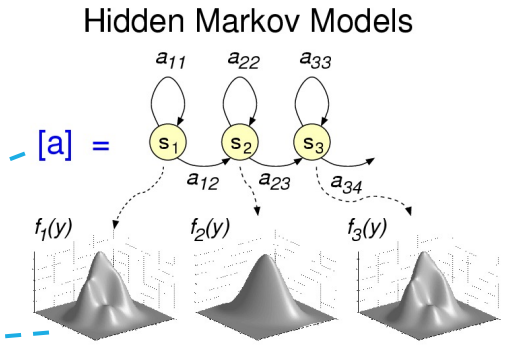
- Gaussian mixture models with HMMs
- Deep models with HMMs ➔ Hybrid HMM/DNN

End-to-End Deep Models based Automatic Speech Recognition

- Connectionist Temporal Classification (CTC)
- Attention based models ➔ Listen, Attend and Spell (LAS)

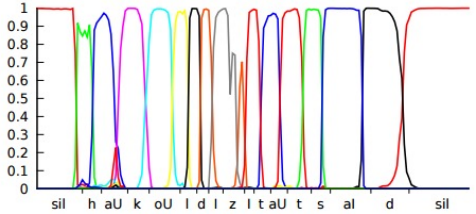
Automatic Speech Recognition

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{W}|\boldsymbol{O}) = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{A}|\boldsymbol{O})P(\boldsymbol{O}|\boldsymbol{Q})P(\boldsymbol{Q}|\boldsymbol{L})P(\boldsymbol{L}|\boldsymbol{W})P(\boldsymbol{W})$$



$P(\boldsymbol{A}|\boldsymbol{O})$ | $P(\boldsymbol{O}|\boldsymbol{Q})$ | $P(\boldsymbol{Q}|\boldsymbol{L})$ | $P(\boldsymbol{L}|\boldsymbol{W})$ | $P(\boldsymbol{W})$

Feature Extraction → Frame Classification → Sequence Model → Lexicon Model → Language Model



t ah m aa t ow

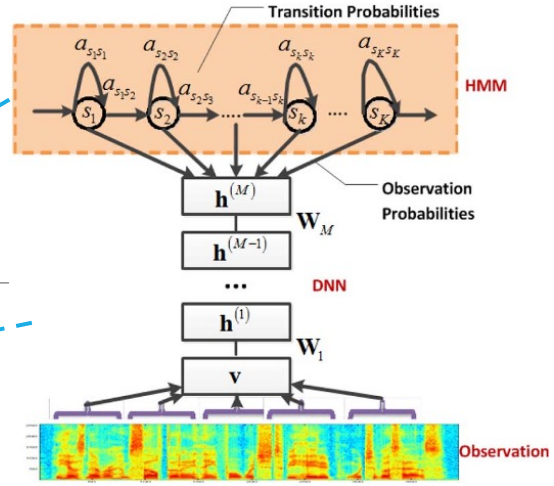
$\boldsymbol{A}$	$\boldsymbol{O}$	$\boldsymbol{Q}$	$\boldsymbol{L}$	$\boldsymbol{W}$	
Speech Audio	Feature Frames	Sequence States	Phonemes	Words	Sentence

DNN-HMM

Kaldi

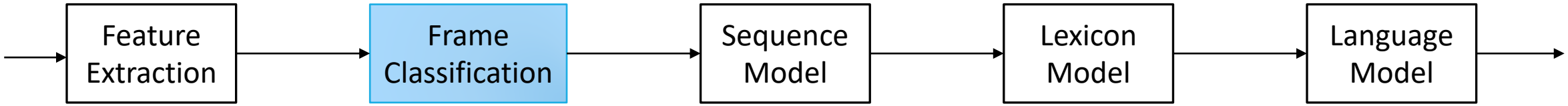
DNN: Deep Neural Networks

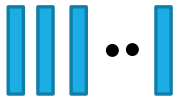
HMMs: Hidden Markov Models

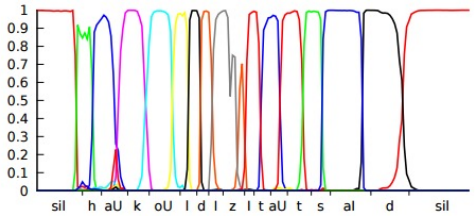


DNNs

HMMs







t ah m aa t ow

A Speech Audio

O Feature Frames

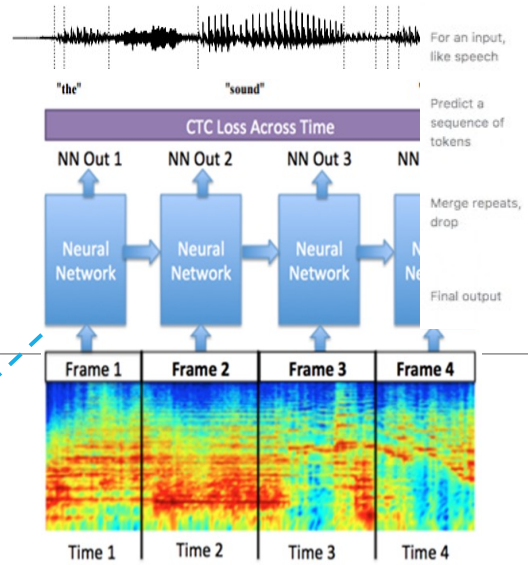
Q Sequence States

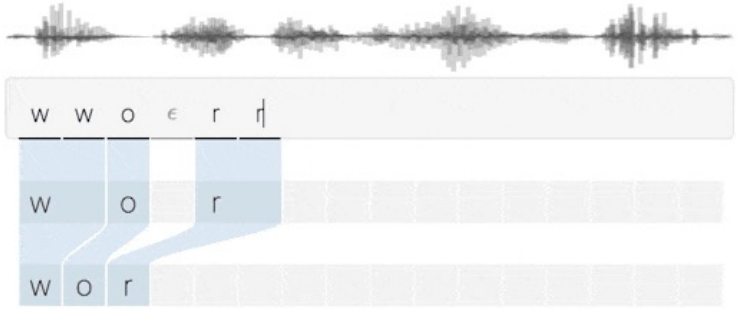
L Phonemes

W Words

Sentence

End-to-End: CTC



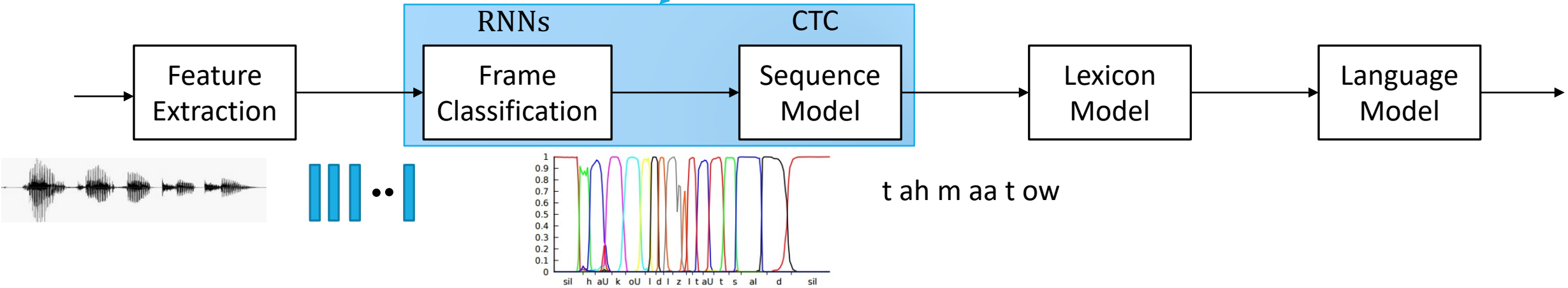


DeepSpeech2

RNN: Recurrent Neural Networks

CTC: Connectionist Temporal Classification

$$O^{ML}(S, \mathcal{N}_w) = -\sum_{(x,z)\in S} \ln\left(p(z|x)\right)$$



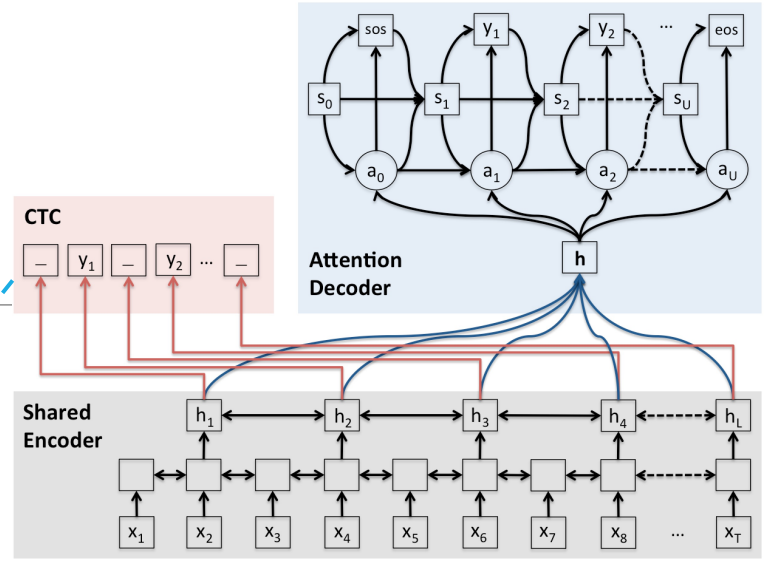
A	O	Q	L	W	
Speech Audio	Feature Frames	Sequence States	Phonemes	Words	Sentence

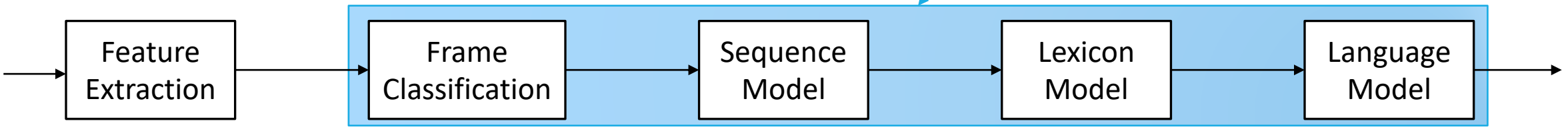
End-to-End: Attention

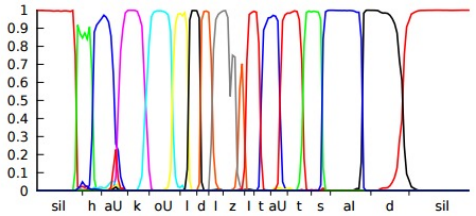
Listen, Attend and Spell (LAS)

Predict character sequence directly

Sequence-to-Sequence with Attention



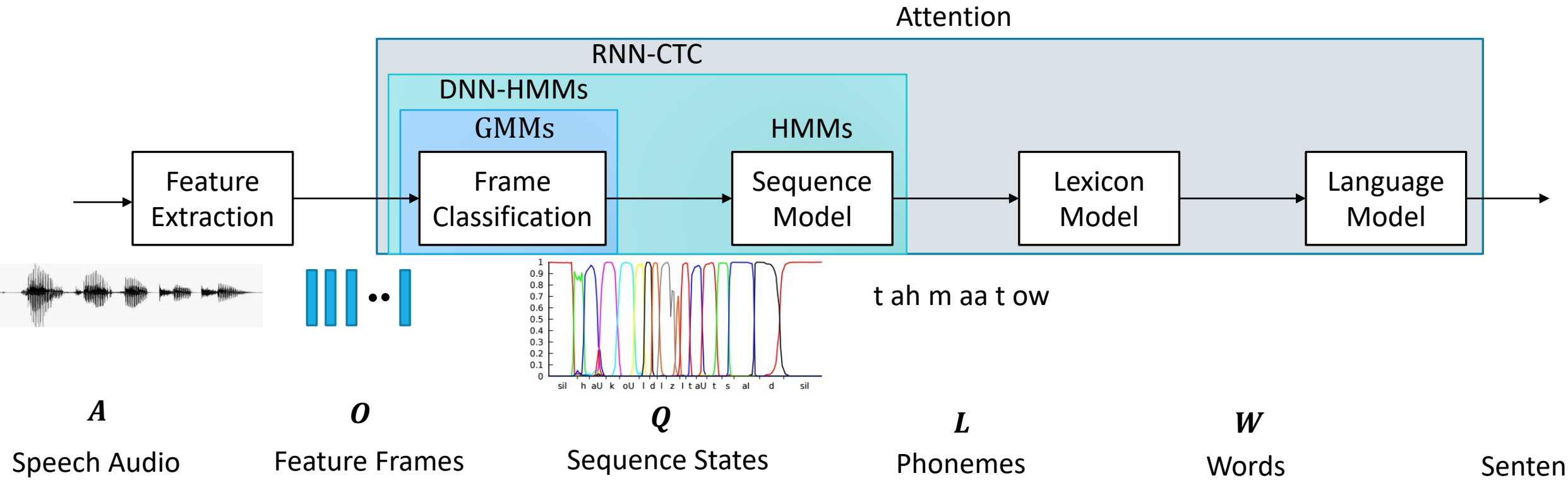




t ah m aa t ow

A	O	Q	L	W	
Speech Audio	Feature Frames	Sequence States	Phonemes	Words	Sentence

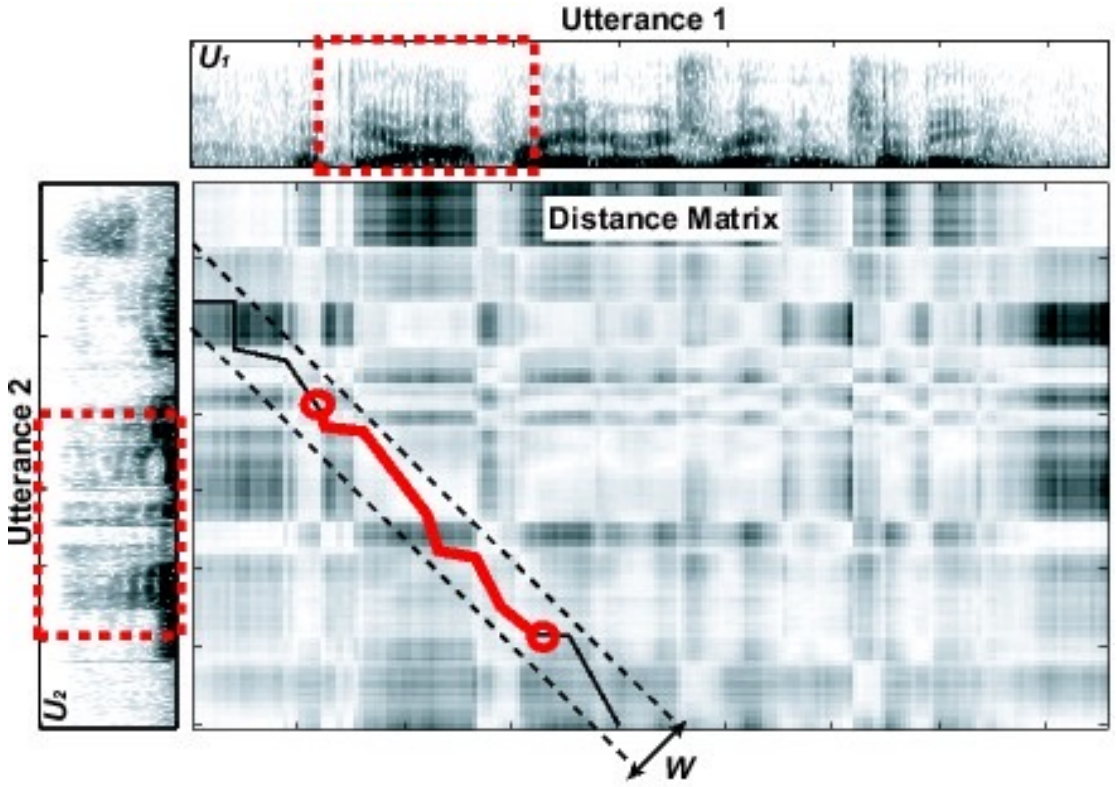
Deep learning in Speech Recognition



History of Automatic Speech Recognition

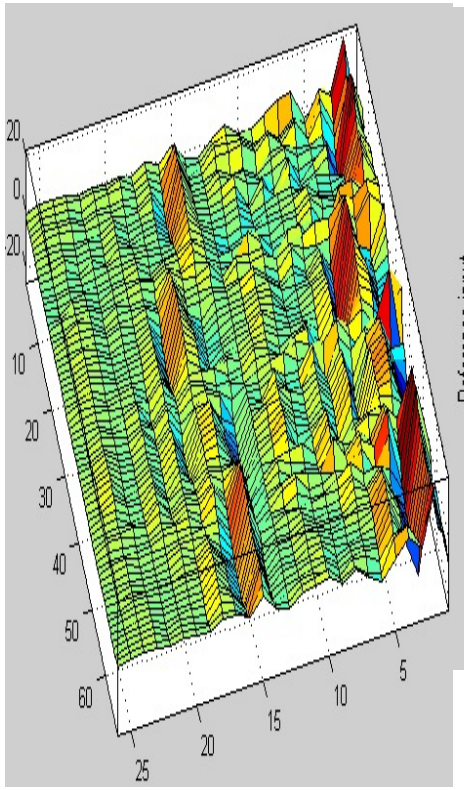
Early 1970s: Dynamic Time Warping (DTW) to handle time variability

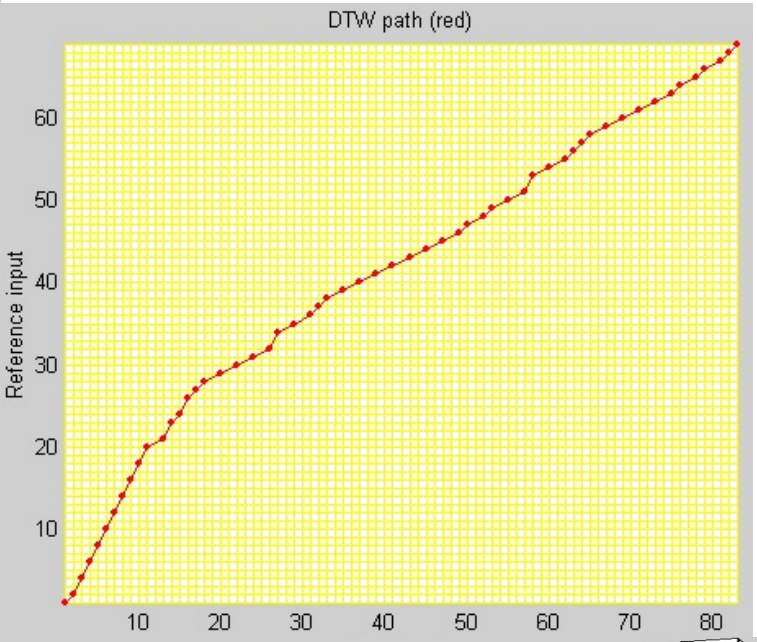
- Distance measure for spectral variability

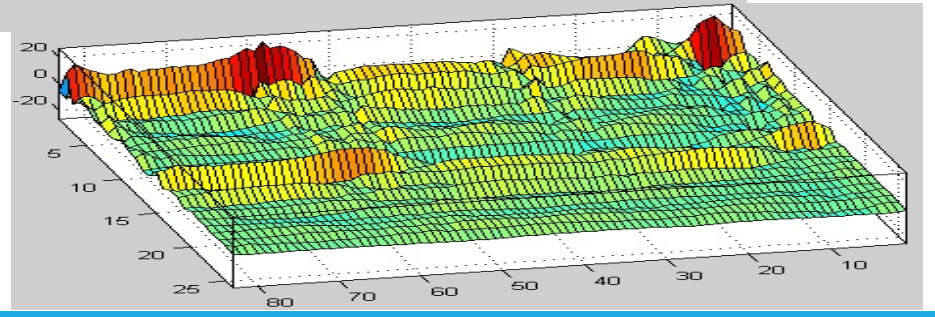


http://publications.csail.mit.edu/abstracts/abstracts06/malex/seg_dtw.jpg

Example DTW Path of "Match Ends"



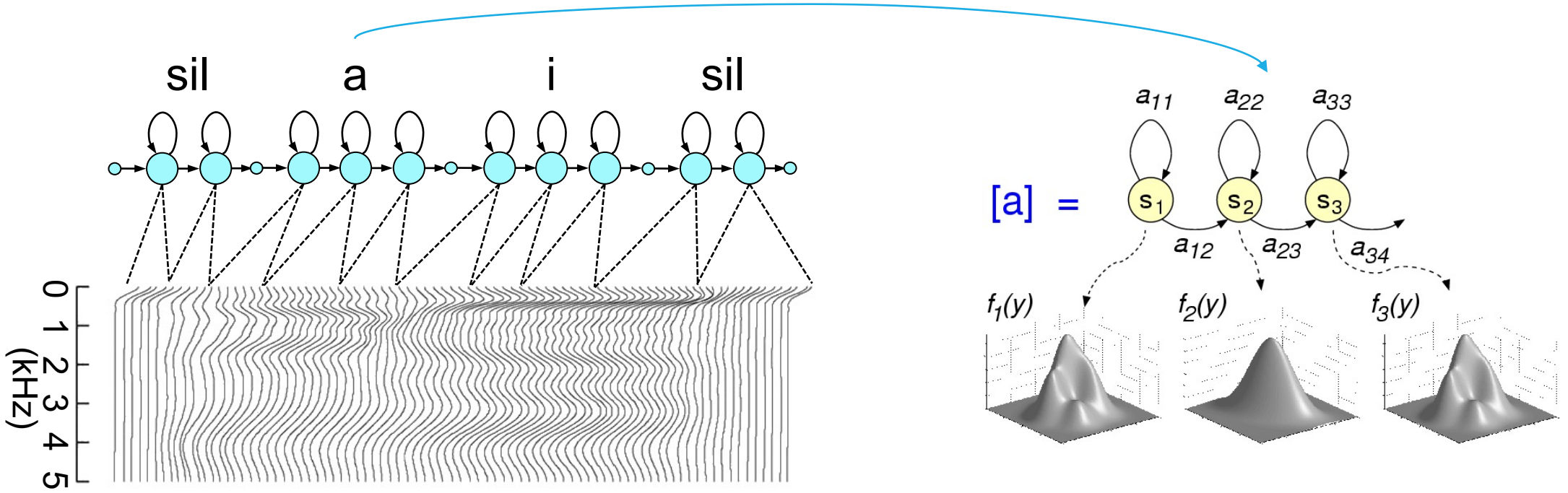




History of Automatic Speech Recognition

Mid-Late 1970s: Hidden Markov Models (HMMs) – statistical models of spectral variations, for discrete speech.

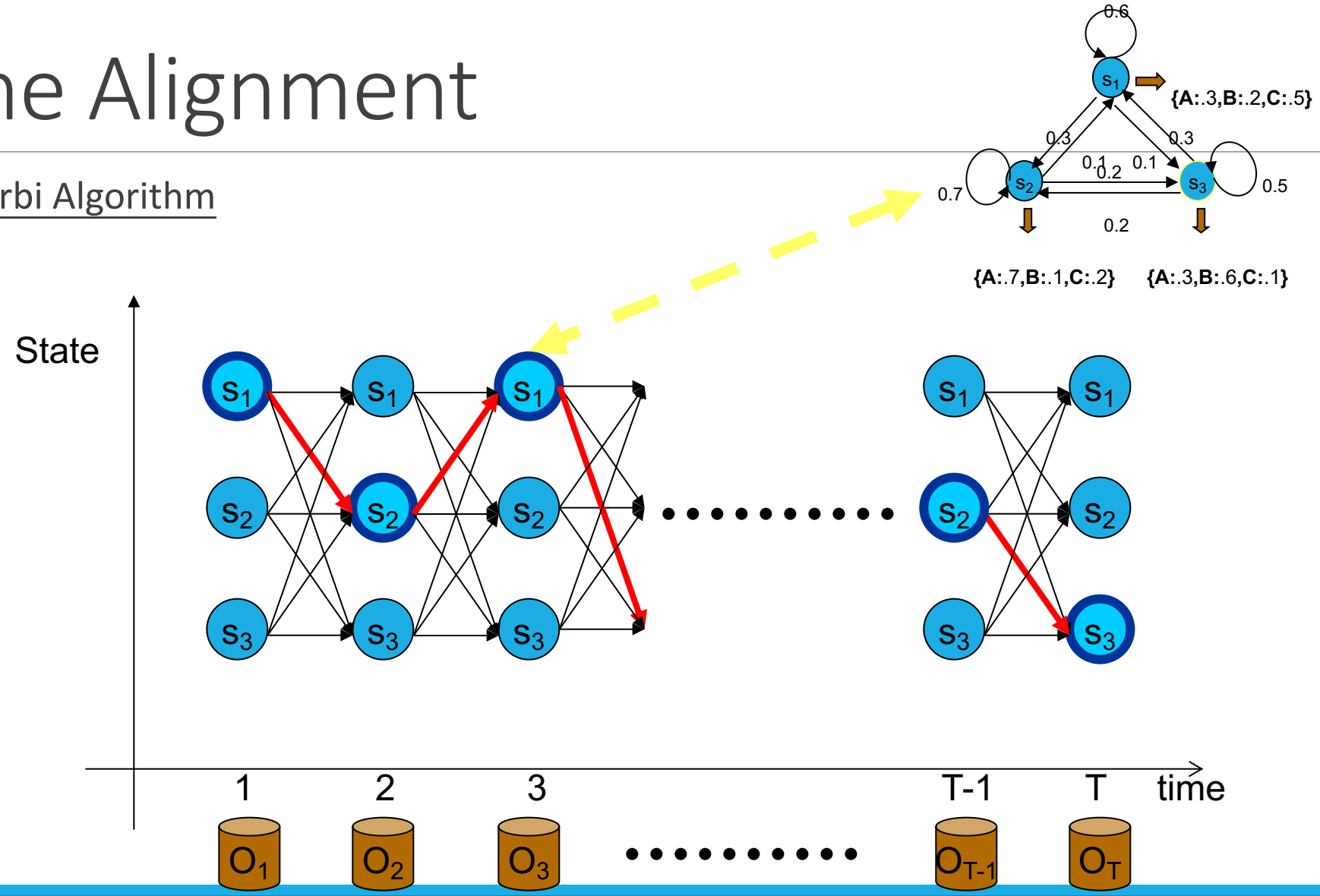
Mid 1980s: HMMs become the dominant technique for all ASR



http://hts.sp.nitech.ac.jp/archives/2.3/HTS_Slides.zip

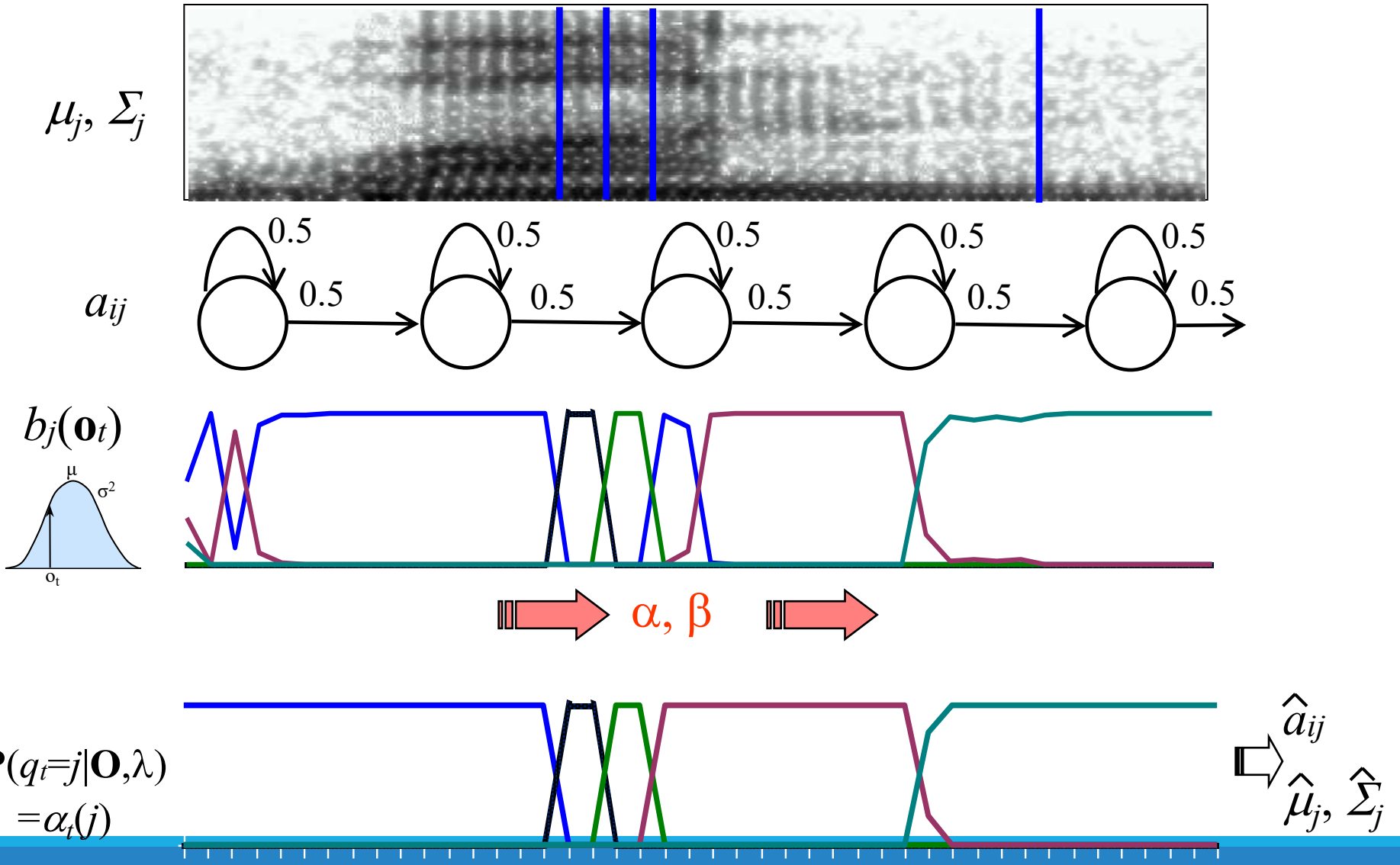
Time Alignment

The Viterbi Algorithm



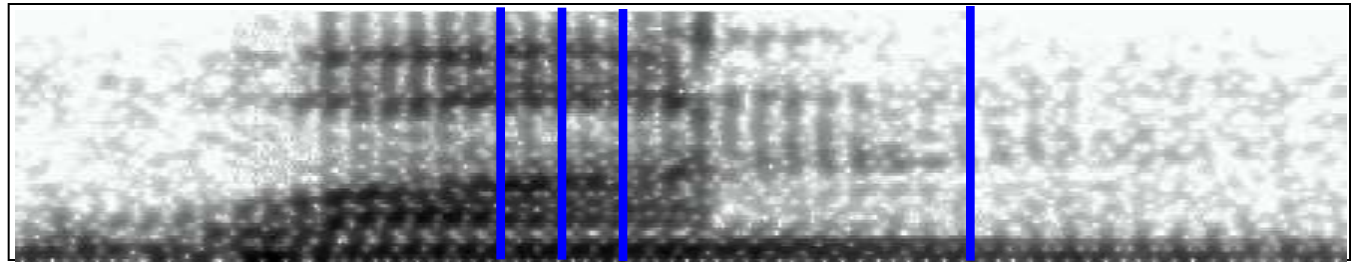
Expectation-Maximization: Forward-Backward Illustration

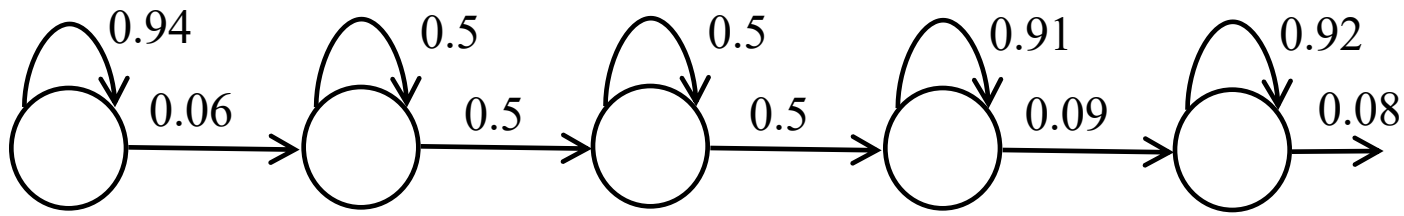
- Forward-Backward Algorithm, Iteration 1:

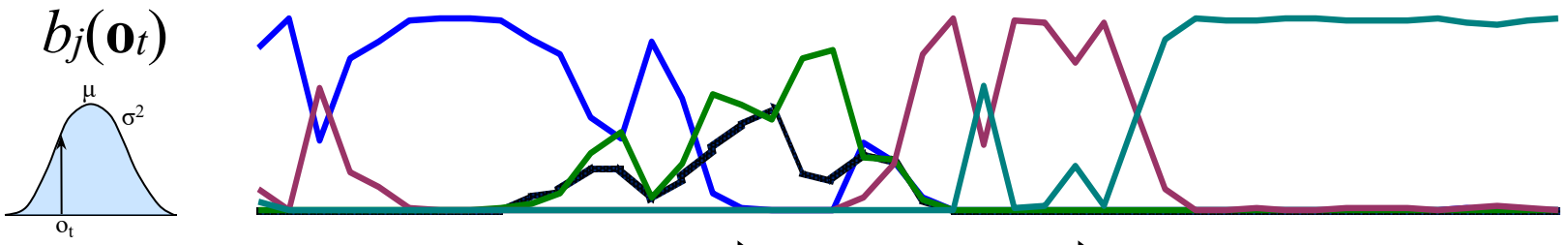


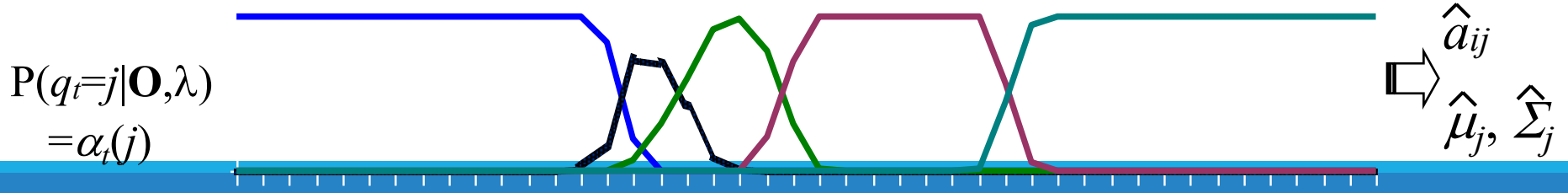
Expectation-Maximization: Forward-Backward Illustration

- Forward-Backward Algorithm, Iteration 2:



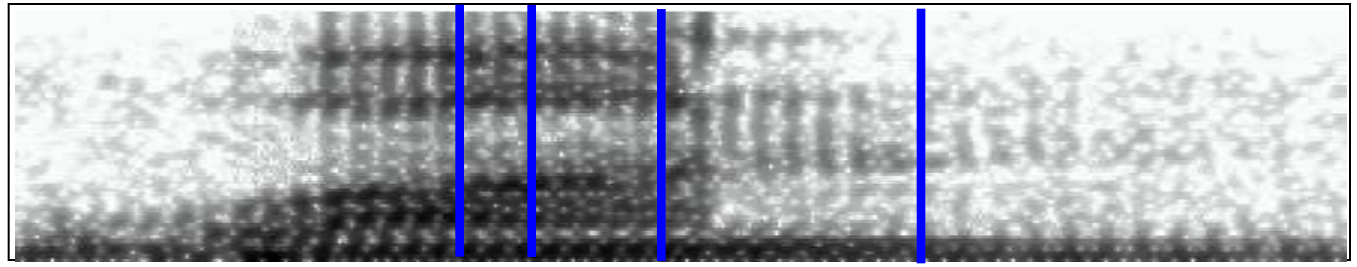


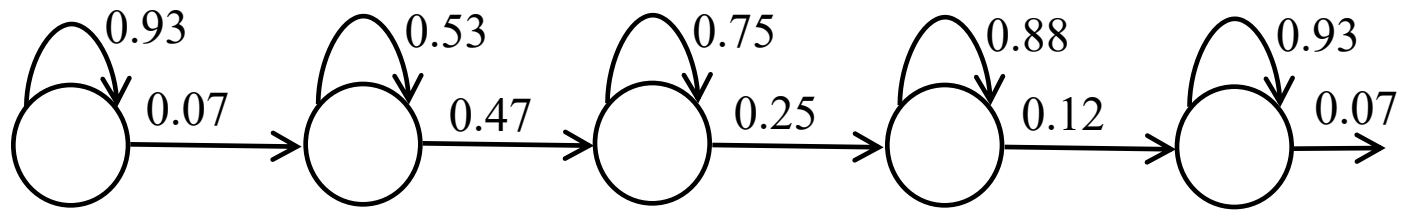


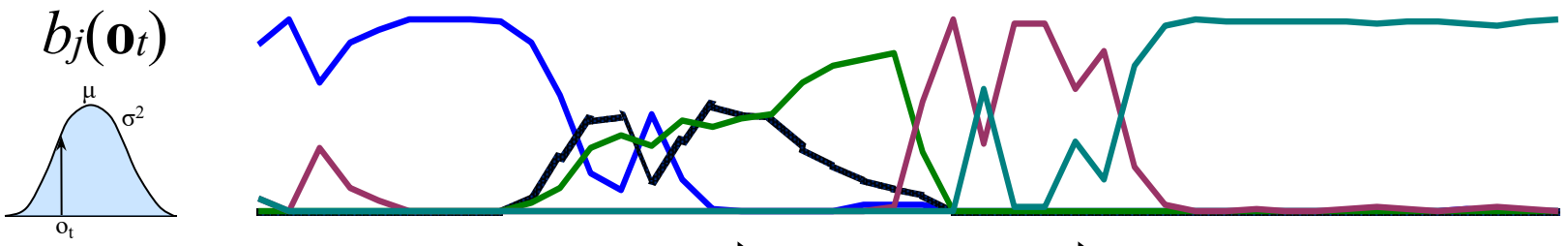


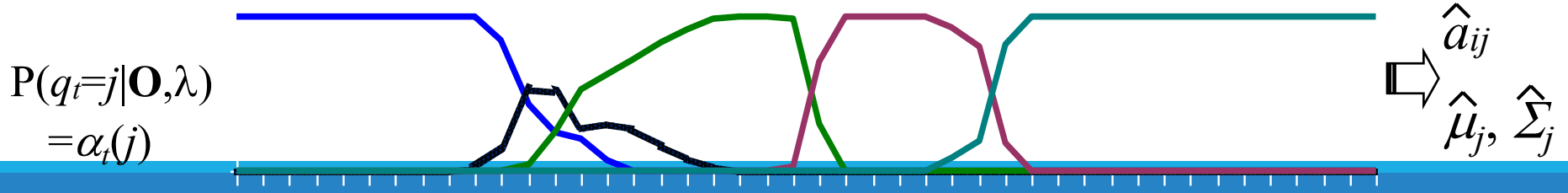
Expectation-Maximization: Forward-Backward Illustration

- Forward-Backward Algorithm, Iteration 3:



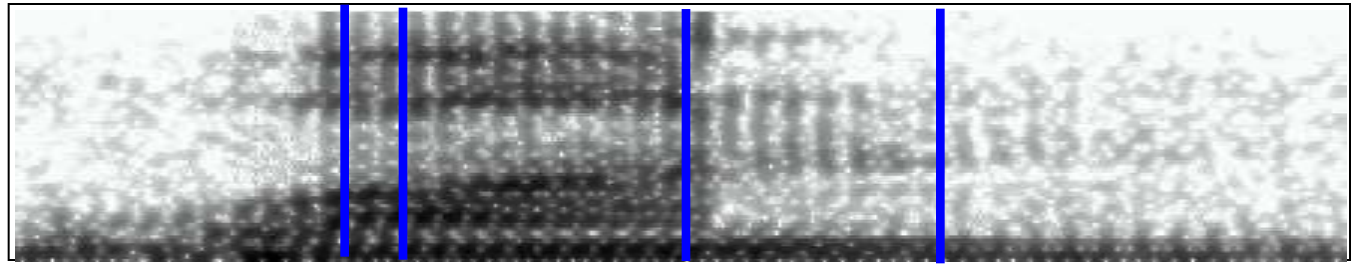




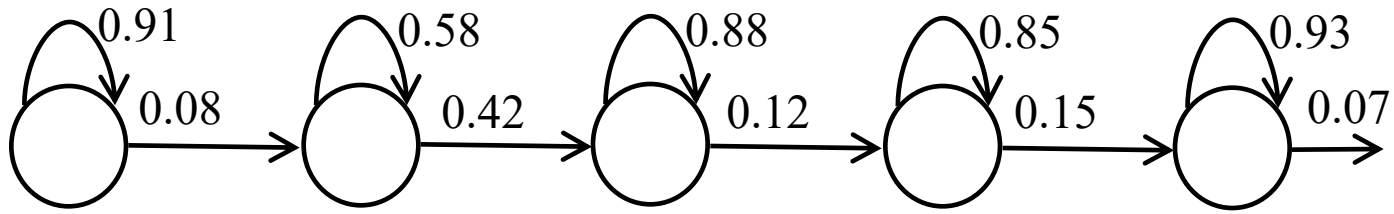


Expectation-Maximization: Forward-Backward Illustration

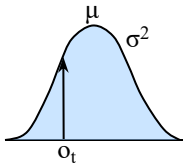
- Forward-Backward Algorithm, Iteration 4:

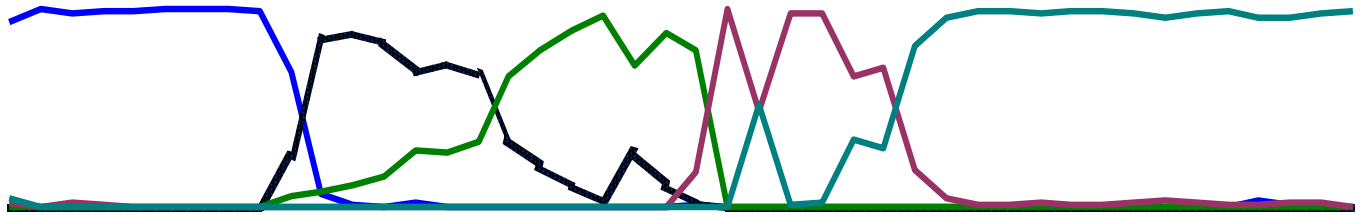


a_{ij}



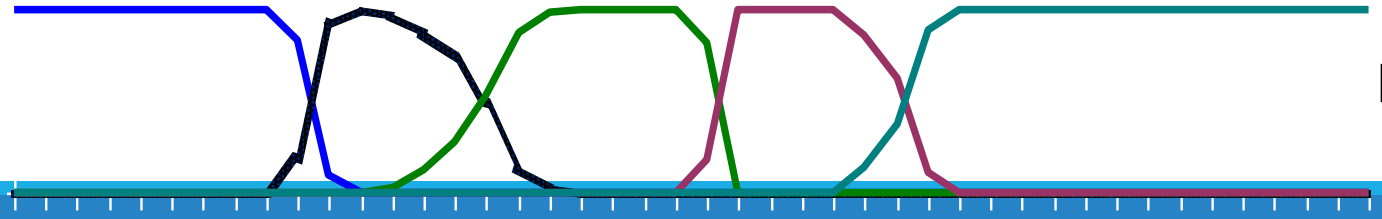
$b_j(\mathbf{o}_t)$





α, β

$P(q_t=j|\mathbf{O},\lambda) = \alpha_t(j)$

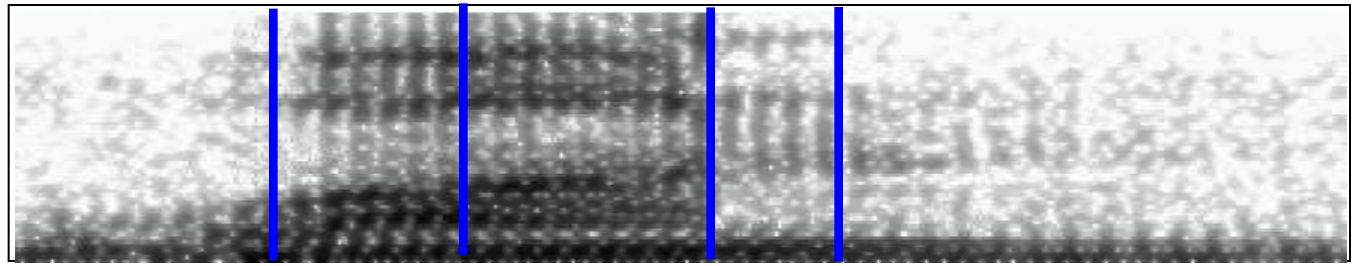


$\hat{a}_{ij}$

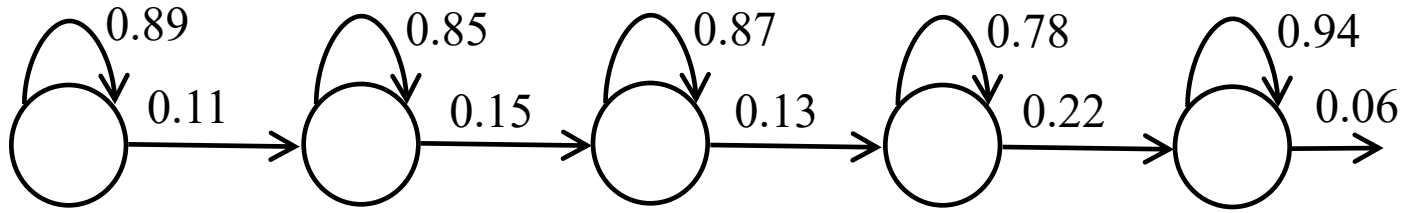
$\hat{\mu}_j, \hat{\Sigma}_j$

Expectation-Maximization: Forward-Backward Illustration

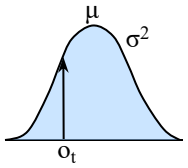
- Forward-Backward Algorithm, Iteration 10:

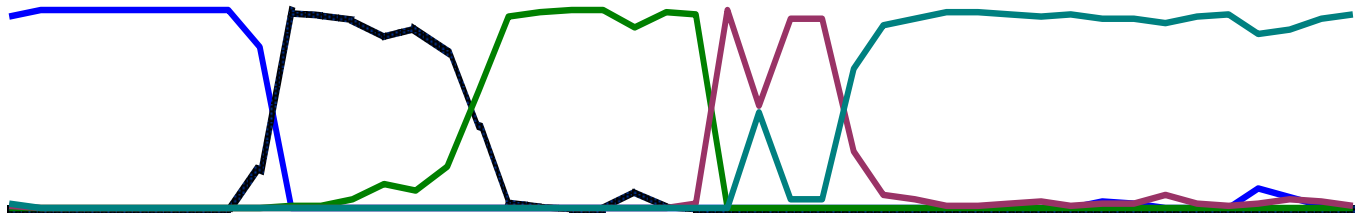


a_{ij}



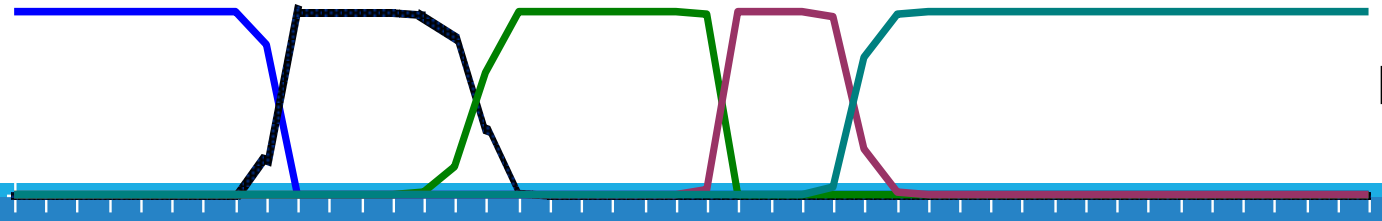
$b_j(\mathbf{o}_t)$





α, β

$P(q_t=j|\mathbf{O},\lambda) = \alpha_t(j)$

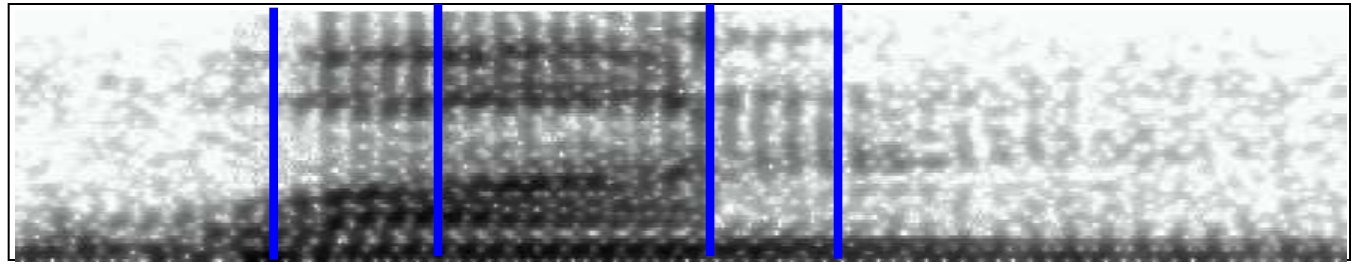


$\hat{a}_{ij}$

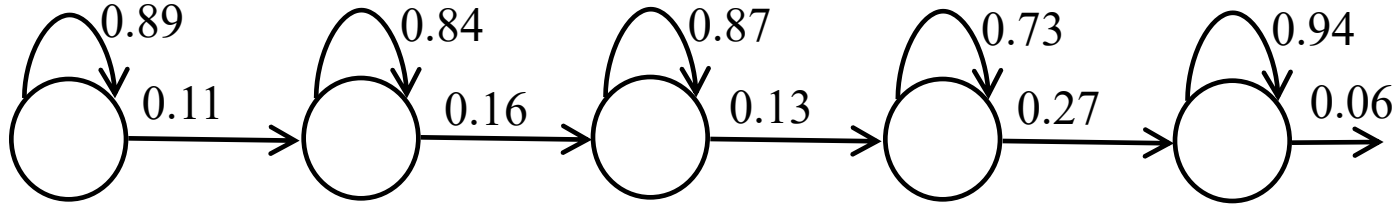
$\hat{\mu}_j, \hat{\Sigma}_j$

Expectation-Maximization: Forward-Backward Illustration

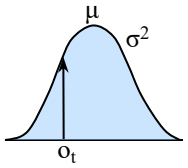
- Forward-Backward Algorithm, Iteration 20:

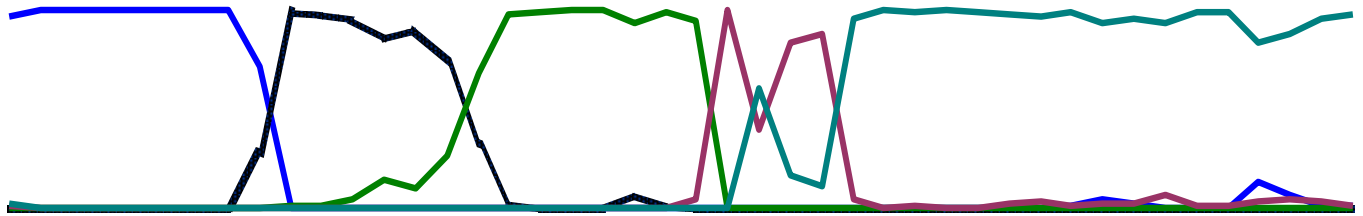


a_{ij}



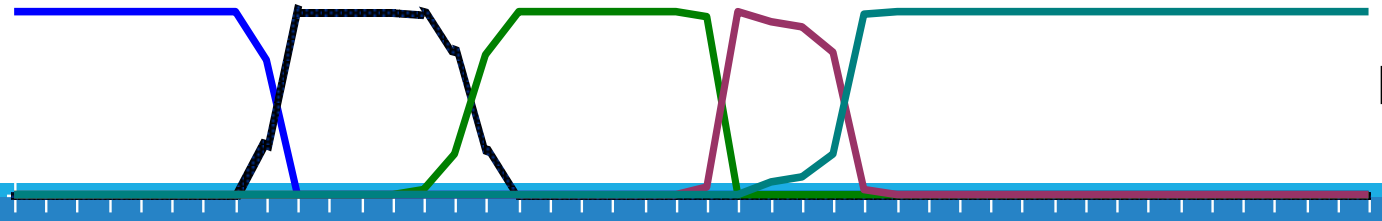
$b_j(\mathbf{o}_t)$





α, β

$P(q_t=j|\mathbf{O},\lambda) = \alpha_t(j)$



$\hat{a}_{ij}$

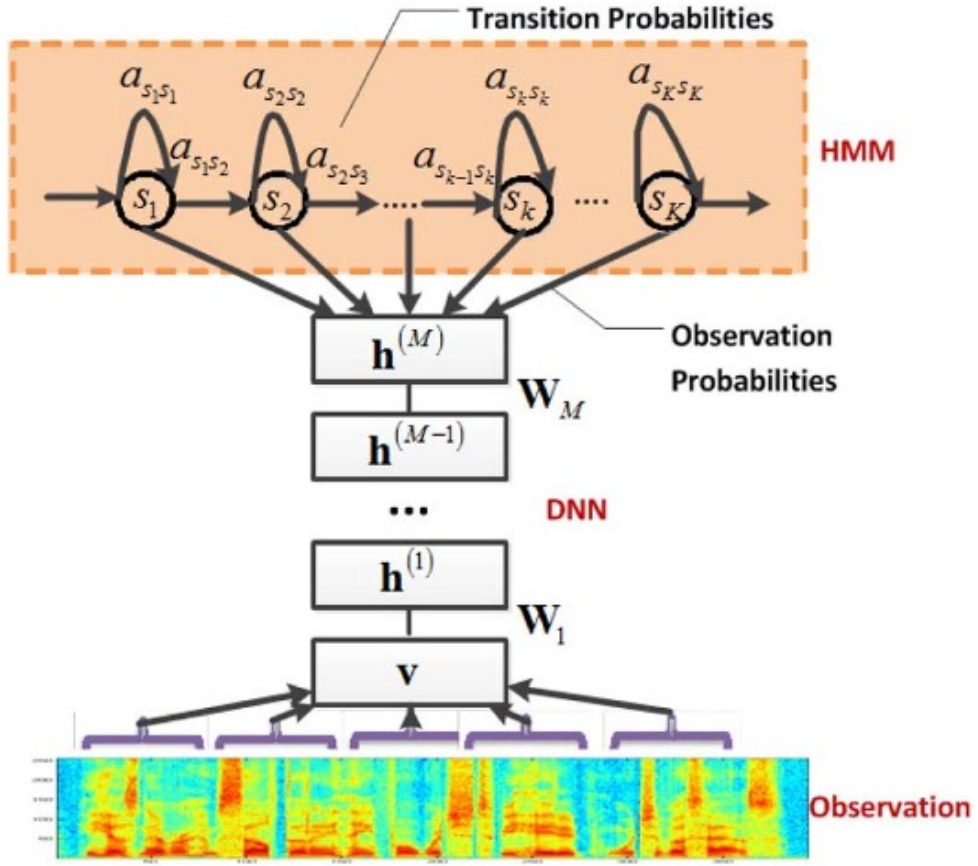
$\hat{\mu}_j, \hat{\Sigma}_j$

History of Automatic Speech Recognition

1990s: Large vocabulary continuous dictation

2000s: Discriminative training (minimize word/phone error rate)

2010s: Deep learning significantly reduce error rate



George E. Dahl, et al. Context-Dependent Pre-Trained Deep Neural Networks for Large-Vocabulary Speech Recognition. IEEE Trans. Audio, Speech & Language Processing, 2012.

Aim of Automatic Speech Recognition

Find the most likely sentence (word sequence) $\boldsymbol{W}$, which transcribes the speech audio $\boldsymbol{A}$:

$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{W}|\boldsymbol{A}) = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{A}|\boldsymbol{W})P(\boldsymbol{W})$$

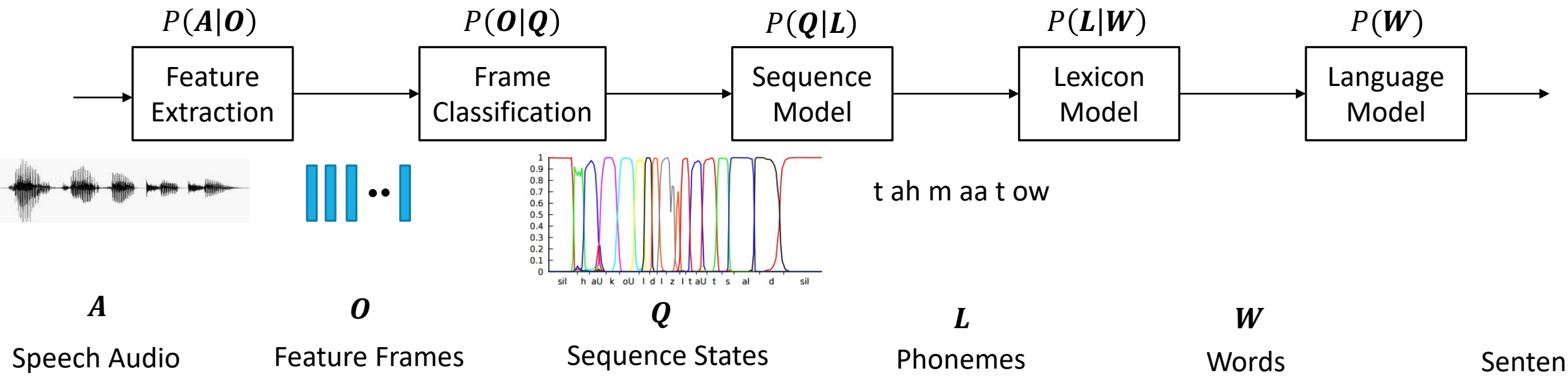
- Acoustic model $P(\boldsymbol{A}|\boldsymbol{W})$
- Language model $P(\boldsymbol{W})$

Training: find parameters for acoustic and language model separately

- Speech Corpus: speech waveform and human-annotated transcriptions
- Language model: with extra data (prefer daily expressions corpus for spontaneous speech)

Architecture of Speech Recognition

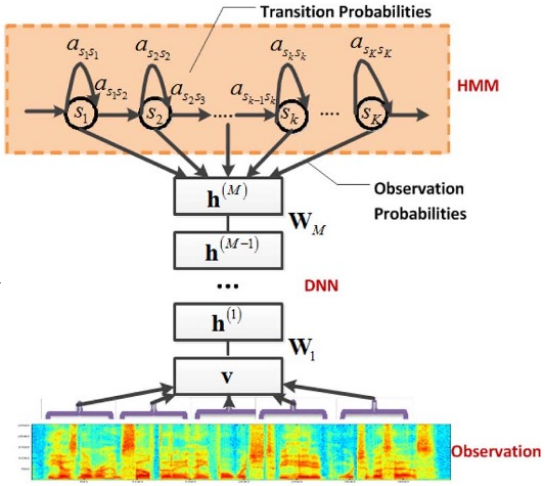
$$\widehat{\boldsymbol{W}} = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{W}|\boldsymbol{O}) = \underset{\boldsymbol{W}}{\operatorname{argmax}}\, P(\boldsymbol{A}|\boldsymbol{O})P(\boldsymbol{O}|\boldsymbol{Q})P(\boldsymbol{Q}|\boldsymbol{L})P(\boldsymbol{L}|\boldsymbol{W})P(\boldsymbol{W})$$



DNN-HMM in Speech Recognition

DNN: Deep Neural Networks

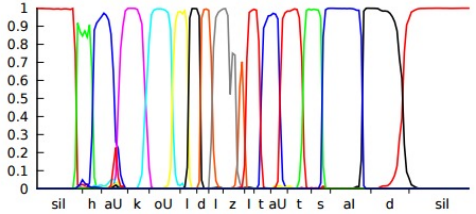
HMMs: Hidden Markov Models



DNNs

HMMs

Feature Extraction → Frame Classification → Sequence Model → Lexicon Model → Language Model



t ah m aa t ow

A	O	Q	L	W	
Speech Audio	Feature Frames	Sequence States	Phonemes	Words	Sentence

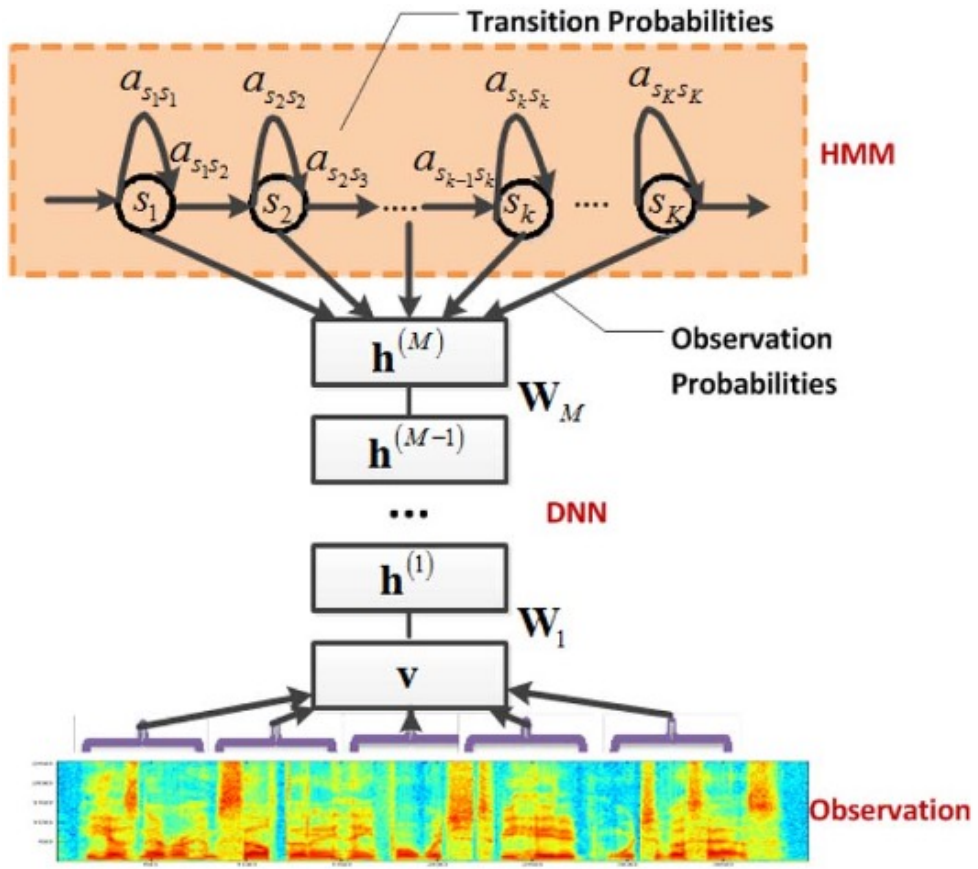
Ingredients for Deep Learning

Acoustic features

- Frequency domain features extracted from waveform
- 10ms interval between frames
- ~40 dimensions for each frame

State alignments

- Tied-state of context-dependent HMMs
- Mapping between acoustic features and states



Deep Models in HMM-based ASR

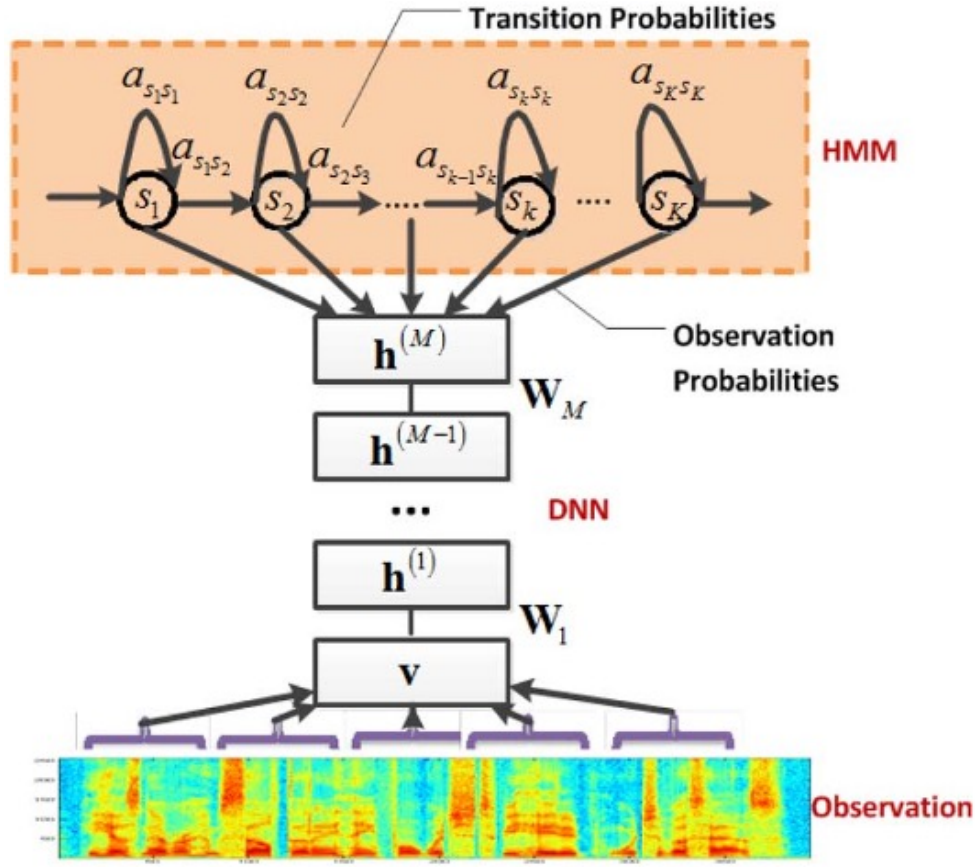
Classify acoustic features for state labels

Take softmax output as a posterior $P(state|\boldsymbol{o}_t) = P(\boldsymbol{q}_t|\boldsymbol{o}_t)$

Work as output probability in HMM

$$b_{\boldsymbol{q}}(\boldsymbol{o}_t) = b(\boldsymbol{o}_t|\boldsymbol{q}_t) = \frac{P(\boldsymbol{q}_t|\boldsymbol{o}_t)P(\boldsymbol{o}_t)}{P(\boldsymbol{q}_t)}$$

where $P(\boldsymbol{q}_t)$ is the prior probability for states



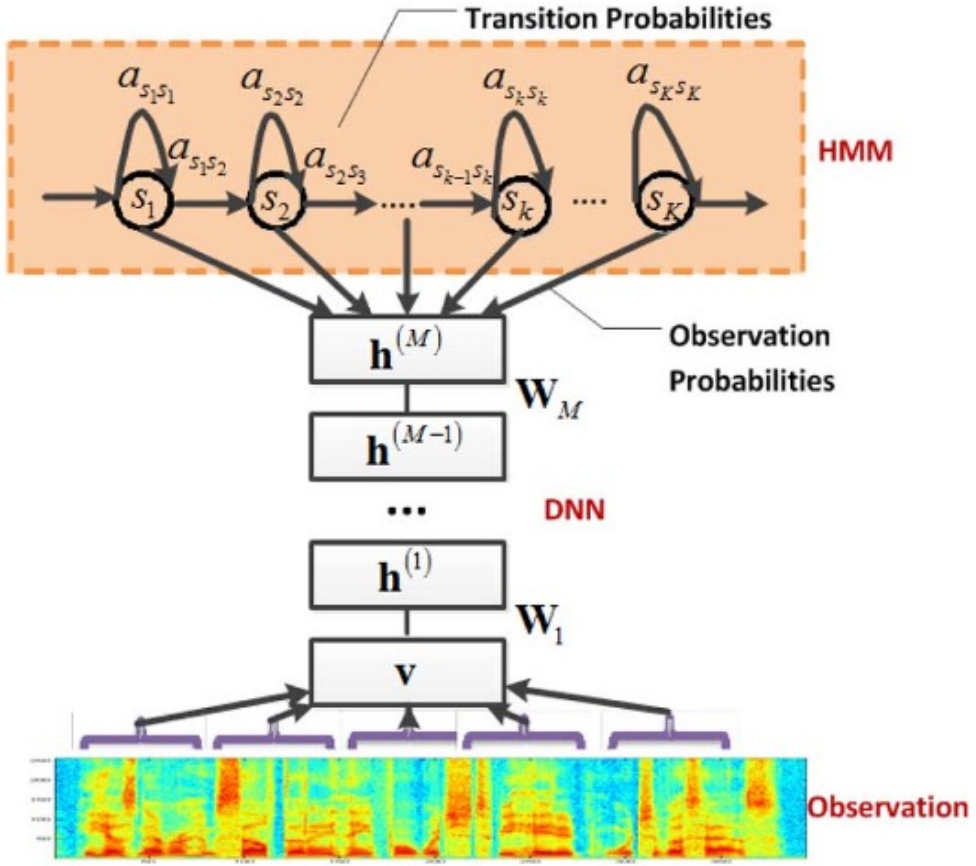
Fully Connected Networks

Features including 2 X 5 neighboring frames

- 1D convolution with kernel size 11

Classify 9,304 tied states

7 hidden layers X 2048 units with sigmoid activation



G. Hinton, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. Signal Processing Magazine (2012).

Fully Connected Networks

Comparison on different large datasets

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	

G. Hinton, et al. Deep Neural Networks for Acoustic Modeling in Speech Recognition. Signal Processing Magazine (2012).

DNN-HMM vs. GMM-HMM

Deep models are more powerful

- GMM assumes data is generated from single component of mixture model
- GMM with diagonal variance matrix ignores correlation between dimensions

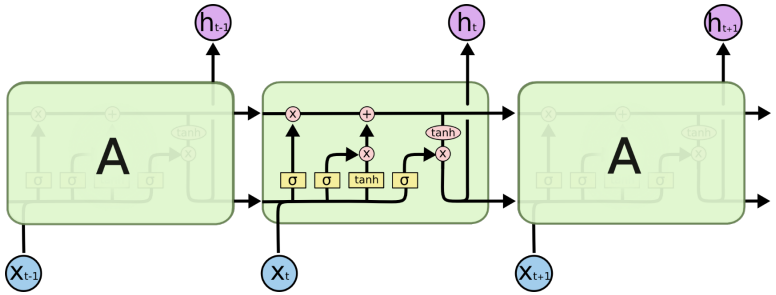
Deep models take data more efficiently

- GMM consists with many components and each learns from a small fraction of data

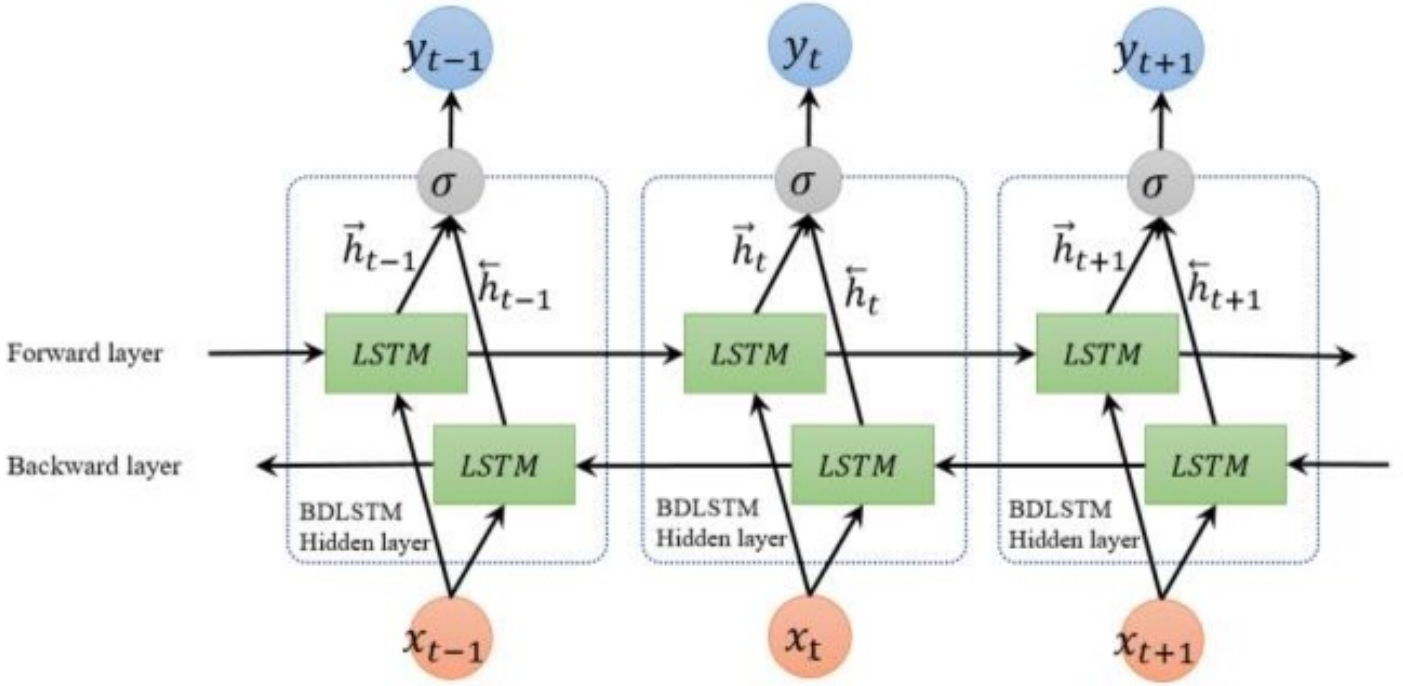
Deep models can be further improved by recent advances in deep learning

Recurrent Networks

Long Short Term Memory networks



Bi-Directional Long Short Term Memory networks



http://colah.github.io/posts/2015-08-Understanding-LSTMs/

Recurrent Networks

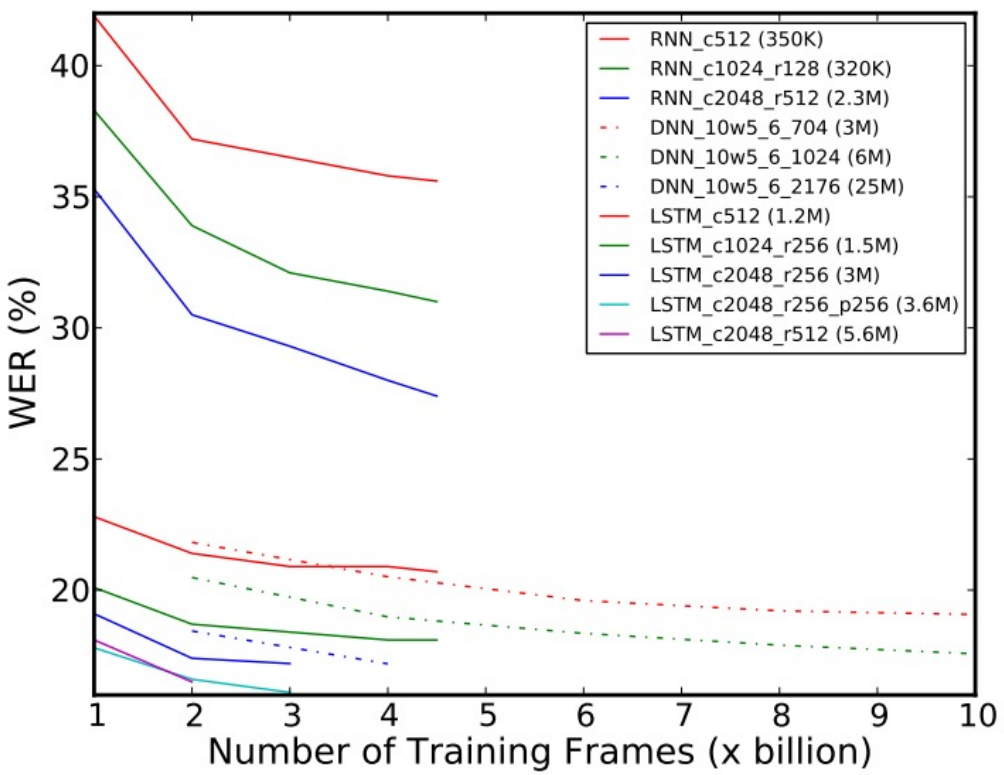


Fig. 5. 126 context independent phone HMM states.

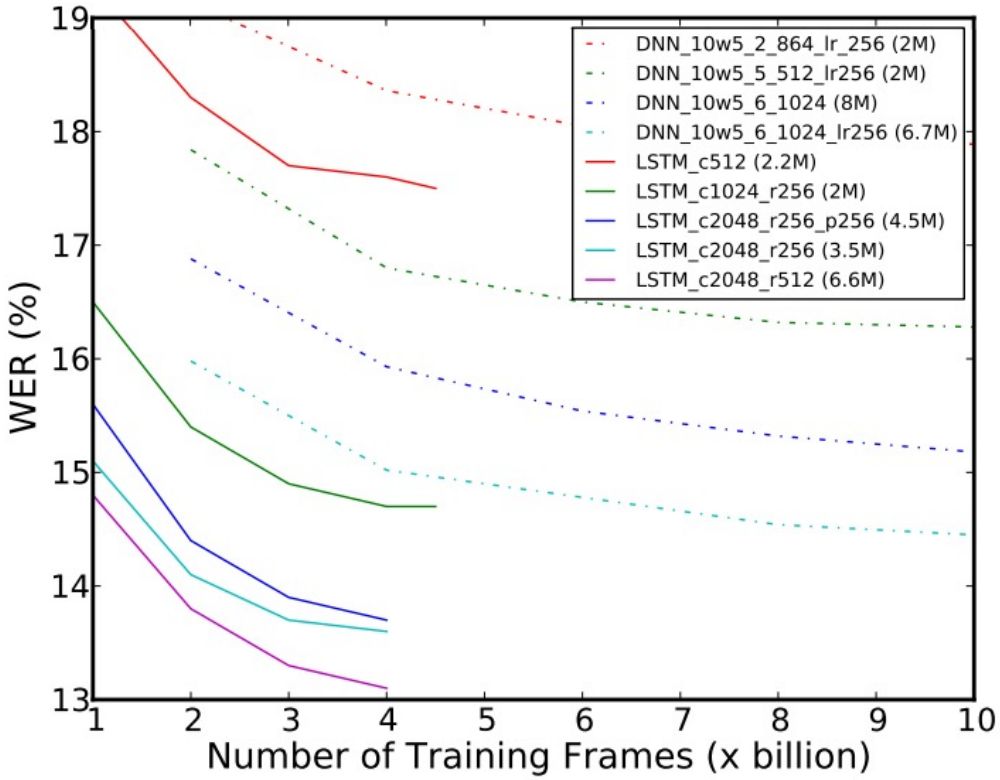
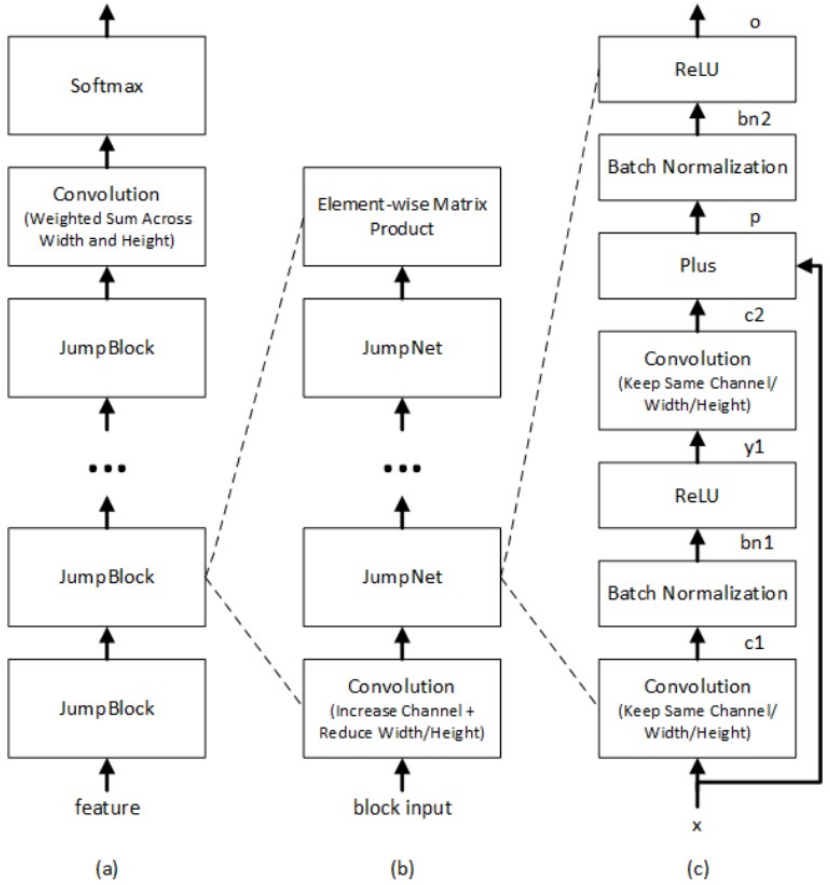


Fig. 6. 2000 context dependent phone HMM states.

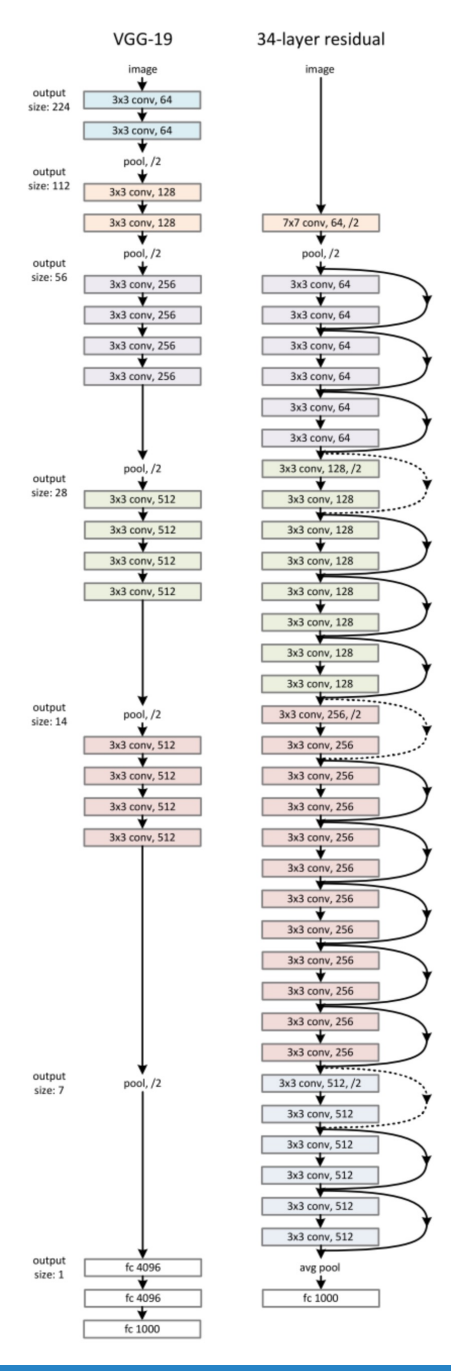
H. Sak, A. Senior, and F. Beaufays. Long Short-Term Memory Recurrent Neural Network Architectures for Large Scale Acoustic Modeling. arXiv preprint arXiv:1402.1128v1 (2014).

Very Deep Networks

LACE



VGG Net (85M Parameters)	Residual-Net (38M Parameters)	LACE (65M Parameters)
14 weight layers	49 weight layers	22 weight layers
40x41 input	40x41 input	40x61 input
3 – conv 3x3, 96	3 – [conv 1x1, 64 conv 3x3, 64 conv 1x1, 256]	5 – conv 3x3, 128
Max pool	4 – [conv 1x1, 128 conv 3x3, 128 conv 1x1, 512]	5 – conv 3x3, 256
4 – conv 3x3, 192	6 – [conv 1x1, 256 conv 3x3, 256 conv 1x1, 1024]	5 – conv 3x3, 512
Max pool	3 – [conv 1x1, 512 conv 3x3, 512 conv 1x1, 2048]	5 – conv 3x3, 1024
4 – conv 3x3, 384	Average pool	1 – conv 3x4, 1
Max pool	Softmax (9000)	Softmax (9000)
2 – FC – 4096		
Softmax (9000)		



W. Xiong, et al. Achieving Human Parity in Conversational Speech Recognition. arXiv preprint arXiv:1610.05256 (2016).

Very Deep Networks

Speaker adaptive training
- Addition speaker id embedded vector as input

Language model with LSTM

System combination
- Greedy-searched weight to combine multiple model system

Results on the test set: CH and SWB
- Exceed human accuracy

Model	N-gram LM		Neural net LM	
	CH	SWB	CH	SWB
Povey et al. [54] LSTM	15.3	8.5	-	-
Saon et al. [51] LSTM	15.1	9.0	-	-
Saon et al. [51] system	13.7	7.6	12.2	6.6
2016 Microsoft system	13.3	7.4	11.0	5.8
Human transcription			11.3	5.9

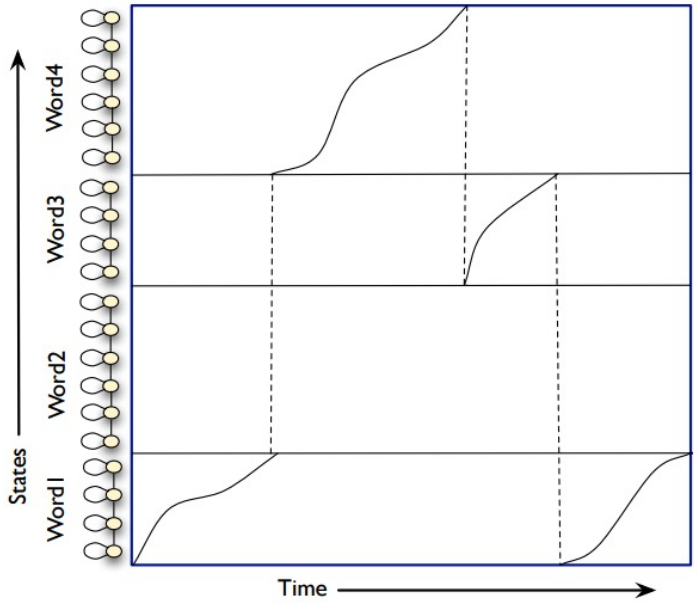
Model	N-gram LM		RNN-LM		LSTM-LM	
	CH	SWB	CH	SWB	CH	SWB
ResNet, 300h training	19.2	10.0	17.7	8.2	17.0	7.7
ResNet	14.8	8.6	13.2	6.9	12.5	6.6
ResNet, GMM alignments	15.3	8.8	13.7	7.3	12.8	6.9
VGG	15.7	9.1	14.1	7.6	13.2	7.1
VGG + ResNet	14.5	8.4	13.0	6.9	12.2	6.4
LACE	15.0	8.4	13.5	7.2	13.0	6.7
BLSTM	16.5	9.0	15.2	7.5	14.4	7.0
BLSTM, spatial smoothing	15.4	8.6	13.7	7.4	13.0	7.0
BLSTM, spatial smoothing, 27k senones	15.3	8.3	13.8	7.0	13.2	6.8
BLSTM, spatial smoothing, 27k senones, alternate dictionary	14.9	8.3	13.7	7.0	13.0	6.7
BLSTM system combination	13.2	7.3	12.1	6.4	11.6	6.0
Full system combination	13.0	7.3	11.7	6.1	**11.0**	**5.8**

Limitations of DNN-HMM

Markov models only depend one previous states

History is discretely represented by 10k states

Decoding is slow to keep all 10k state in dynamic programing



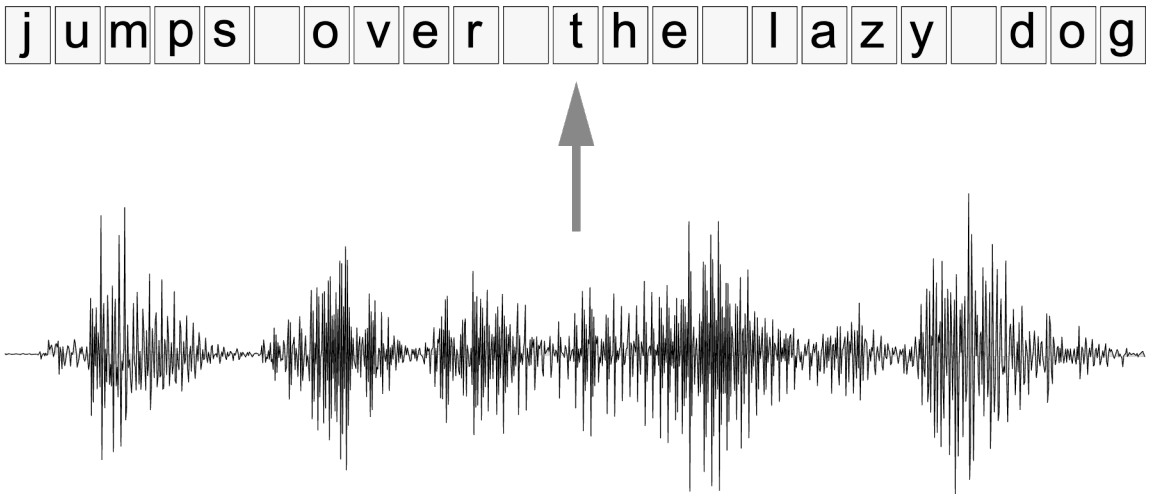
Decoding with dynamic programing

End-to-End Deep Models based Automatic Speech Recognition

Connectionist Temporal Classification (CTC) based models

- LSTM CTC models
- Deep speech 2

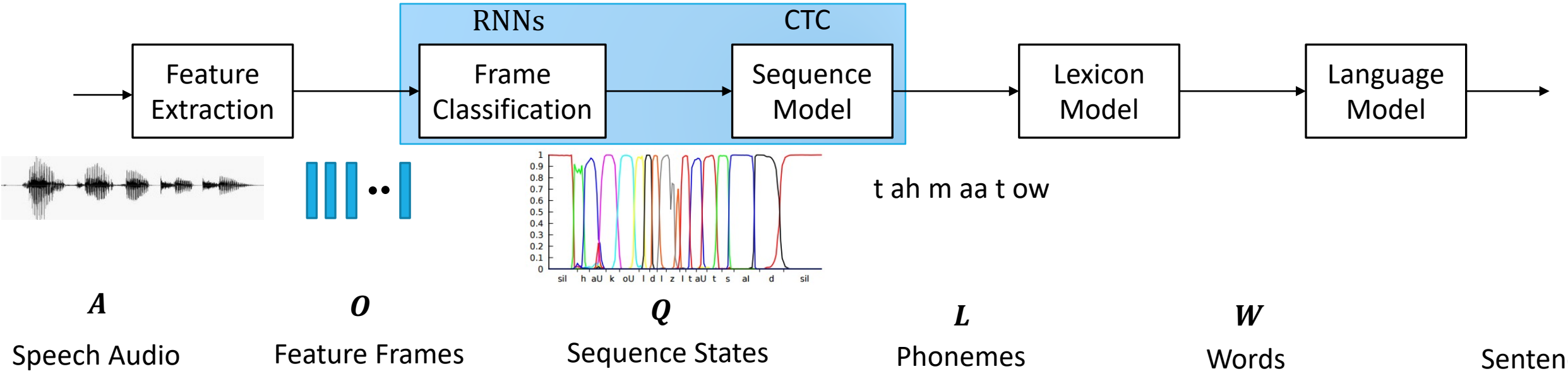
Attention based models



CTC in Speech Recognition

RNN: Recurrent Neural Networks

CTC: Connectionist Temporal Classification

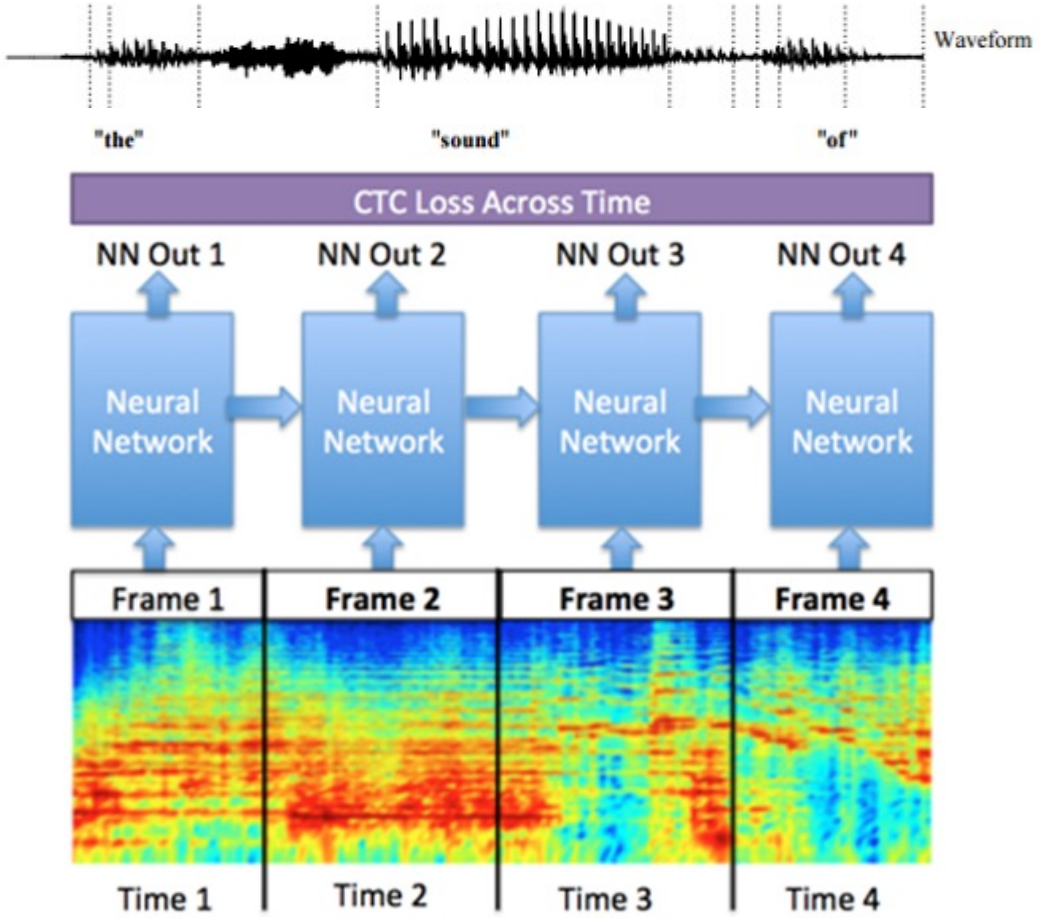


Connectionist Temporal Classification (CTC)

Method for labeling unsegmented data sequences
- Raw waveforms and text transcription

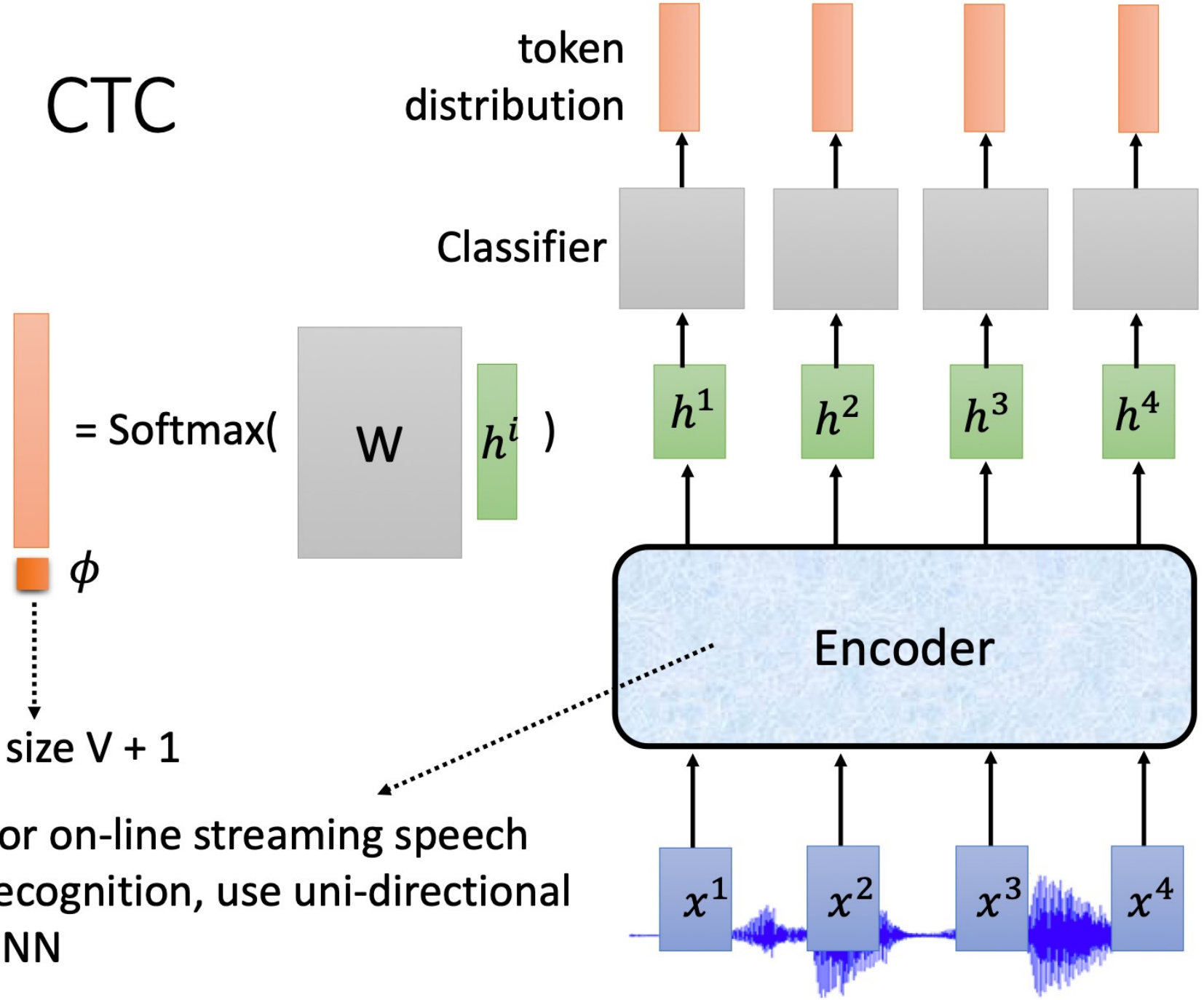
Differentiable objective function
- Gradient based training
- $O^{ML}(S, \mathcal{N}_w) = -\sum_{(x,z)\in S} \ln\big(p(z|x)\big)$
  - $S = \{(x_1, z_1), \ldots \{x_N, z_N\}) \in \mathcal{D}_{\mathcal{X}\times\mathcal{Z}}$ be the training examples
    - $x \in \mathcal{X}$ are waveforms
    - $z \in \mathcal{Z}$ are text transcripts
  - $\mathcal{N}_w(x) = \{y_k^t\}$ be NN with a softmax output

Used in various ASR and TTS architectures
- DeepSpeech (ASR)
- DeepVoice(TTS)



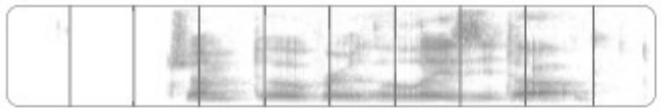
https://gab41.lab41.org/speech-recognition-you-down-with-ctc-8d3b558943f0, ftp://ftp.idsia.ch/pub/juergen/icml2006.pdf

CTC

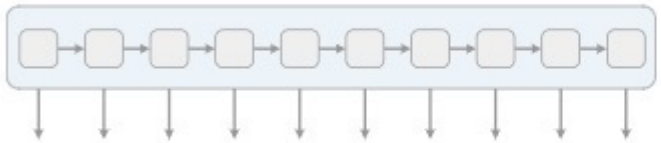


For on-line streaming speech recognition, use uni-directional RNN

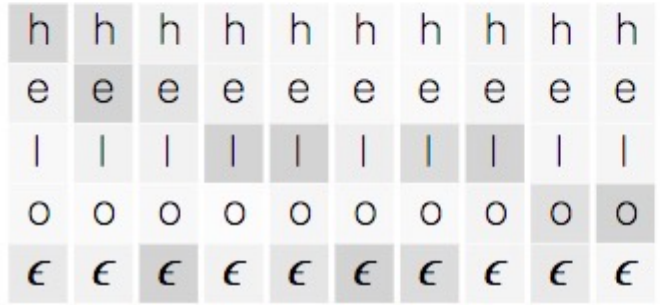
Connectionist Temporal Classification (CTC)



We start with an input sequence, like a spectrogram of audio.



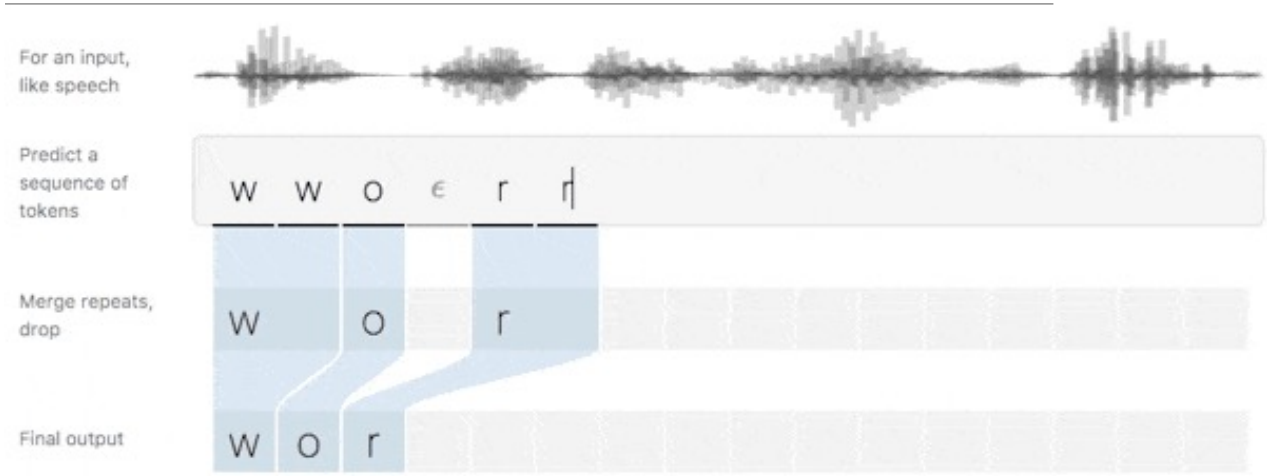
The input is fed into an RNN, for example.



The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.

With the per time-step output distribution, we compute the probability of different sequences

By marginalizing over alignments, we get a distribution over outputs



rg/speech-recognition-you-down-with-ctc-8d3b558943f0, ftp://ftp.idsia.ch/pub/juergen/icml2006.pdf

CTC Problem

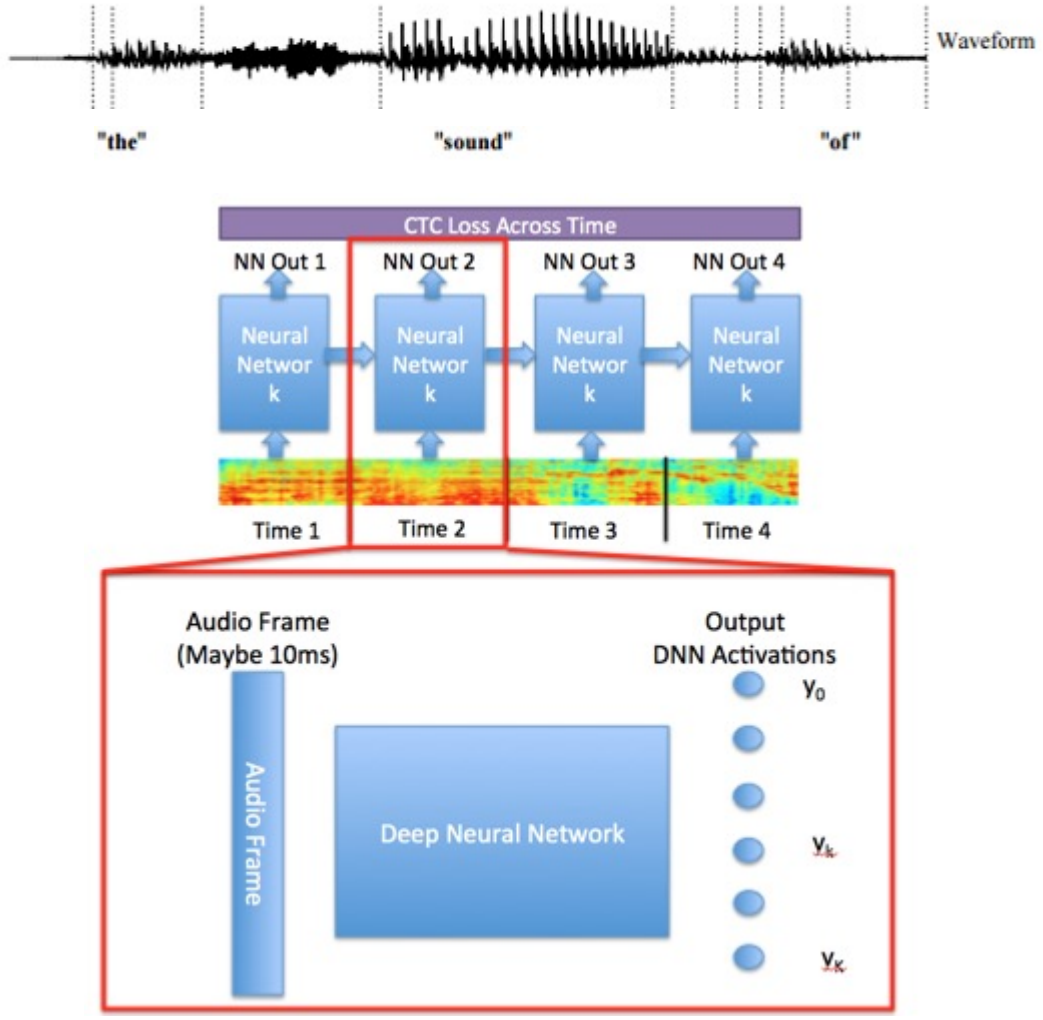
Training examples $S = \{(x_1, z_1), \ldots \{x_N, z_N\}) \in \mathcal{D}_{\mathcal{X} \times \mathcal{Z}}$

- $x \in \mathcal{X}$ are waveforms
- $z \in \mathcal{Z}$ are text transcripts

Goal: train temporal classifier $h : \mathcal{X} \rightarrow \mathcal{Z}$

Architecture

- Input layer accepts audio frame
- Some network (usually CNN or RNN)
- Softmax output over phones



CTC Description

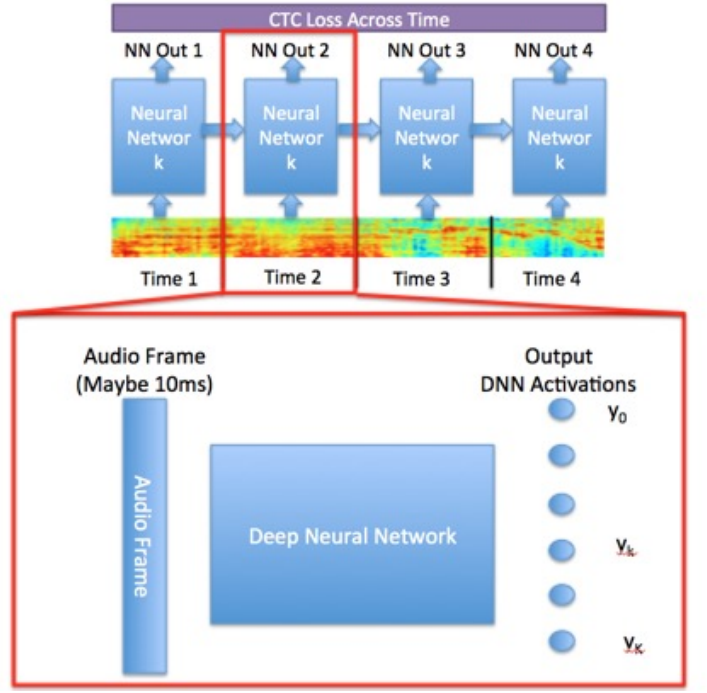
Let $\mathcal{N}_w(x) = \{y_k^t\}$ be NN with a softmax output

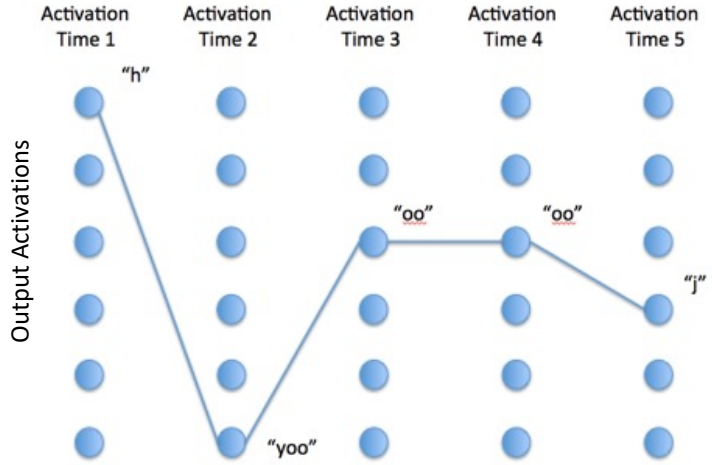
- y_k^t is activation of output unit k at time frame t
- Activations over time define distribution over L^T

Sequences over $L^T \triangleq \pi = \{\pi_1, \ldots, \pi_T\}$ are paths

Optimize for best path:

$$\mathrm{P}(\pi|x) = \prod_{t=1}^{T} y_{\pi_t}^t, \forall \pi \in L^T$$





https://gab41.lab41.org/speech-recognition-you-down-with-ctc-8d3b558943f0, ftp://ftp.idsia.ch/pub/juergen/icml2006.pdf

CTC Objective

Paths are not equivalent to the label, $\pi \neq l$

Optimize for best label:

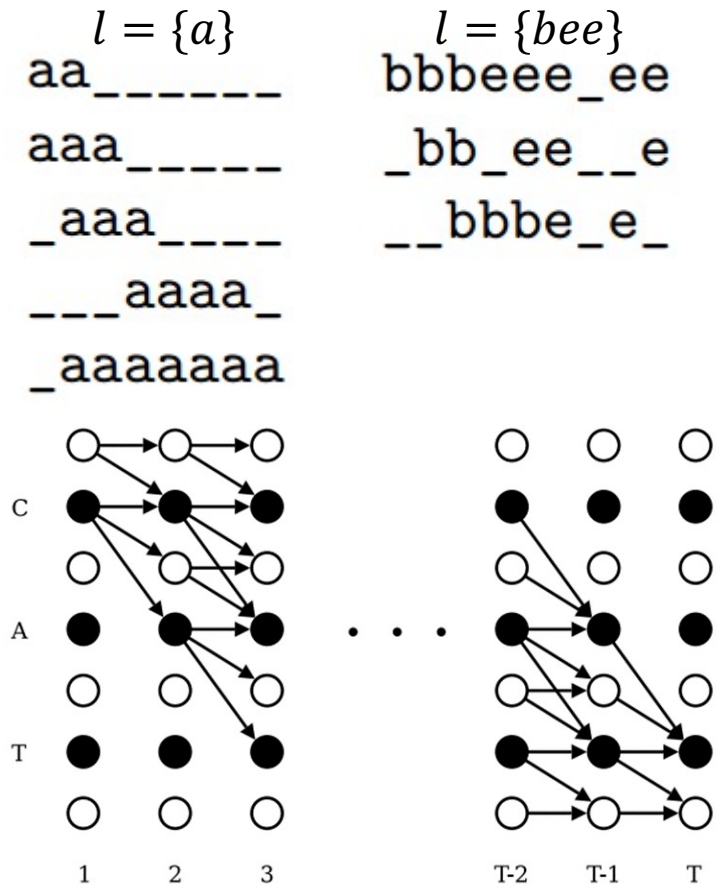
$$P(l|x) = \sum_{\pi} P(l|\pi)P(\pi|x)$$

Solve objective above with dynamic time warping

- Forward-backward algorithm
  - Forward variables α
  - Backward variables β

$$P(l|x) = \sum_{s=1}^{|l|} \frac{\alpha_t(s)\beta_t(s)}{y_{l_s}^t}$$

- Maps and searches only paths that correspond to target label



$$\frac{\partial O^{ML}(\{(\mathbf{x},\mathbf{z})\},\mathcal{N}_w)}{\partial y_k^t} = -\frac{\partial ln(p(\mathbf{z}|\mathbf{x}))}{\partial y_k^t} \rightarrow \frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y_k^t} = \frac{1}{{y_k^t}^2} \sum_{s\in lab(\mathbf{l},k)} \alpha_t(s)\beta_t(s) \rightarrow \frac{\partial ln(p(\mathbf{l}|\mathbf{x}))}{\partial y_k^t} = \frac{1}{p(\mathbf{l}|\mathbf{x})} \frac{\partial p(\mathbf{l}|\mathbf{x})}{\partial y_k^t}$$

CTC Objective and Gradient

unnormalised outputs u_{tk}.

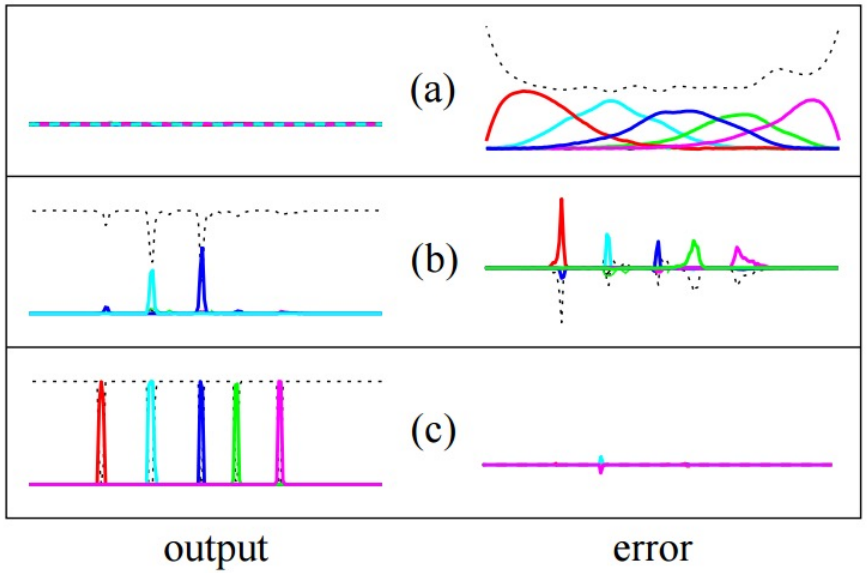
Objective function:

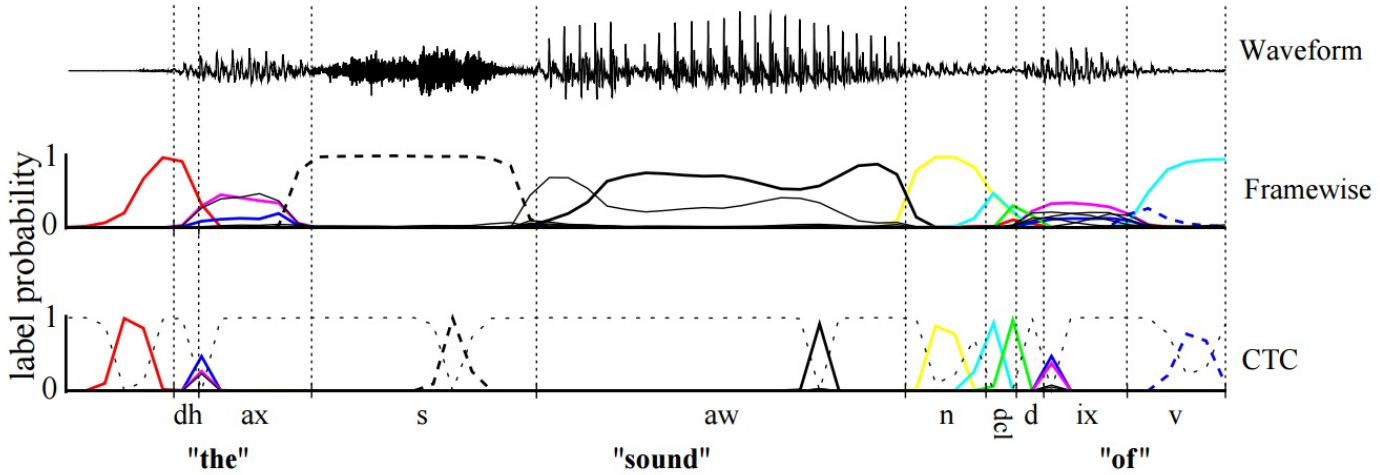
$$O^{ML}(S,\mathcal{N}_w) = -\sum_{(x,z)\in S} \ln(p(z|x))$$

Gradient:

$$\frac{\partial O^{ML}(\{(x,z)\},\mathcal{N}_w)}{\partial u_k^t} = y_k^t - \frac{1}{y_k^t Z_t} \sum_{s\in lab(z,k)} \hat{\alpha}_t(s)\hat{\beta}_t(s)$$

where $Z_t \triangleq \sum_{s=1}^{|l'|} \frac{\alpha_t(s)\beta_t(s)}{y_{l'_s}^t}$





LSTM CTC Models

Word error rate compared with HMM based models

	Context	Error Rate (%)
LSTM-HMM	Uni	8.9
	Bi	9.1
LSTM-CTC	Uni	9.4
	Bi	8.5

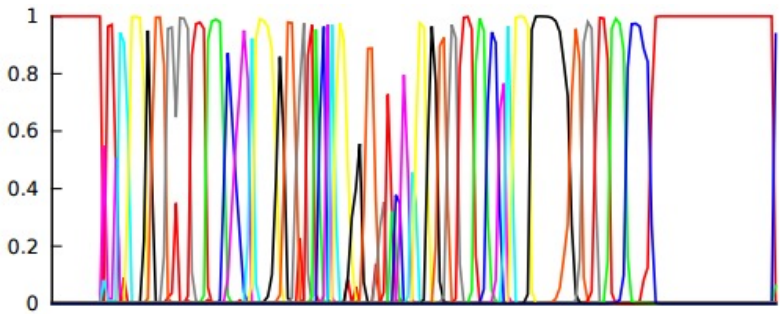
Bi-direction LSTM is more essential for CTC

Sak, Haşim, et al. "Learning acoustic frame labeling for speech recognition with recurrent neural networks." ICASSP, 2015.

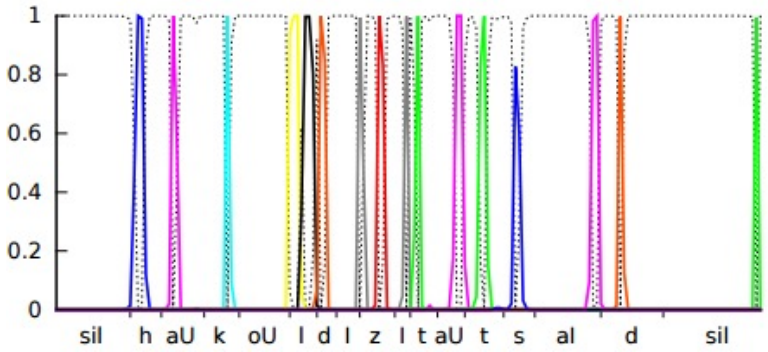
CTC vs HMM

Output probability of LSTM

HMM



CTC



CTC embeds history in continuous hidden space, more capable than HMM

CTC has spiky predictions, more discriminable between states than HMM

CTC with less states (40) is significantly faster for decoding than HMM (10k states)

Sak, Haşim, et al. "Learning acoustic frame labeling for speech recognition with recurrent neural networks." ICASSP, 2015.

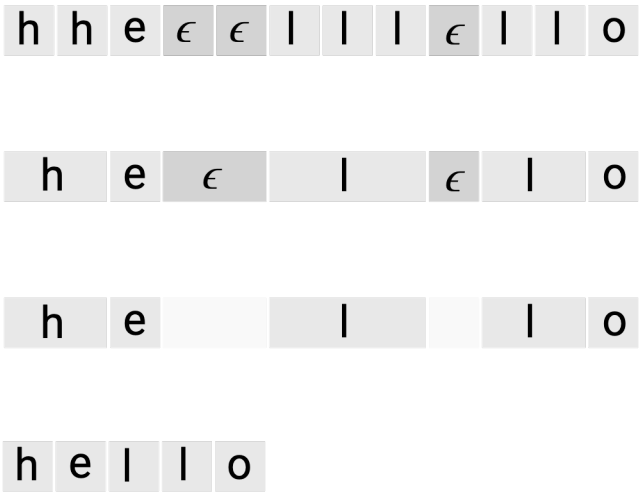
CTC Algorithm (again)

CTC algorithm can assign a probability for any *Y* given an *X*.

- Alignment free - doesn't require an alignment between the input and the output
- However, CTC has to sum over the probability of all possible alignments between the two

Approach

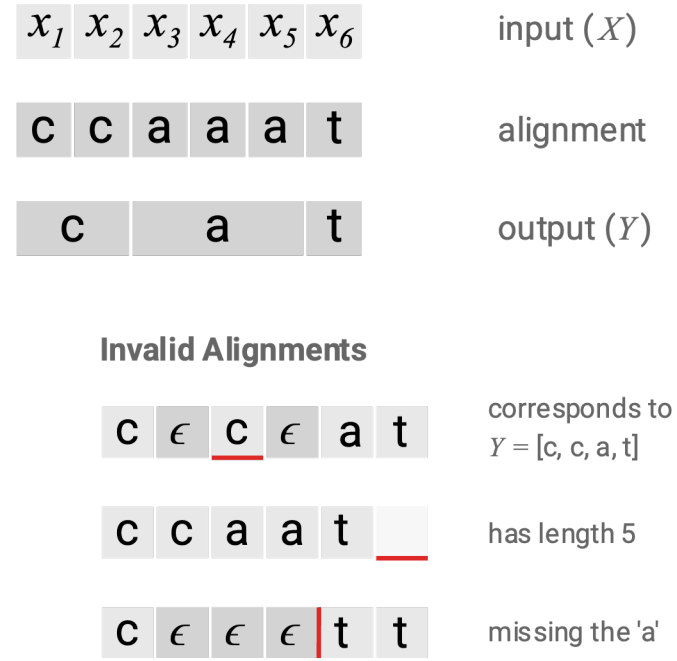
- assign an output character to each input step (change)
- introduces a "blank" token to indicate "no change"
- and collapse repeats.



First, merge repeat characters.

Then, remove any ϵ tokens.

The remaining characters are the output.



CTC Loss Function

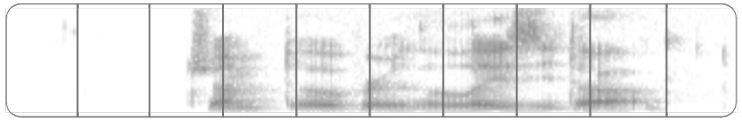
Objective for a single *(X,Y)* pair

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

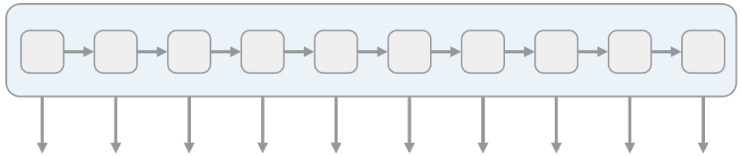
The CTC conditional **probability**

marginalizes over the set of valid alignments

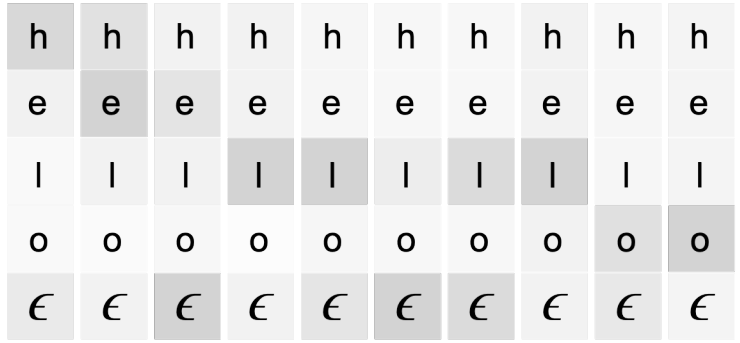
computing the **probability** for a single alignment step-by-step.



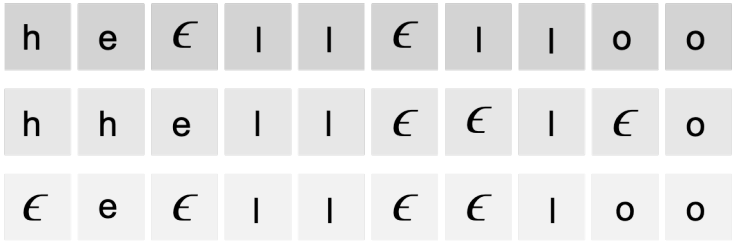
We start with an input sequence, like a spectrogram of audio.



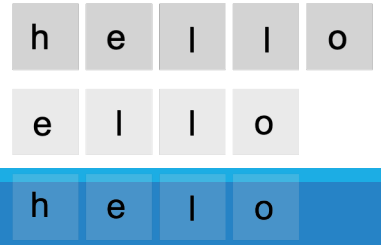
The input is fed into an RNN, for example.



The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.



With the per time-step output distribution, we compute the probability of different sequences



By marginalizing over alignments, we get a distribution over outputs.

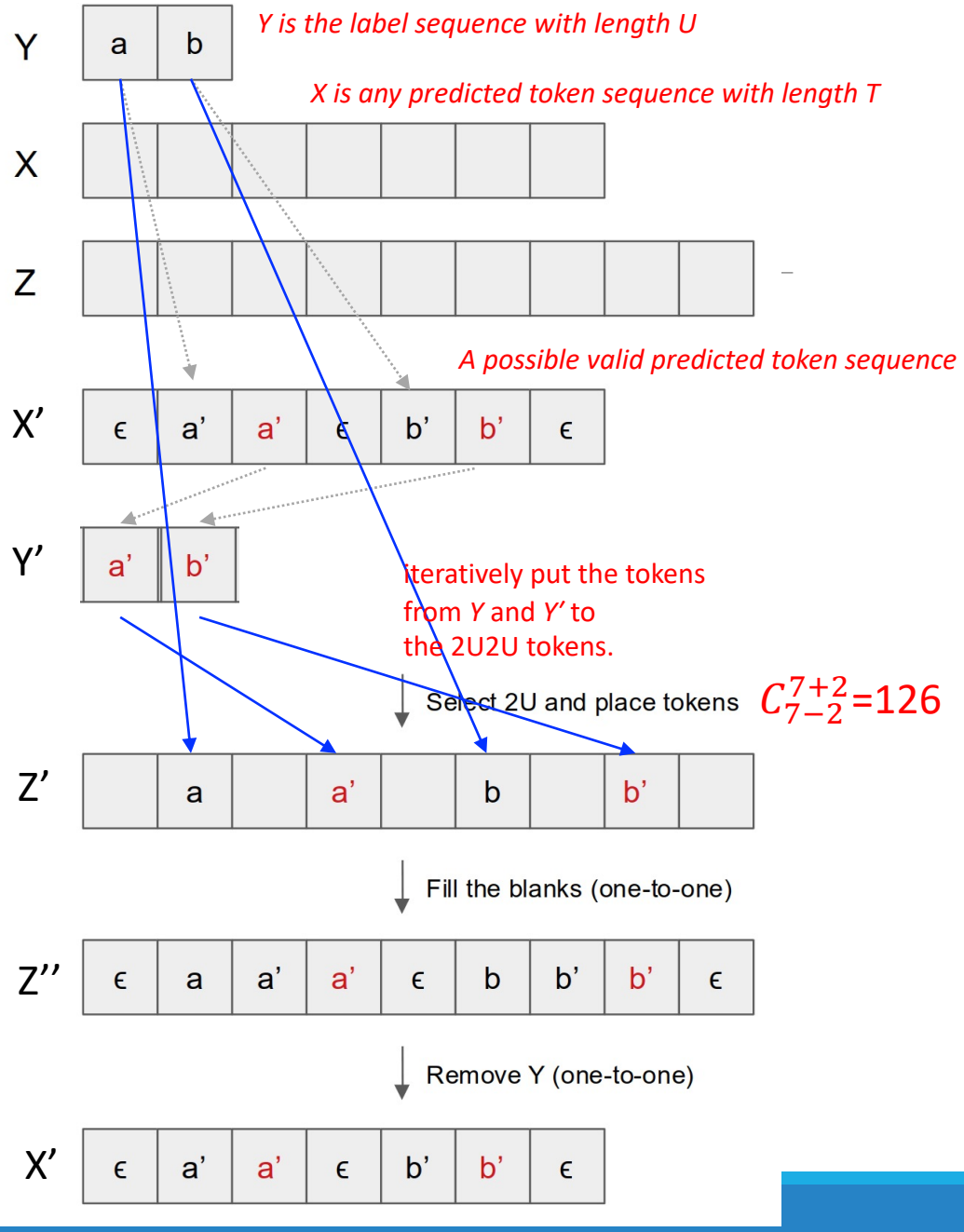
Number of Possible Alignments in CTC

All valid CTC alignments is C_{T-U}^{T+U}

- for a *Y* of length *U* without any repeat characters and *X* of length *T*, assuming *T*≥*U*

Example

- C_{7-2}^{7+2}=126
- "Hello world" ≅ C_{100-11}^{100+11}= 9.5013e+22 !



CTC vs. HMM

CTC state diagram as a special case HMM

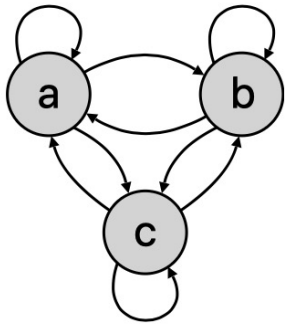
- Incorporating the blank as a hidden state in the HMM allows us to use the alphabet of *Y* as the other hidden states

CTC's shortcomings ➔ conditional independence assumption (as in HMM)

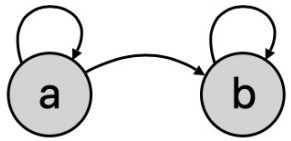
$$p(Y \mid X) \propto p(X \mid Y)\, p(Y)$$

However, CTC is discriminative ➔ It models p(Y|X)p(Y|X) directly

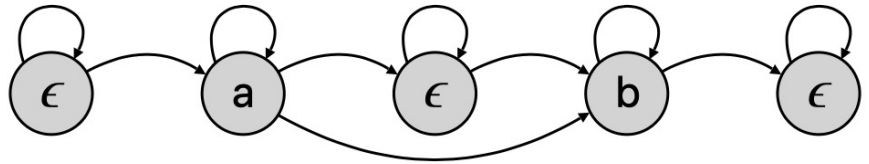
$$p(Y \mid X) \propto p(X \mid Y)\, p(Y)$$



Ergodic HMM: Any node can be either a starting or final state.



Linear HMM: Start on the left, end on the right.



CTC HMM: The first two nodes are the starting states and the last two nodes are the final states.

CTC vs. HMM

$$p(Y \mid X) \propto p(X \mid Y)\, p(Y).$$

Considering all allowed alignments between *X* and *Y* ➔

$$p(X \mid Y) = \sum_{A \in \mathcal{A}} p(X, A \mid Y)$$

Conditionally independent assumption ➔ (remove the conditioning on *Y*)

$$p(X) = \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(x_t \mid a_t) \cdot p(a_t \mid a_{t-1})$$

- $p(X)$: The probability of the input
- $\sum_{A \in \mathcal{A}}$: Marginalizes over alignments
- $p(x_t \mid a_t)$: The emission probability
- $p(a_t \mid a_{t-1})$: The transition probability

assume $p(a_t/a_{t-1})$ are uniform

$$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(x_t \mid a_t)$$

Bayes' rule

$$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} \frac{p(a_t \mid x_t)\, p(x_t)}{p(a_t)}$$

$$\propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} \frac{p(a_t \mid x_t)}{p(a_t)}.$$

assume $p(a_t)$ are uniform

$$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(a_t \mid X)$$

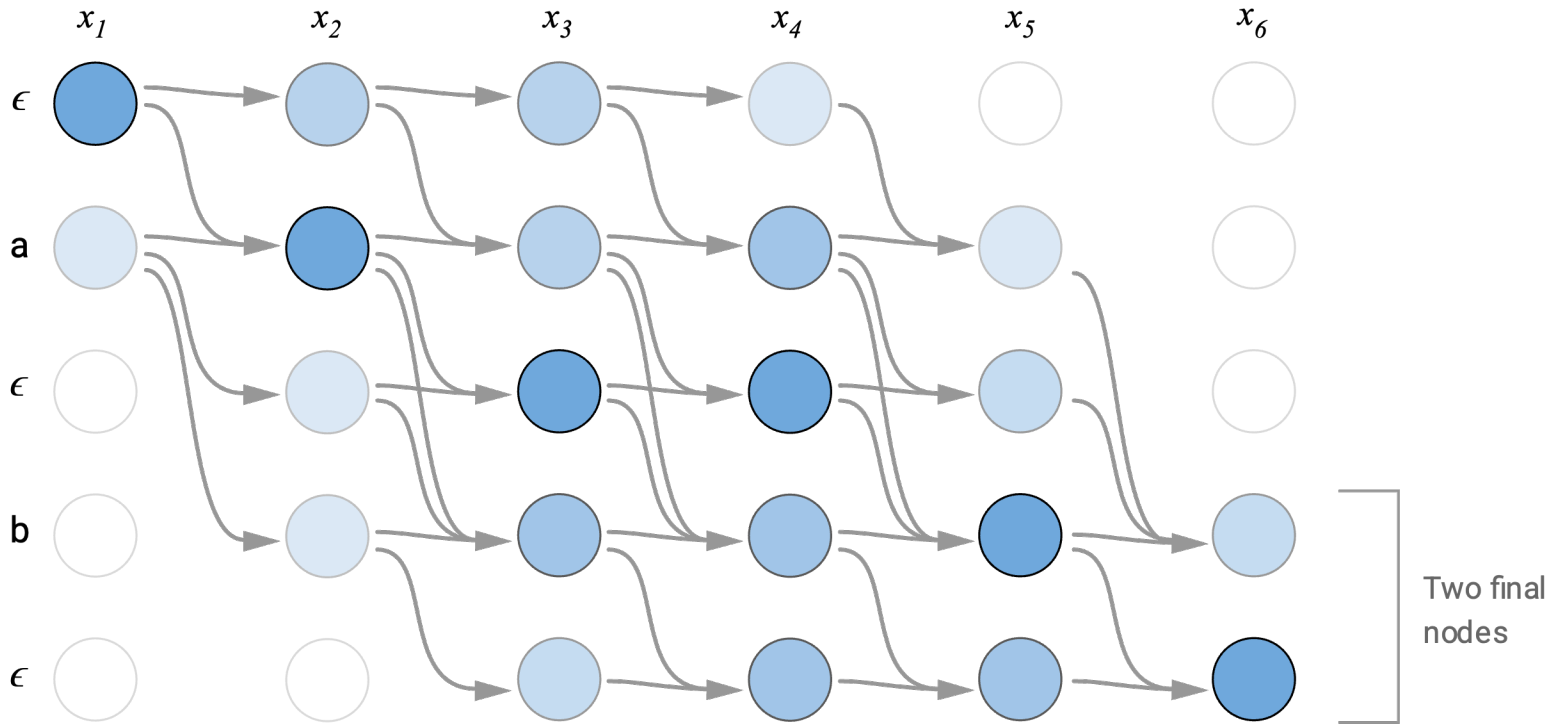
CTC Loss Function

Let ***Z*** is Y with an ϵ at the beginning, end, and between every character

$$Z = [\epsilon, y_1, \epsilon, y_2, \ldots, \epsilon, y_U, \epsilon]$$

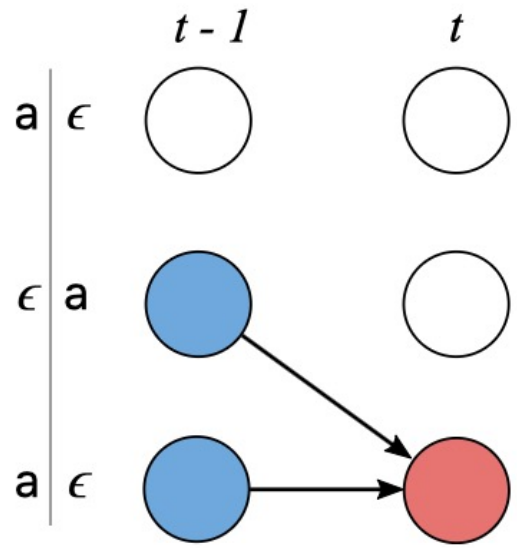
Let $\alpha_{s,t}$ be the CTC score of the subsequence $Z_{1:s}$ after t input steps

➔ $P(Y|X)$ = α's at the last time-step



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 $z_s = z_{s-2}$.



To ensure we don't skip z_{s-1}, we can either be there at the previous time-step or have already passed through at some earlier time-step. As a result there are two positions we can transition from.

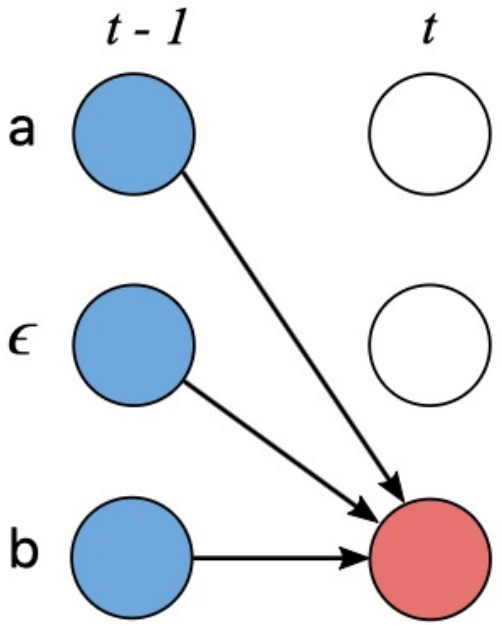
$$\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$$

The CTC probability of the two valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.

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.



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

The CTC probability of the three valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.

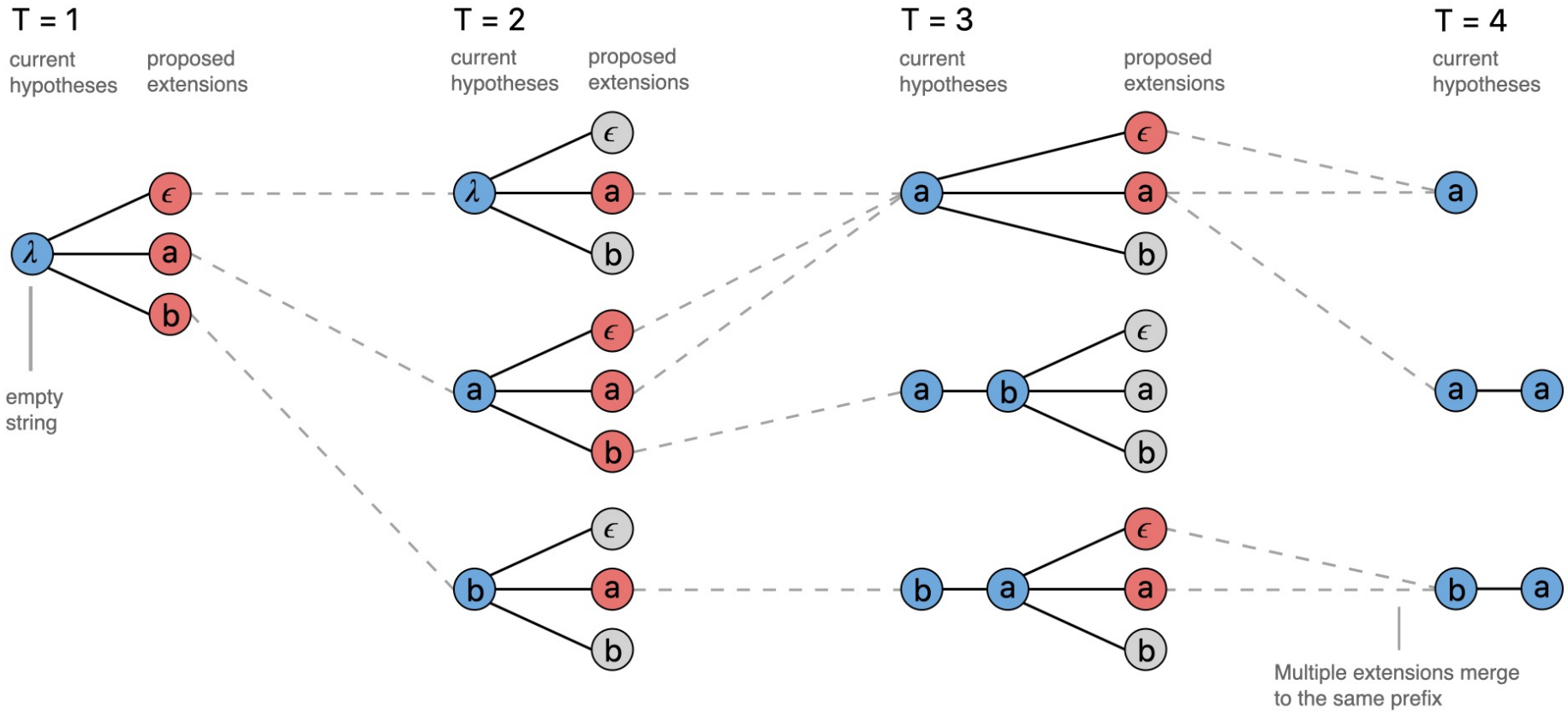
Beam Search

Heuristic ✗ alignment

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

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

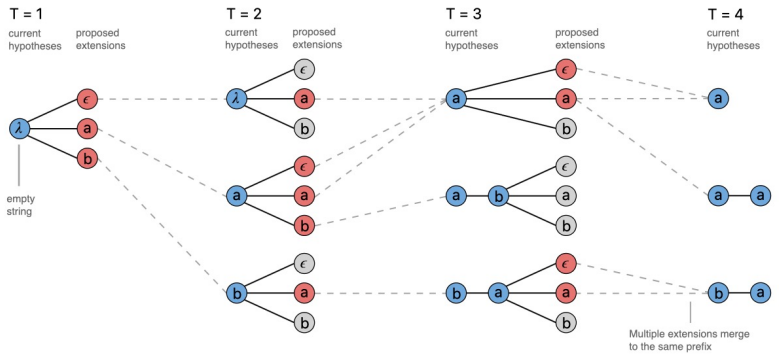
The CTC beam search algorithm to handle multiple alignments that mapping to the same output with an output alphabet {ϵ,a,b} and a beam size of three

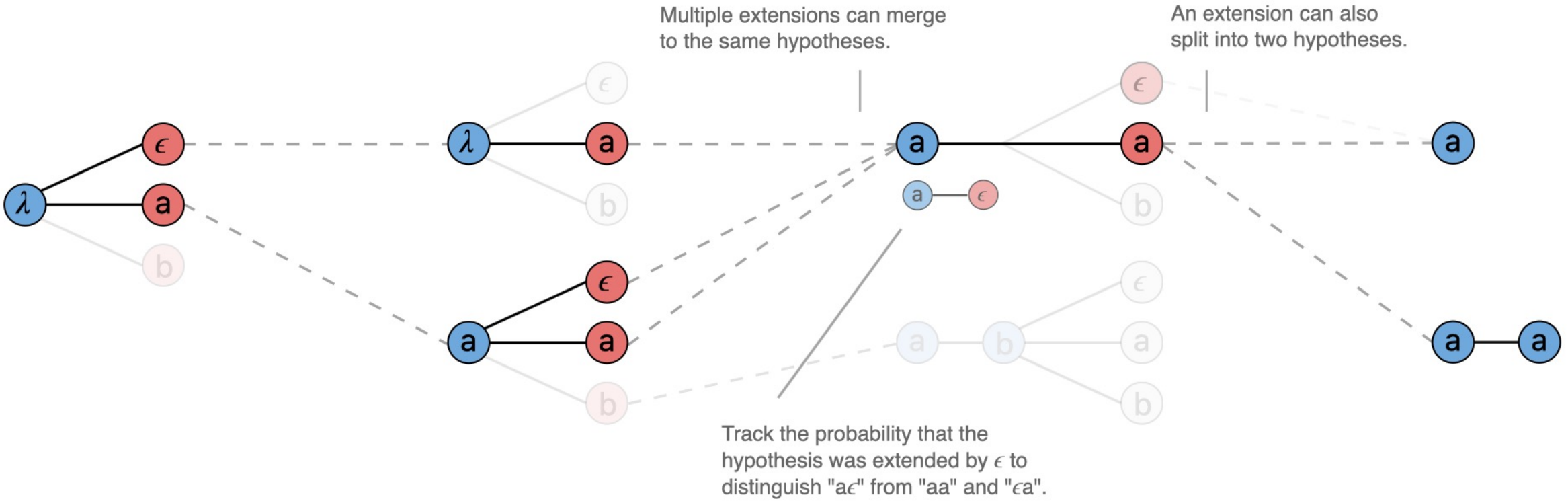


Beam Search

Only keep track of two probabilities for each prefix in the beam

- The probability of all alignments which end in ϵ
- The probability of all alignments which don't end in ϵ





Language Model Integration

$$Y^* = \underset{Y}{\operatorname{argmax}} \; p(Y \mid X) \cdot p(Y)^{\alpha} \cdot L(Y)^{\beta}$$

The CTC conditional probability.

The language model probability.

The "word" insertion bonus.

- $L(Y)$ → the length of Y in terms of the language model tokens

Deep Speech 2

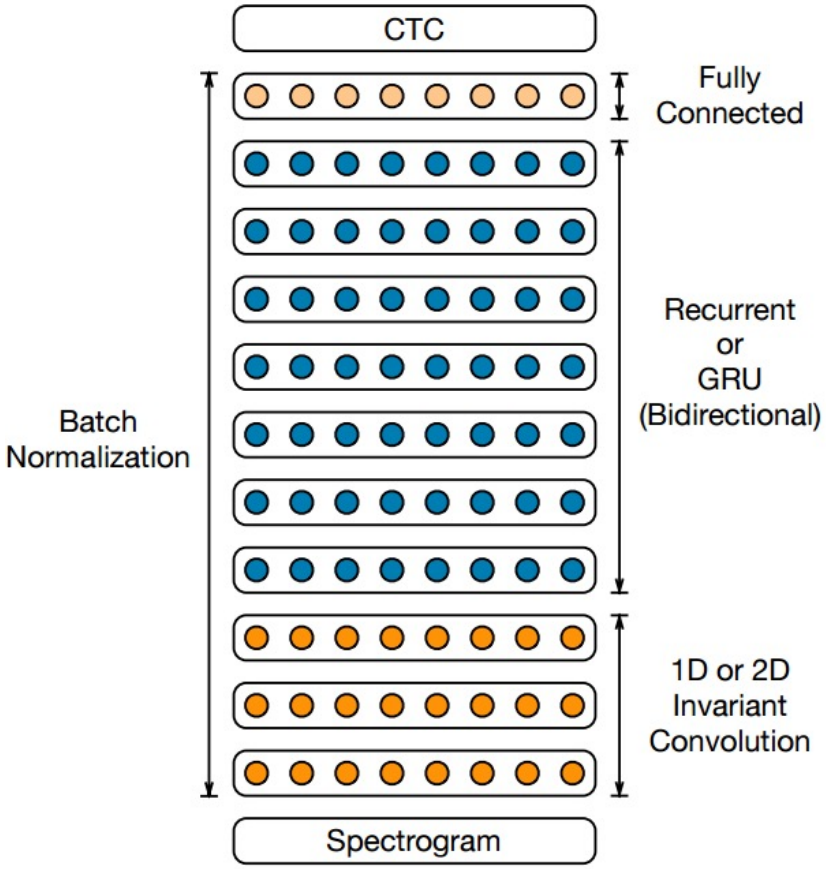
3 layers of 2D-invariant convolution

7 layers of bidirectional simple recurrence

100M parameters

Word error rates

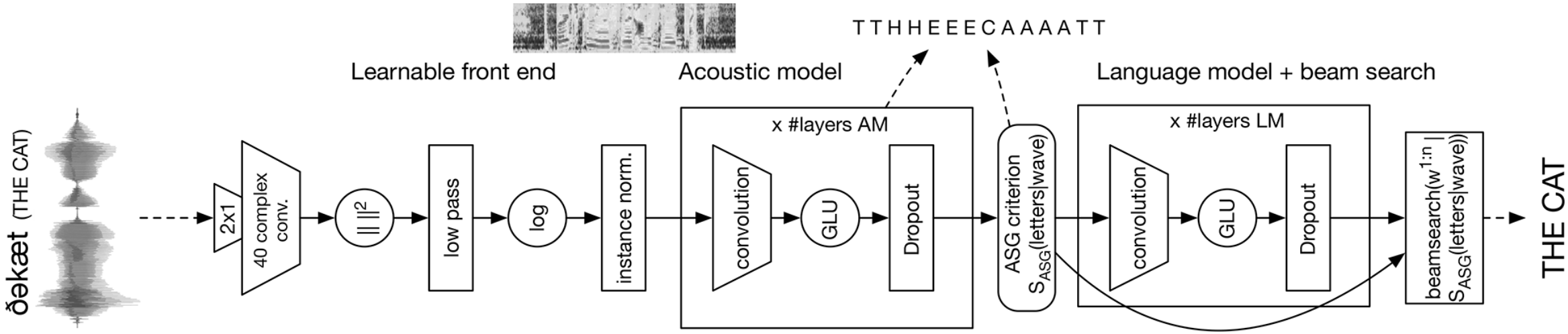
Test set	Deep speech 2	Human
WSJ eval'92	3.60	5.03
WSJ eval'93	4.98	8.08
LibriSpeech test-clean	5.33	5.83
LibriSpeech test-other	13.25	12.69



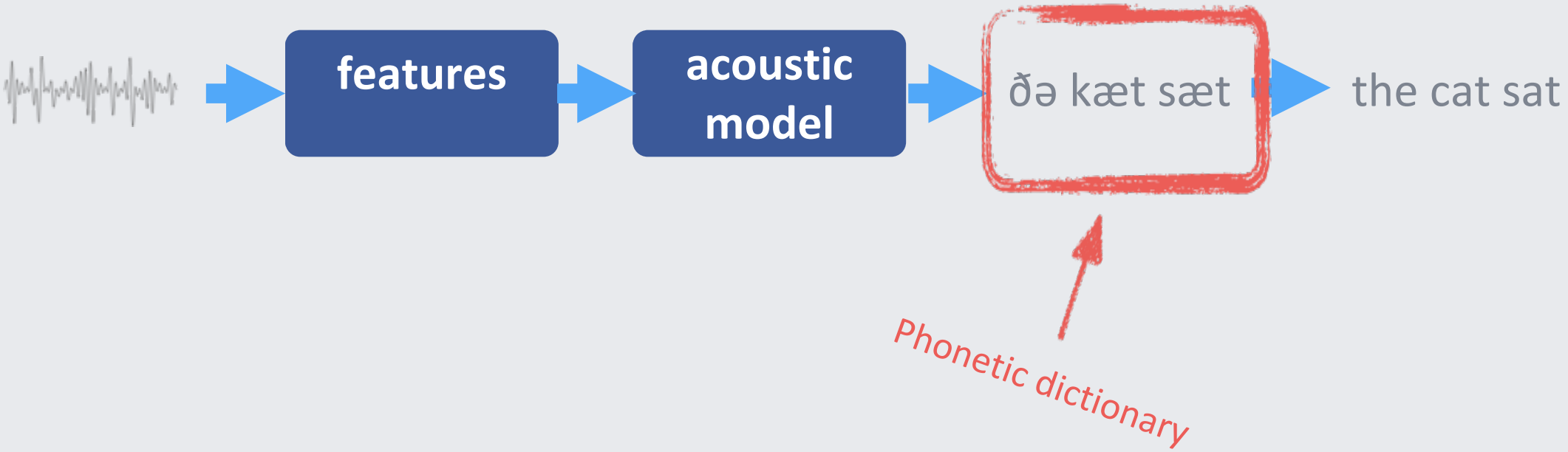
Amodei, Dario, et al. "Deep speech 2: End-to-end speech recognition in english and mandarin." *arXiv preprint arXiv:1512.02595* (2015).

wav2letter++

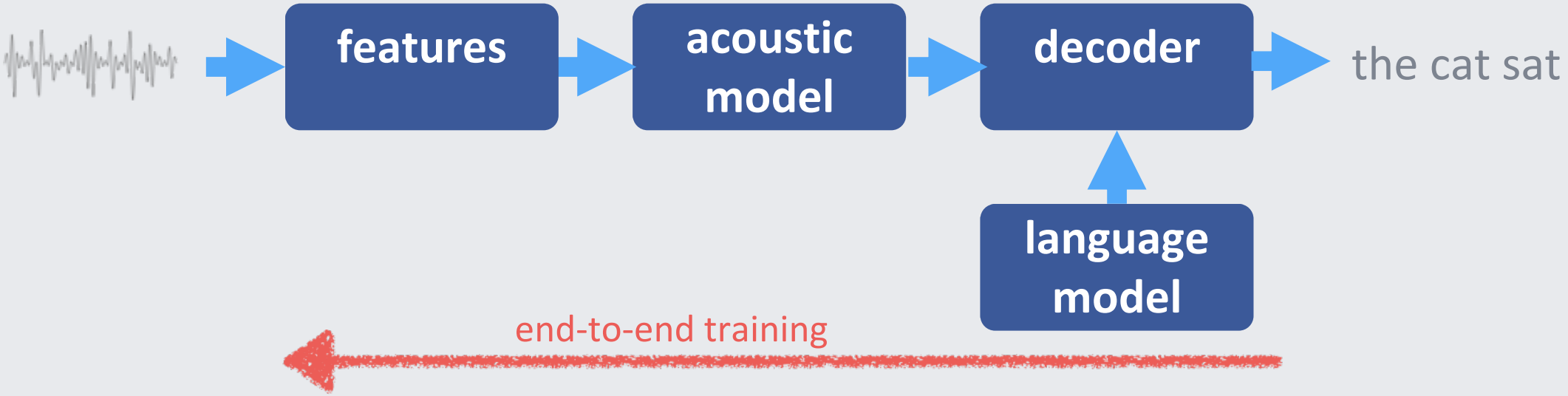
Facebook's fast open-source speech recognition system



automatic speech recognition



end-to-end speech recognition



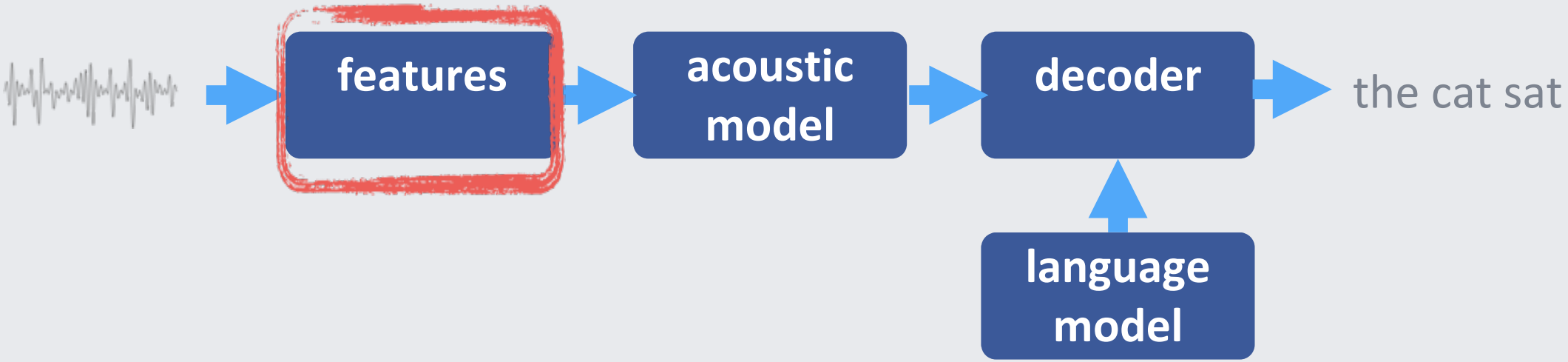
acoustic model — Can we make it **simple**?

language model — Better but **scalable**?

features — Can we **train** them?

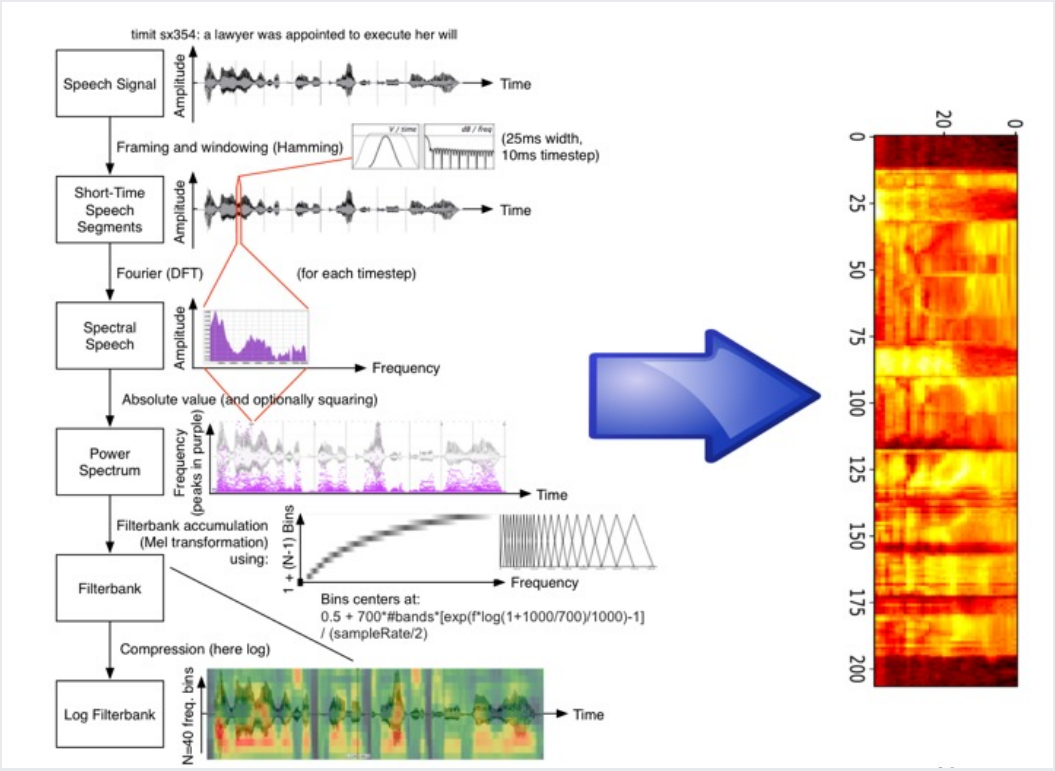
decoder — Can we make it **differentiable**?

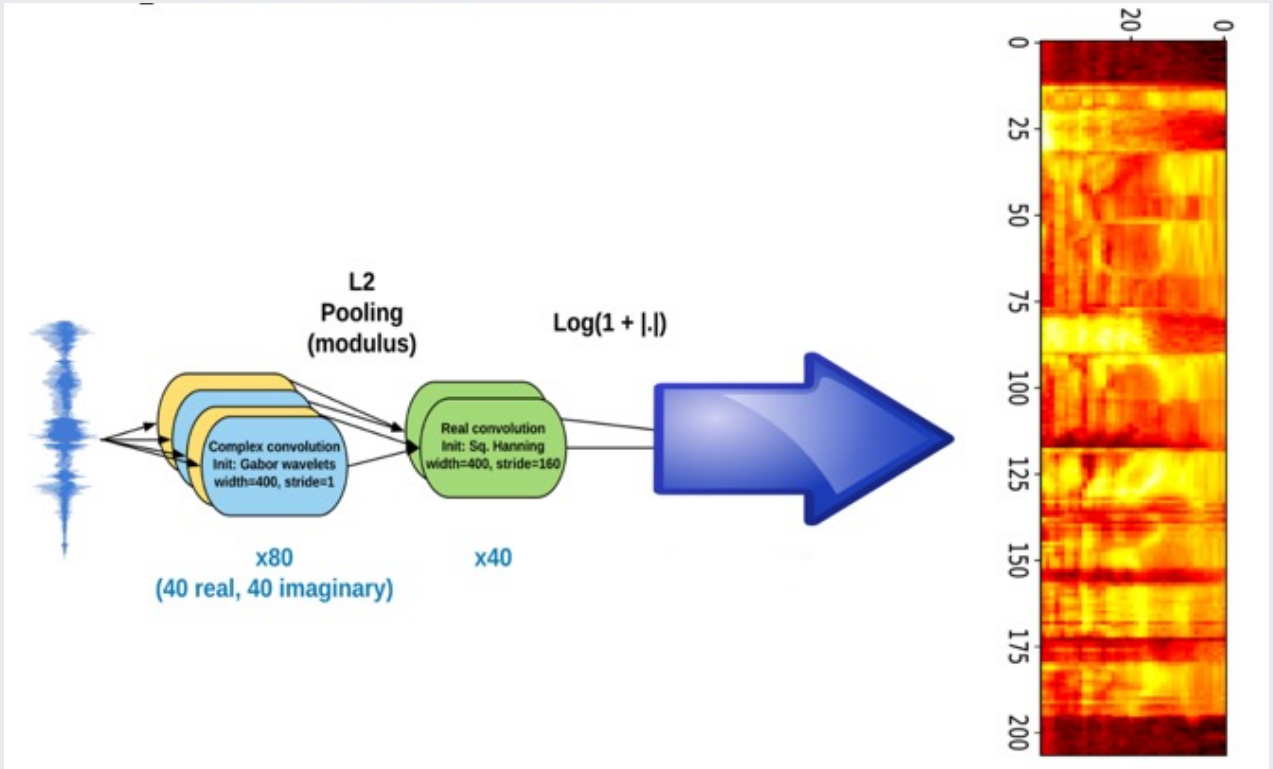
end-to-end speech recognition



features

Train them





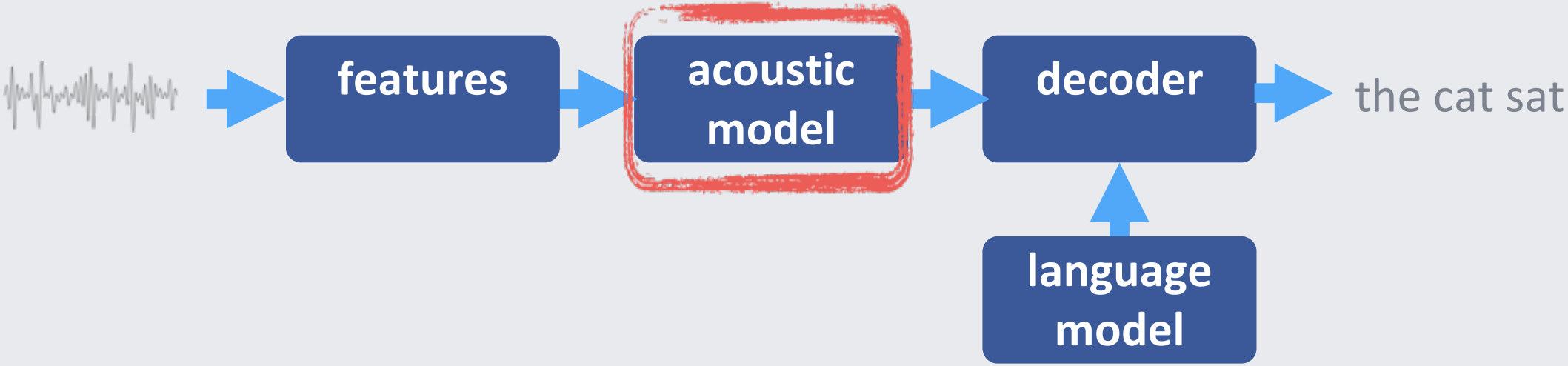
Log-mel filterbanks

Trainable front-end

- Approximate Log-mel filterbanks at initialization
- Trained with the rest of the network

"Learning filterbanks from raw speech for phone recognition", Zeghidour et al., *ICASSP 2018*
"End-to-end speech recognition from the raw waveform", Zeghidour et al., Interspeech 2018

end-to-end speech recognition



acoustic model

How it works

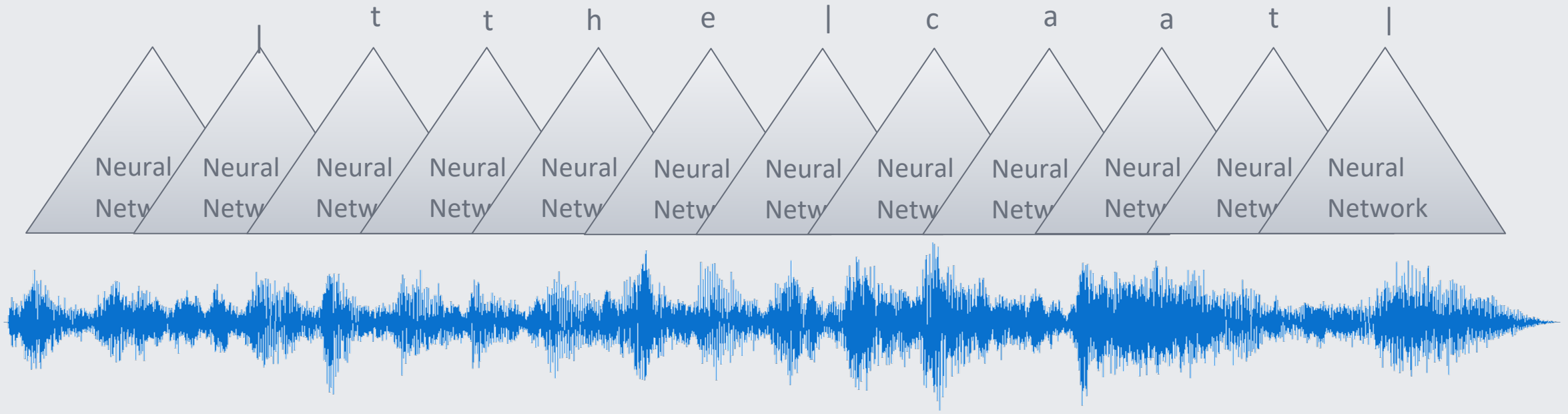
Duration model:

max

|	10.356
a	0.31
b	-1.254
c	0.001
...	...
y	10.211
z	-10.21

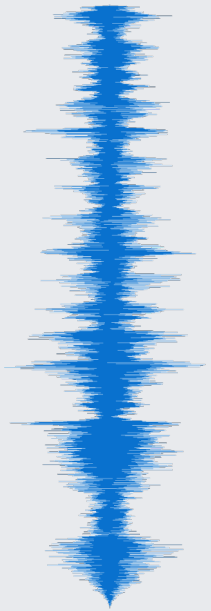
|||the|caat|||ssaattt| → |the|cat|sat|

* let's remember that | stands for silence

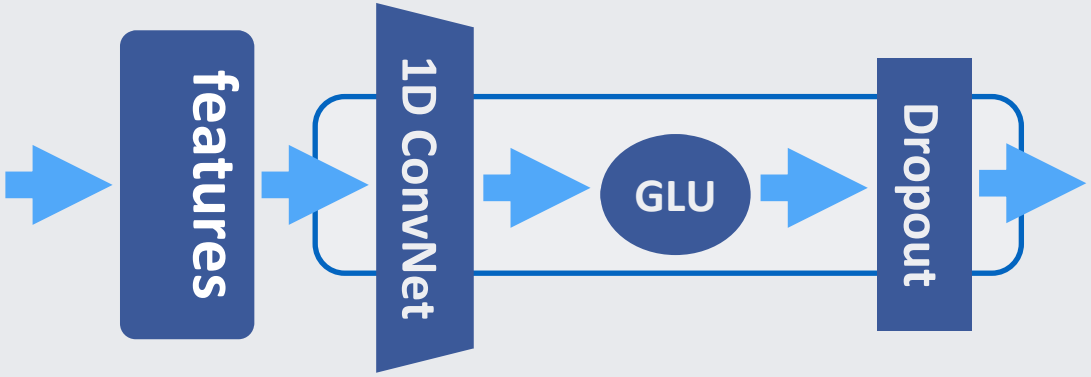


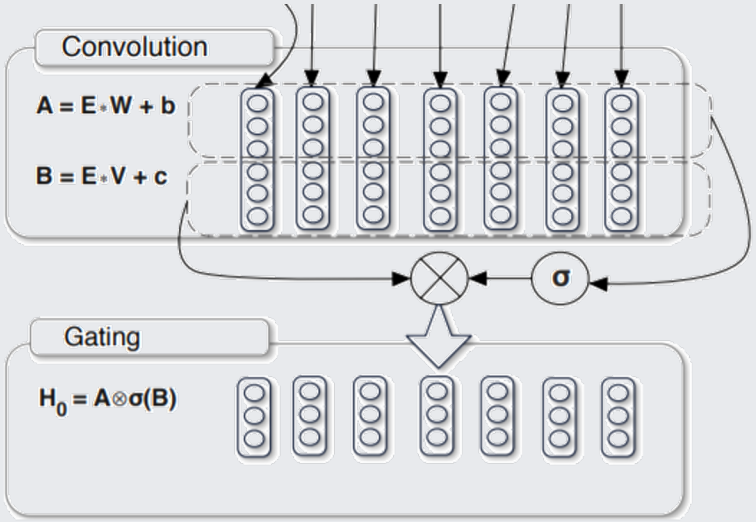
acoustic model

Make it **simple**



Gated convnet block





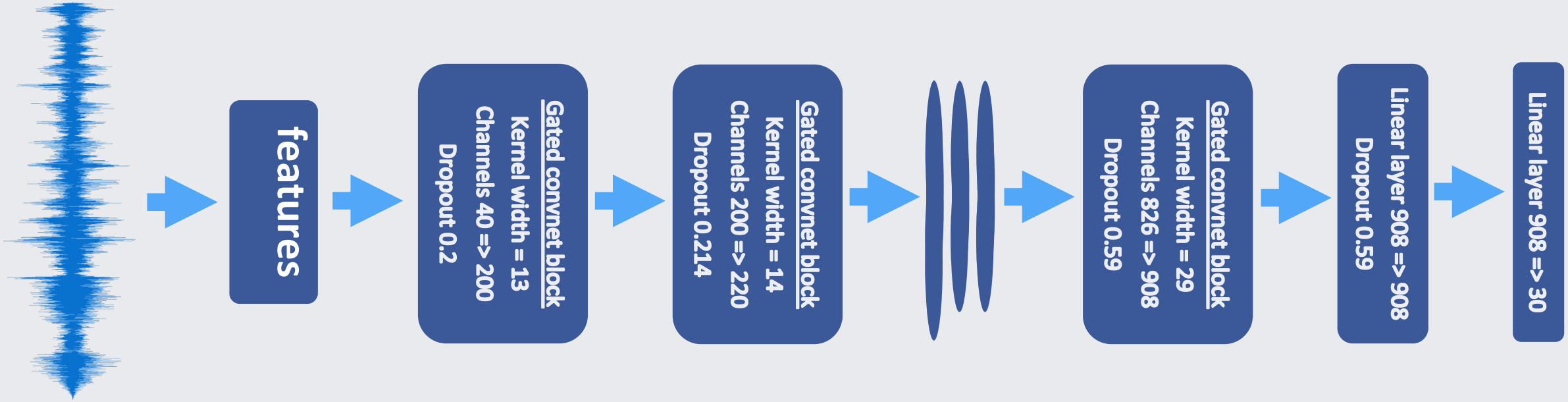
Gated Linear Units (GLU):

- Address vanishing gradient problem
- Successful application to NLP problems
- Gates: $\sigma(x) = 1/(1 + e^{-x})$
- $\otimes$ is element-wise product between matrices

"Language Modeling with Gated Convolutional Networks", Dauphin et al., *ICML, 2017*
"Letter-Based Speech Recognition with Gated ConvNets", Liptchinsky et al., arXiv 2017

acoustic model

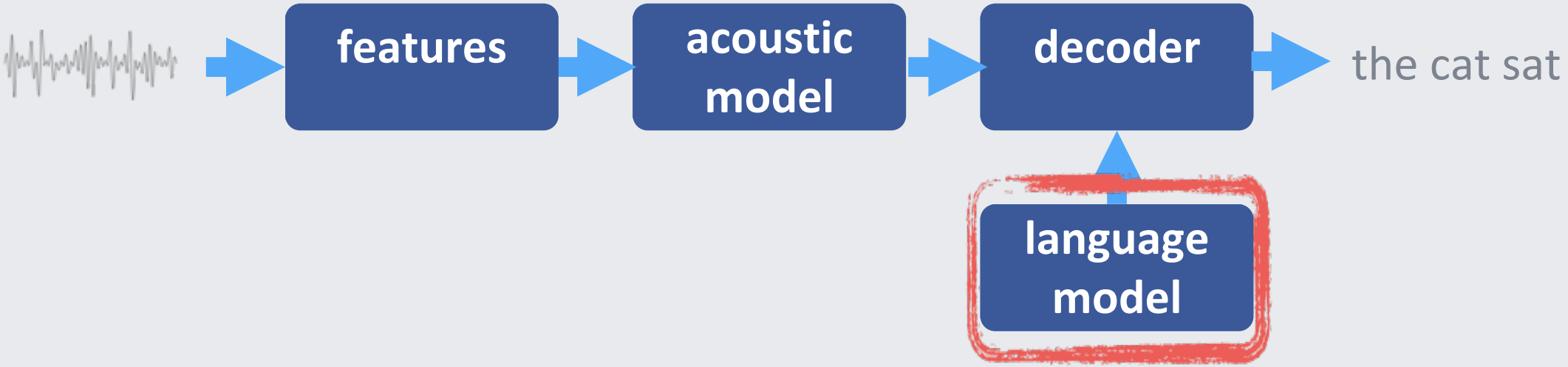
Architecture and few tricks



- For each consecutive convolutional layer: increase kernel width, increase channels, increase dropout
- Overall network receptive field of ~2.2 seconds, i.e. 2.2 seconds of audio correspond to one character
- Motivation: more modeling capacity and regularization towards output layers

"Fully Convolutional Speech Recognition", Zeghidour et al., *arXiv, 2019*

end-to-end speech recognition



language model

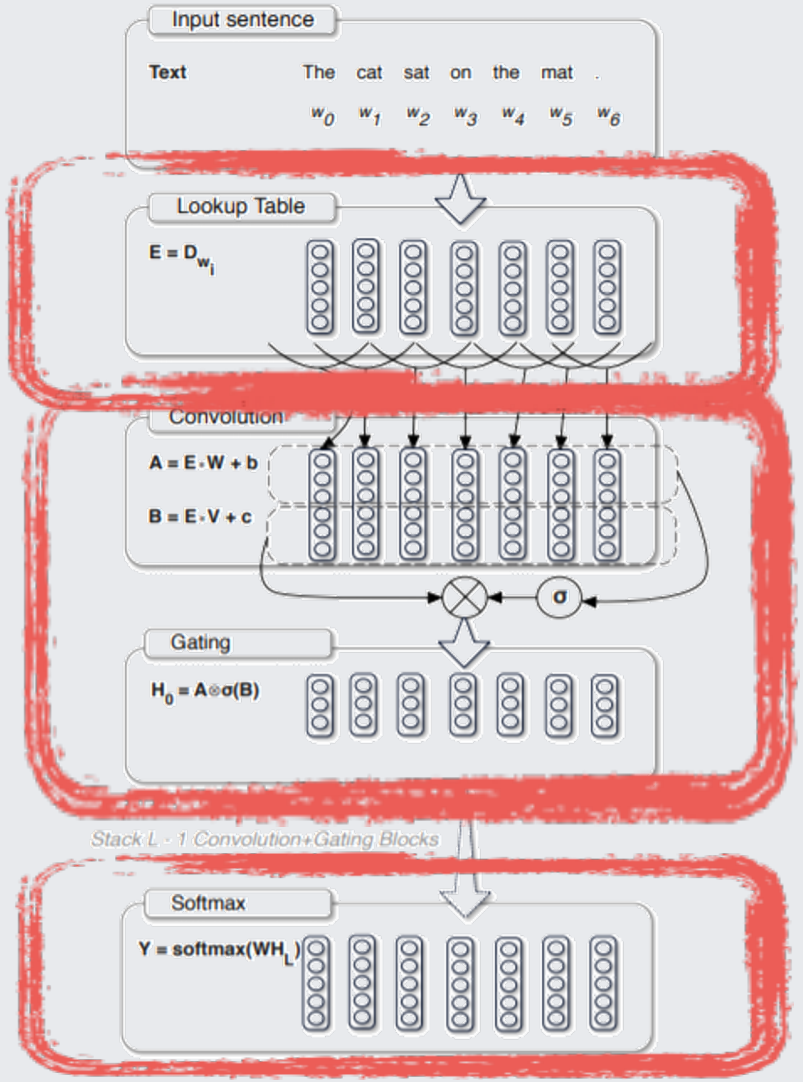
How it works

- **Language model**
  - Statistical (n-gram) language models estimate probability distribution of a sequence of words, i.e. 3-gram language model generates $P('the',\ 'cat',\ 'sat')$
  - Feed-forward Neural Network models generate probability of a next word given a sequence of words, i.e. $P('sat'|'the','\ cat')$.
  - Character based language models: $P('s'|'t','\ h','\ e','\ c','\ a','\ t')$
- **Do acoustic models learn language modeling?**
  - Anecdotally, acoustic models were observed to output $ttttthheeee$ instead of $aaaaaaaa$ in noisy audio segments.
  - Thus, more regularization and more capacity at the output layers of acoustic models.

language model

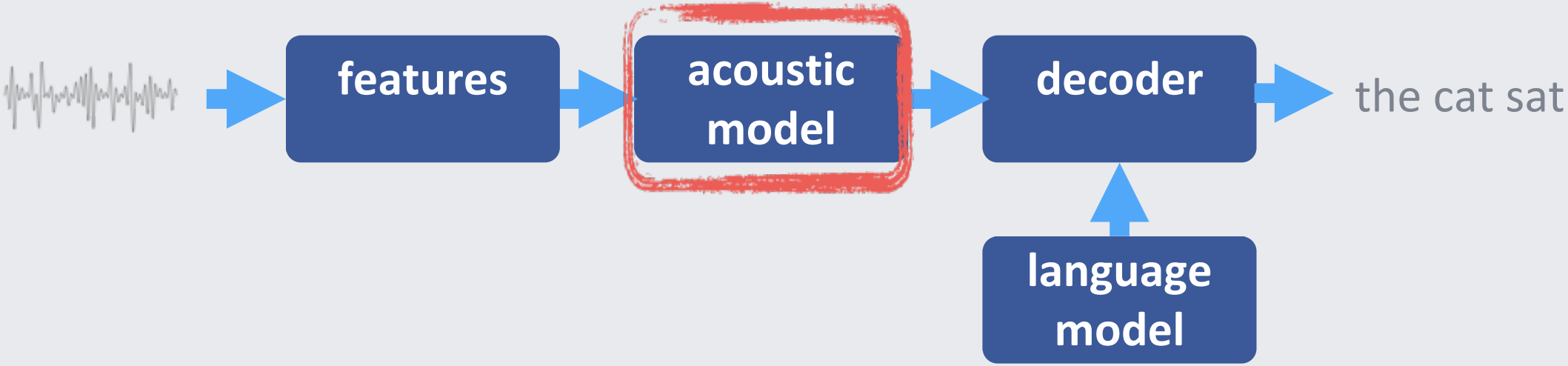
Architecture of feed-forward neural network language models

- **Word embeddings**
- **Gated Linear Units to the rescue!**
- **Hierarchical Softmax: output probabilities for all words in the dictionary**



"Language Modeling with Gated Convolutional Networks", Dauphin et al., *ICML, 2017*

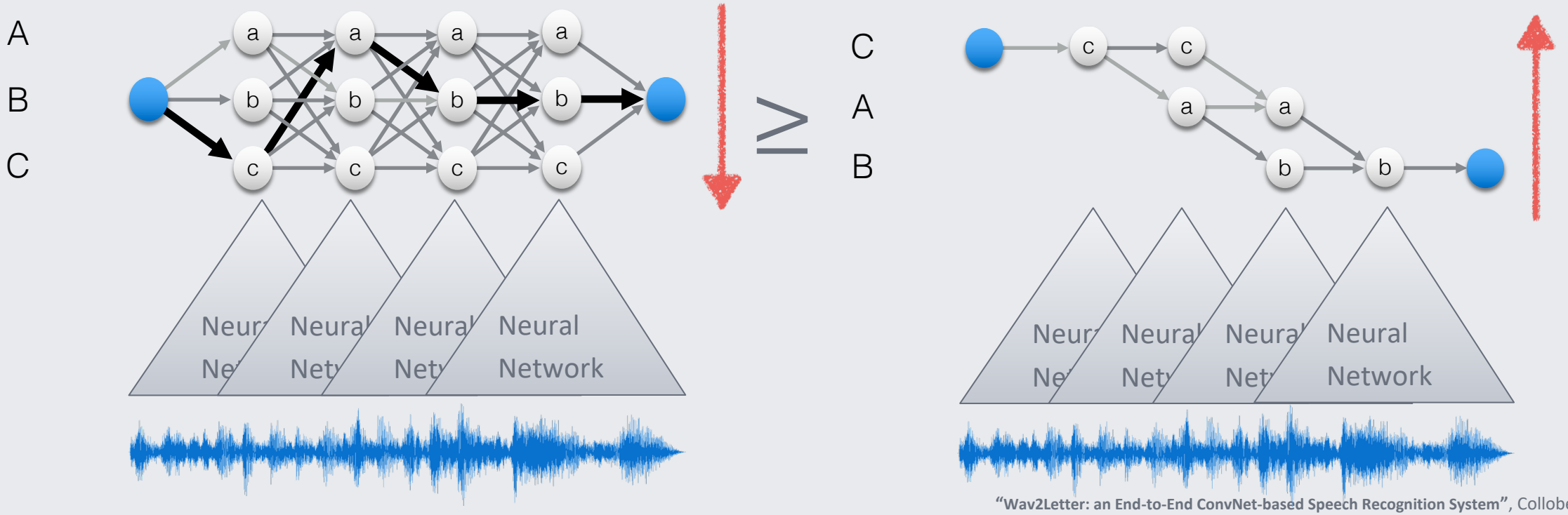
end-to-end speech recognition



acoustic model

Training: ASG loss (criterion)

- ASG stands for Auto Segmentation
- Segmentation problem: say "cab" is the target (the letter vocabulary is $\{a, b, c\}$)
  - Over 4 frames, can be written caab, ccab, cabb, etc…
- Unnormalized transition scores: $a \to a, a \to b, \ldots, c \to b, c \to c$
  - $score(caab) = score_{acoustic}\ (c) + score_{transition}\ (c \to a) + score_{acoustic}\ (a) + \ \ldots + score_{acoustic}\ (b)$

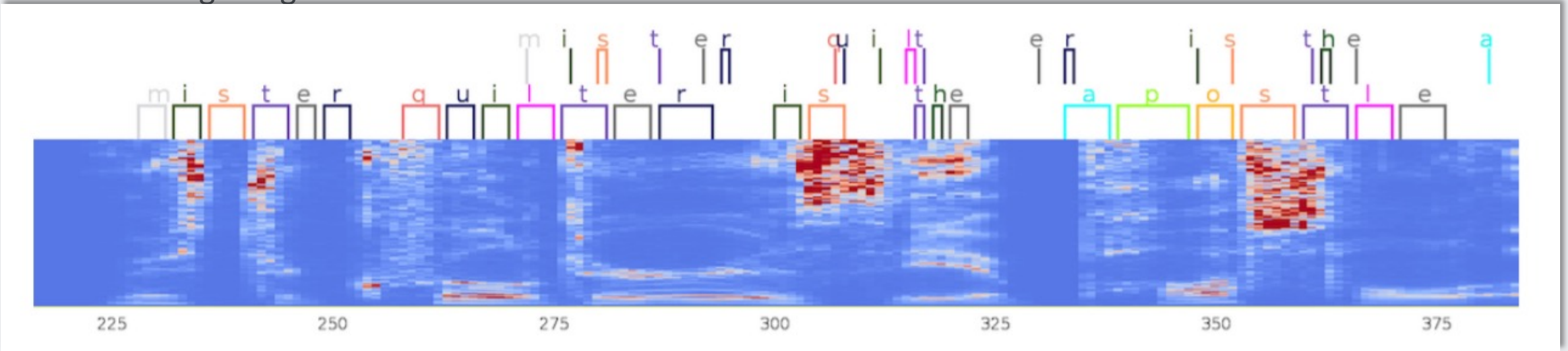


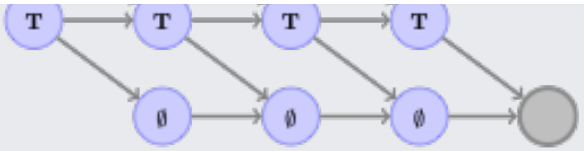
"Wav2Letter: an End-to-End ConvNet-based Speech Recognition System", Collobert et al., *arXiv, 2016*

acoustic model

Training: CTC vs ASG

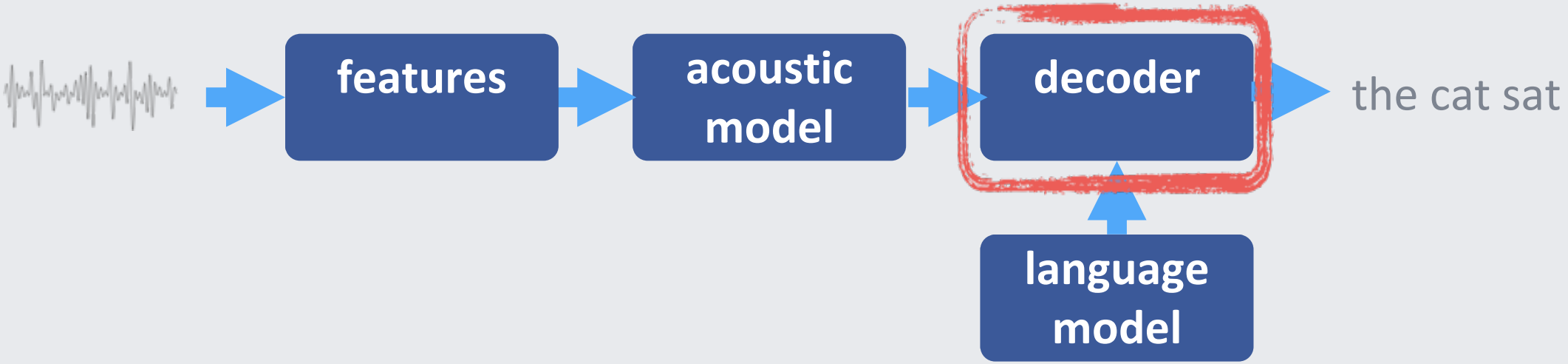
- CTC stands for Connectionist Temporal Classification
- Extensively used in Speech Recognition, Optical Character Recognition
- Has a **blank label ø**
  - Handles letøter repetitions
  - Handles garbage frames





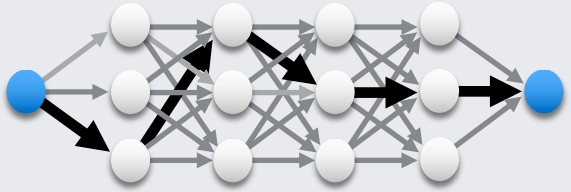
"Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks", A. Graves et al., *ICML, 2006*
"Letter-Based Speech Recognition with Gated ConvNets", Liptchinsky et al., arXiv 2017

end-to-end speech recognition

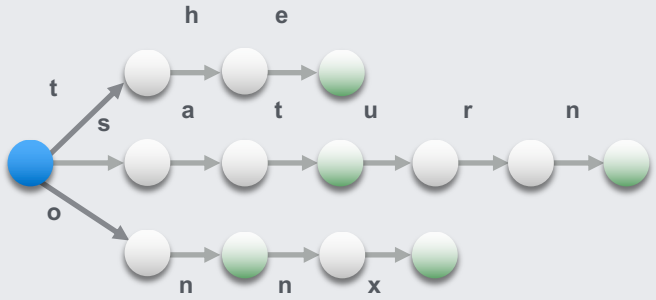


decoder

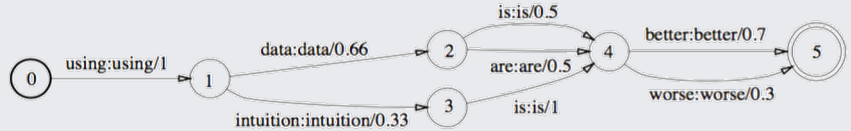
How it works



acoustic
A

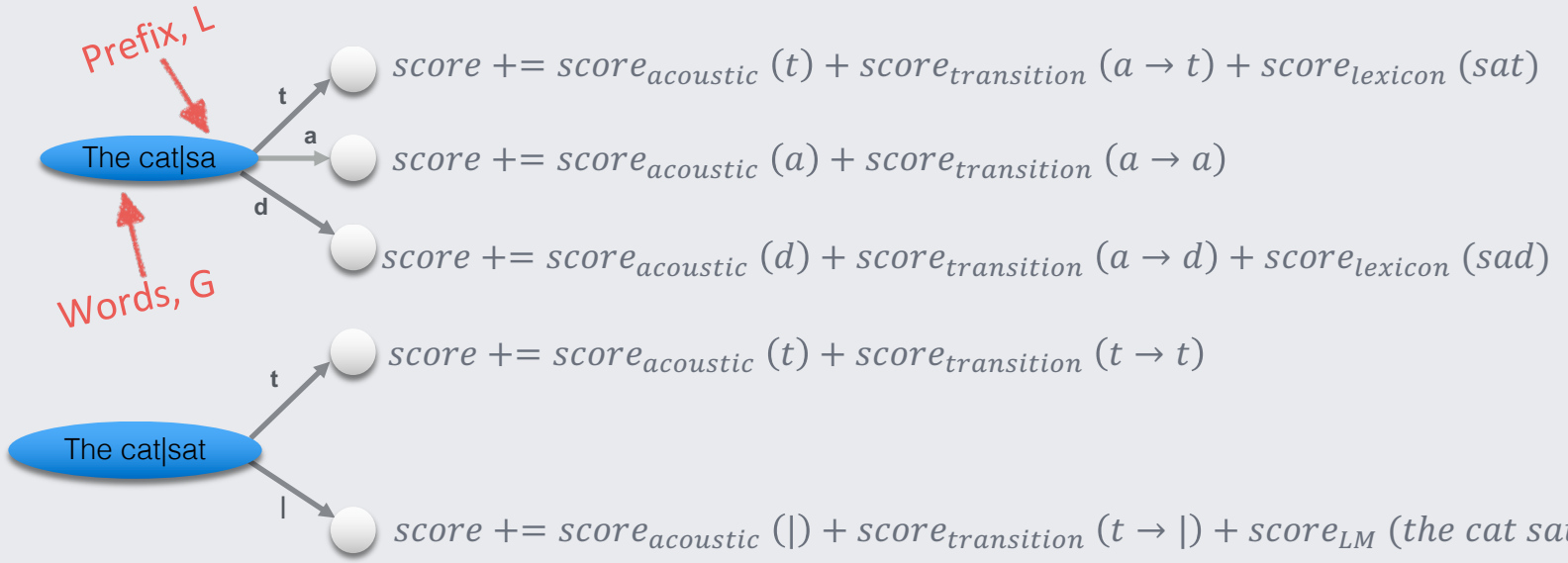


lexicon
L



word LM
G

- Beam search, constrained to fixed beam size
- Bookkeeping of (A, L, G) positions
- At **each time step**:
  - For each previous hypothesis (A, L, G, score)
    - Add *new hypothesis* constrained to **L**
    - If word is emitted, add score from **G**
  - **Merge** *new hypothesis* leading to same (L, G) states
  - For details on the differentiable decoder please check the paper in the footnote.



"A Fully Differentiable Beam Search Decoder", Collobert et al., *arXiv, 2019*

Word Error Rate (WER)

- **How it is computed:**
  - Levenstein distance between transcription produced by ASR system and the reference, at the word level
  - $WER = \frac{Substitutions + Deletions + Insertions}{Total\ number\ of\ words\ in\ the\ reference}$
  - Examples:
    - REF: *the cat sat on the mat*
    - HYP: *the cat sat mat*
    - WER: 33%, 2 deletions
    - REF: *the cat sat on the mat*
    - HYP: *the bat sat on at the mat*
    - WER: 33%, 1 substitution, 1 insertion

Fully Convolutional ASR

all results in Word Error Rate

	WSJ
CNN-BLSTM-HMM [1] *phone-based*	3.7
Deep Speech 2 [2] letter-based + 12Kh audio data	3.6
Encoder-Decoder [3] *letter-based*	6.7
Wav2letter GLU ConvNet count-based word LM	**5.6**
Wav2letter GLU ConvNet + ConvLM	**4.1**
Wav2letter GLU ConvNet WAV + ConvLM	**3.5**

	LibriSpeech Clean	LibriSpeech Other
HMM+CNN+iVectors [1] *phone-based*	4.8	12.5
Deep Speech 2 [2] letter-based + 12Kh audio	5.3	13.3
Attention Model [3] *letter-based*	3.8	12.8
Wav2letter GLU ConvNet count-based word LM	**4.26**	**14.54**
Wav2letter GLU ConvNet + ConvLM	**3.45**	**11.92**
Wav2letter GLU ConvNet WAV + ConvLM	**3.26**	**10.47**

[1] **Deep Recurrent Neural Networks for Acoustic Modelling,** Chan and Lane
[2] **Deep Speech 2: End-to-End Speech Recognition in English and Mandarin,** Amodei et al.
[3] **Towards better decoding and language model integration in sequence to sequence models**, Chorowski and Jaitly

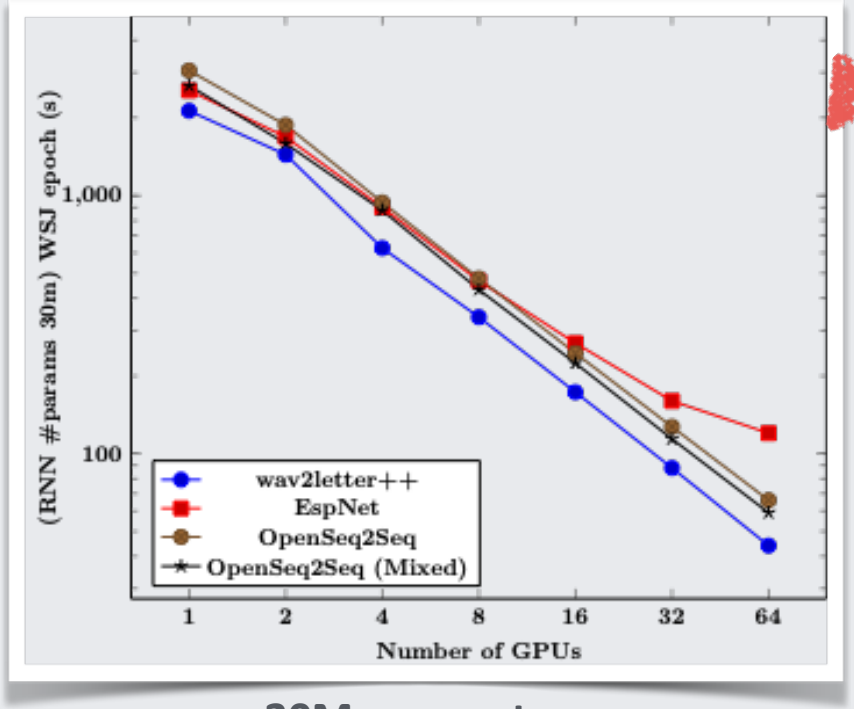
ASR toolkits

	Language	Model(s)	ML backend
Kaldi	C++, Bash	HMM/GMM/DNN **LF-MMI**	-
ESPNet	Python, Bash	**CTC**, seq2seq, hybrid	**Pytorch** Chainer
OpenSeq2Seq	Python, Perl, Bash	**CTC**, seq2seq	**TensorFlow**
wav2letter++	C++	**CTC**, seq2seq, ASG	**ArrayFire**

benchmark: training epoch time

- 8 GPUs nodes (Tesla V100), 100Gbps InfiniBand
- CTC training; Kaldi: LF-MMI

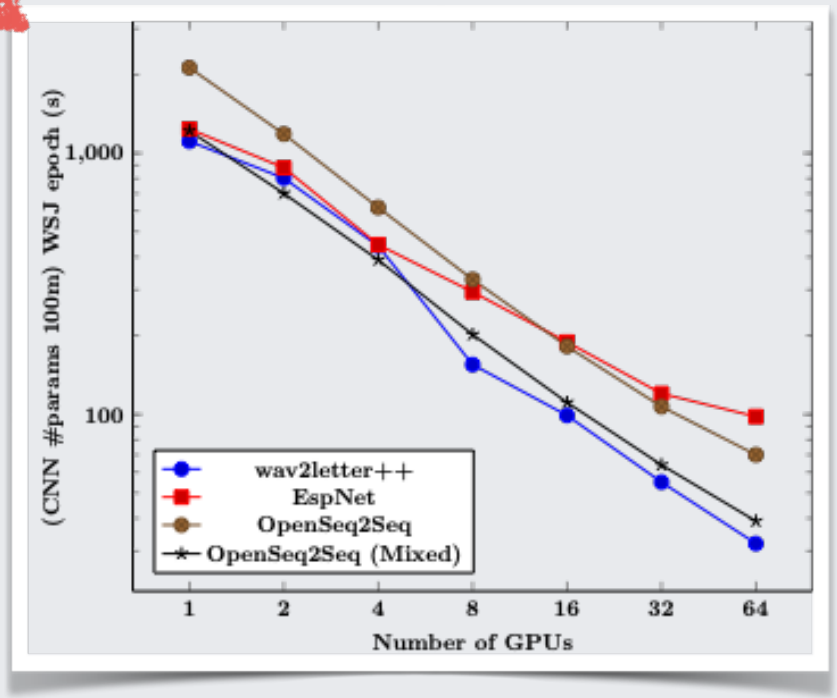
log-scale



30M parameters

2 Convolutions

5 bi-LSTM

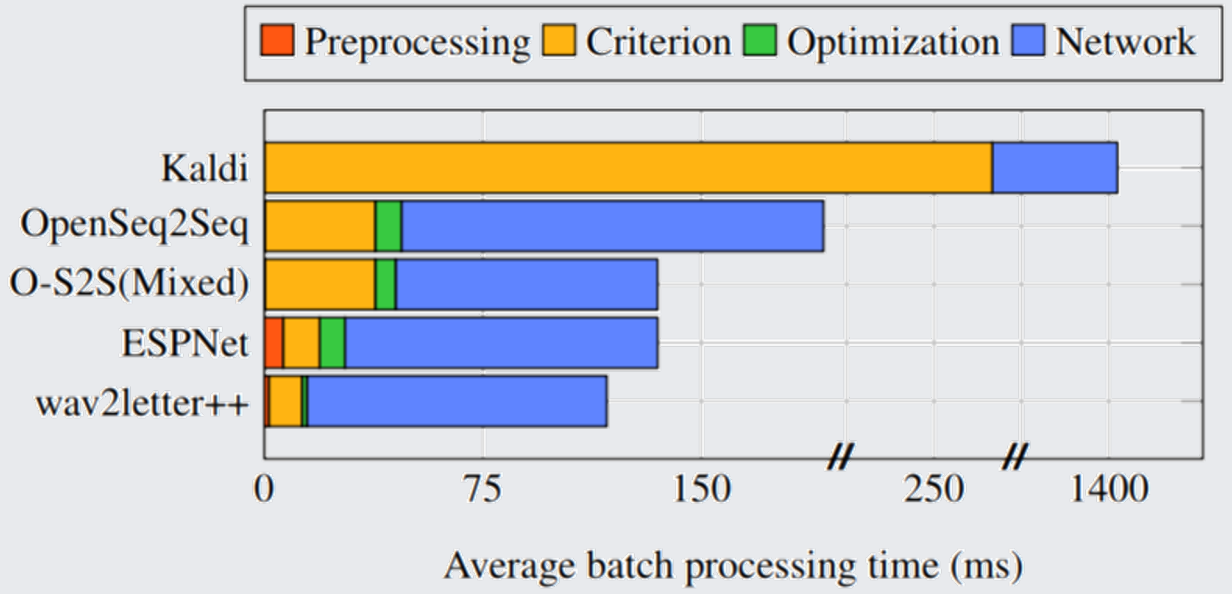


100M parameters

19 Convolutions

benchmark: training epoch time

- 8 GPUs nodes (Tesla V100), 100Gbps InfiniBand
- CTC training; Kaldi: LF-MMI



100M parameters

19 Convolutions

benchmark: decoding

- Same **pre-computed** emissions for all frameworks
- LibriSpeech dev-clean, **4-gram LM**

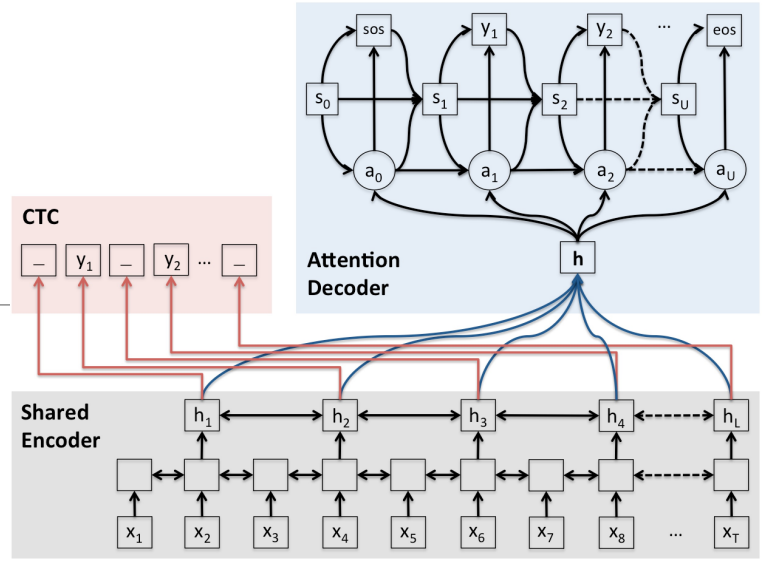
does not support n-gram LM

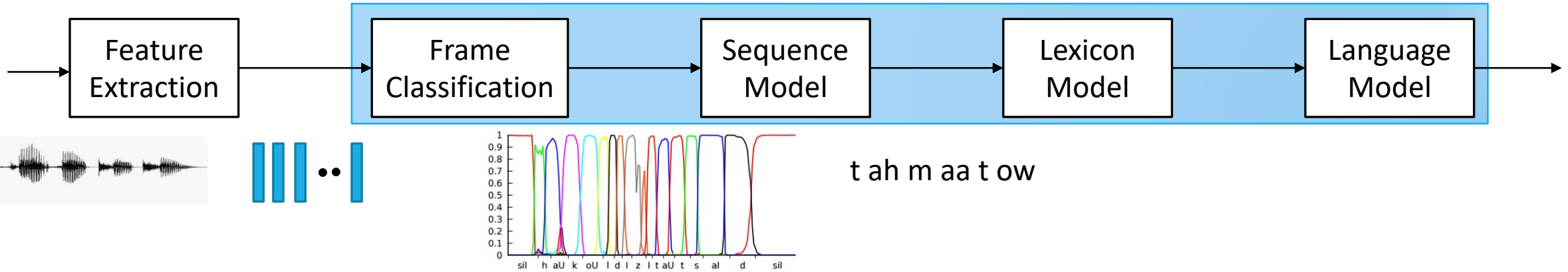
	WER (%)	Time/sample (ms)	Memory (GB)
ESPNet	7.20	1548	-
OpenSeq2Seq	5.00	1700	7.8
OpenSeq2Seq	4.92	9500	26.6
wav2letter++	5.00	10	3.9
wav2letter++	4.91	140	5.5

Attention in Speech Recognition

Predict character sequence directly

Sequence-to-Sequence with Attention





A Speech Audio

O Feature Frames

Q Sequence States

L Phonemes

W Words

Sentence

Goal: Model Characters directly from Acoustics

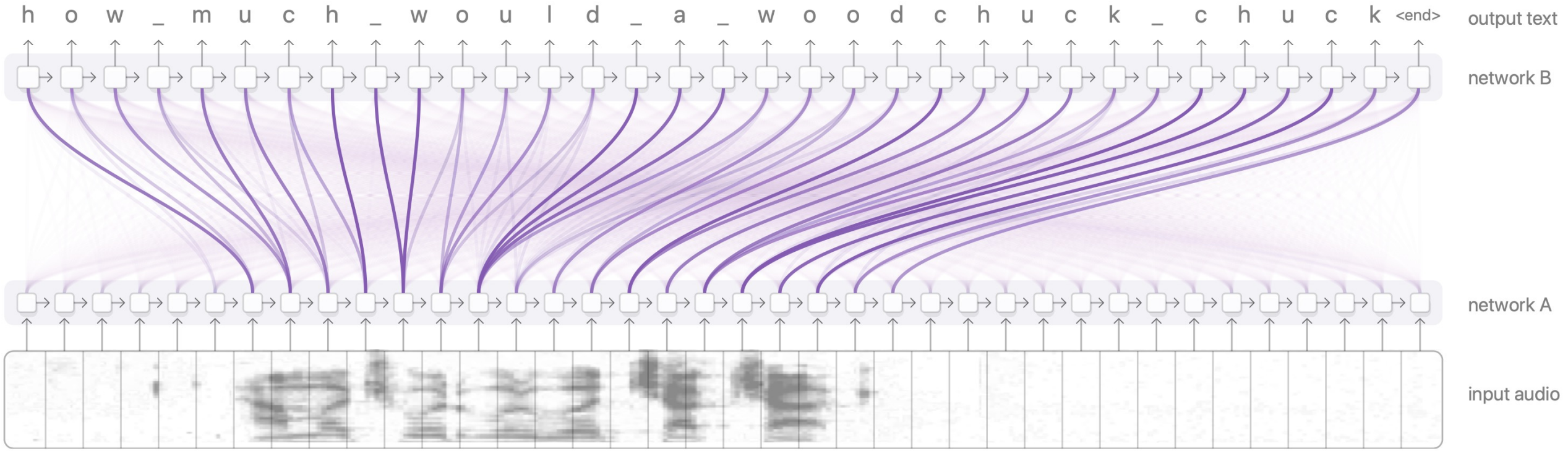
Input $\mathbf{x} = (x_1, \ldots, x_T)$

- Acoustic signal (e.g., filterbank spectra)

Output $\mathbf{y} = (y_1, \ldots, y_T)$

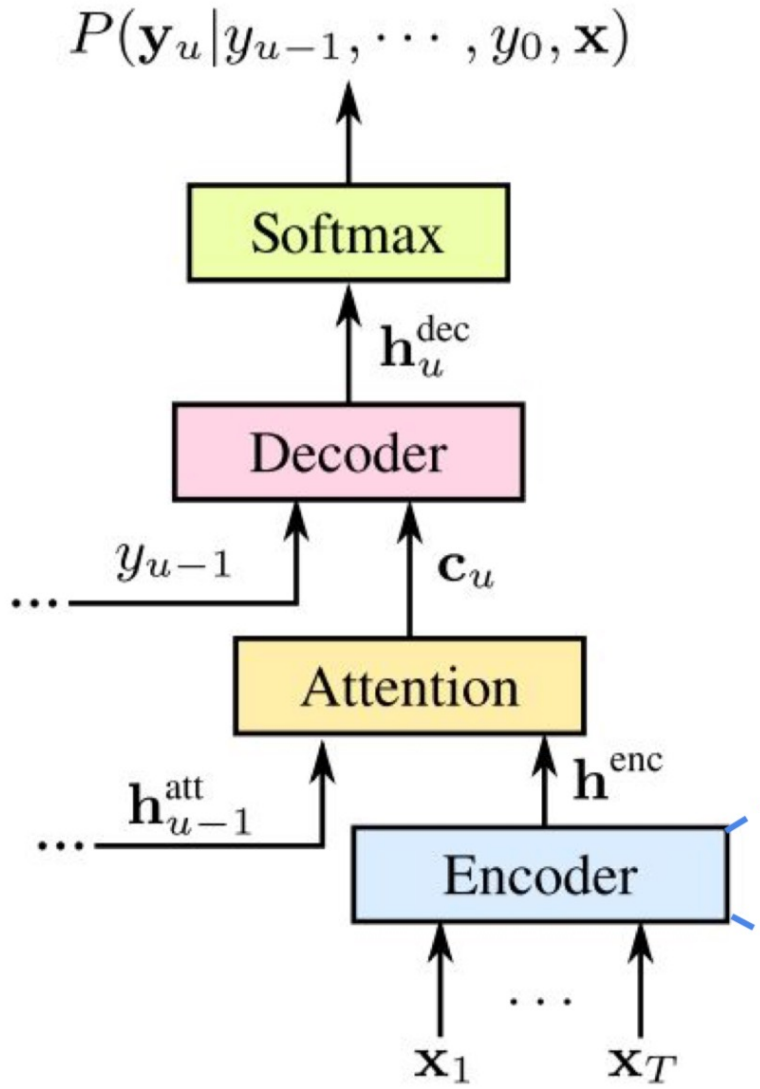
- (English) characters
- Don't make assumptions about the our distributions

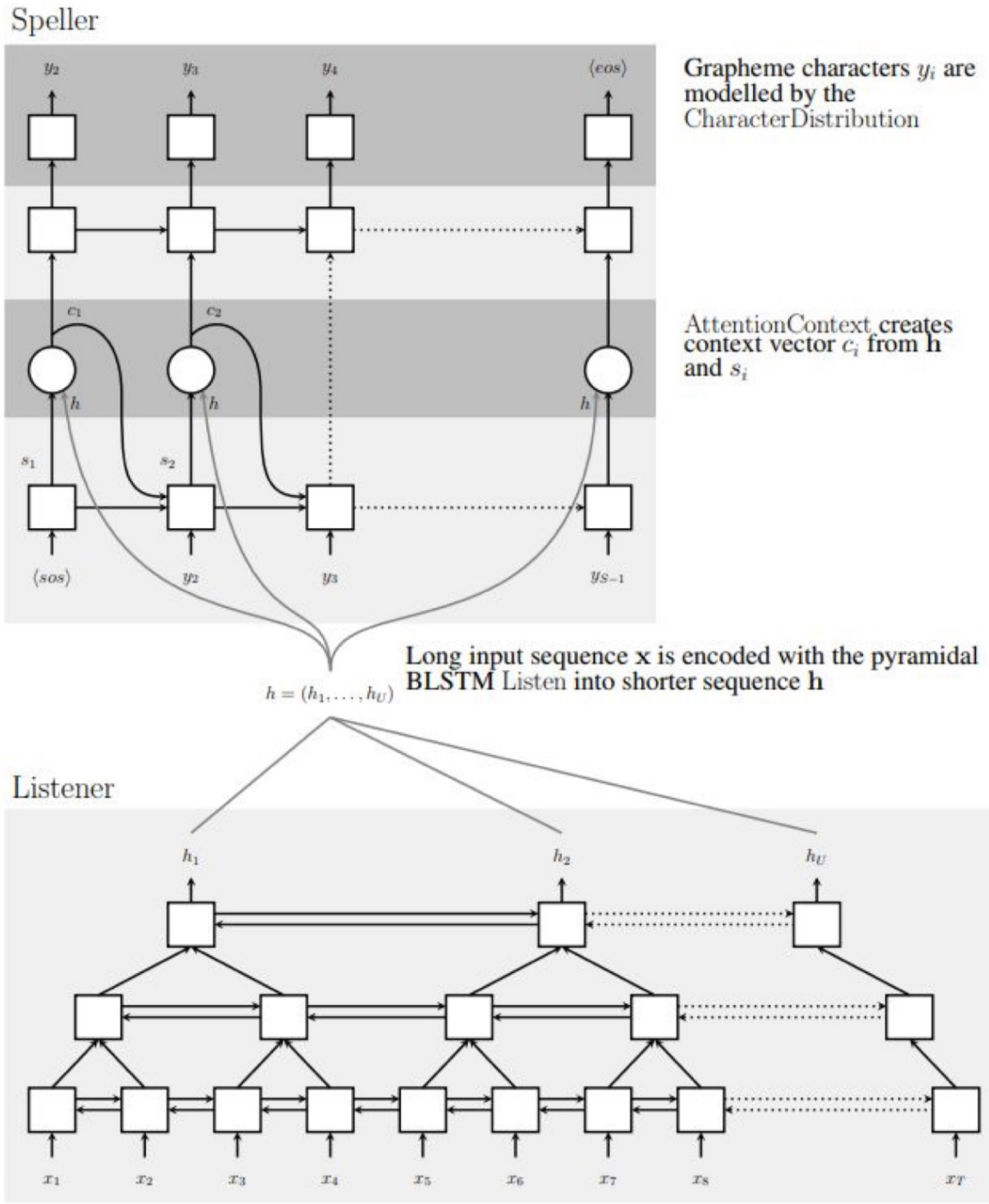
Goal: Model Characters directly from Acoustics



Attention-based Models

Sequence-to-Sequence with Attention



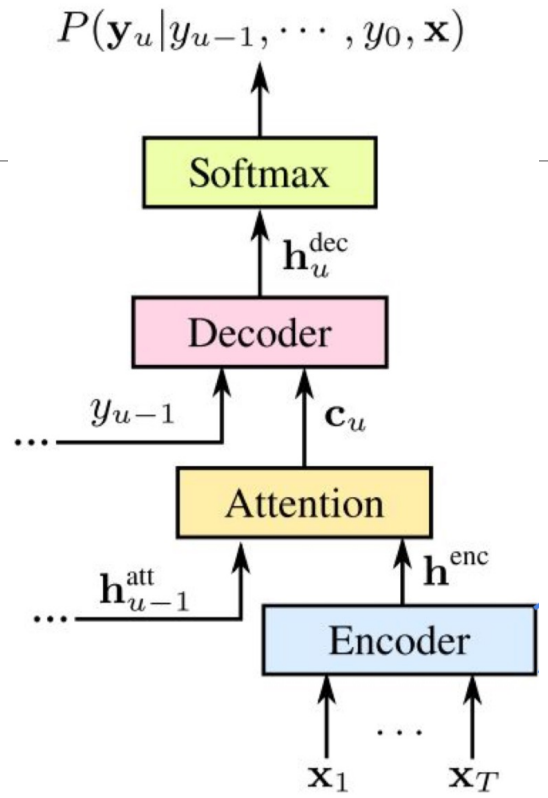


End-to-End Model

Signal Processing

Listen, Attend and Spell (LAS)

Language Model?



One model optimized end-to-end

- Learn pronunciation, acoustic, dictionary all in one end-to-end model

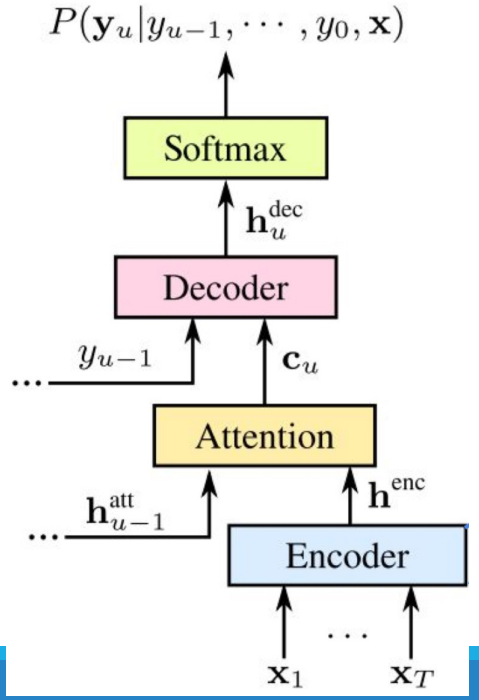
Listen, Attend and Spell

Let **x** be our acoustic features

Let **y** be the sequence we are trying to model (i.e., character sequence)

$$\mathbf{h} = \text{Listen}(\mathbf{x})$$

$$P(y_i|\mathbf{x}, y_{<i}) = \text{AttendAndSpell}(y_{<i}, \mathbf{h})$$

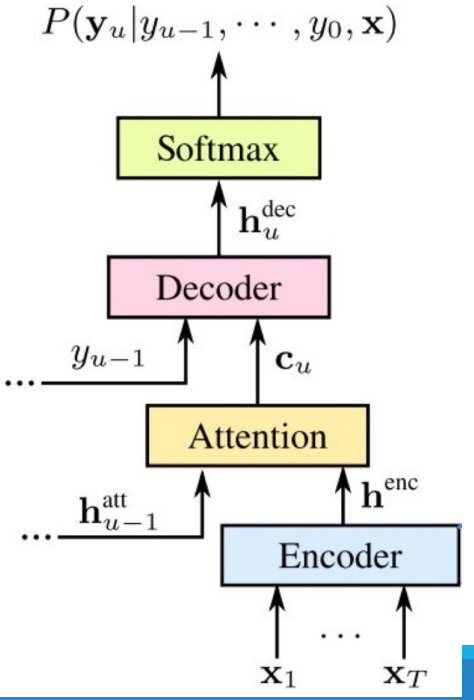


Implicit Language Model

HMM/CTC have conditional independence assumption

seq2seq models have a conditional dependence on the previously emitted symbols:

$$P(y_i|\mathbf{x}, y_{<i})$$



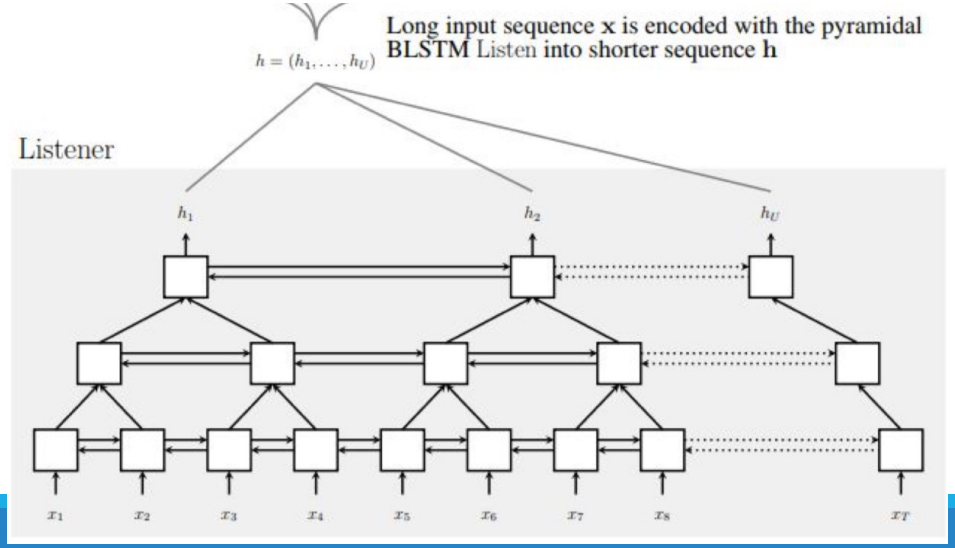
Listen

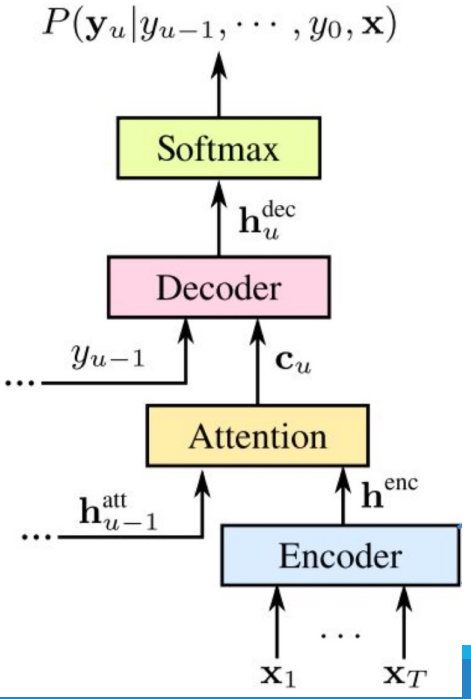
Listen(**x**) can be a RNN (i.e., LSTM).

- Transform our input features x into some higher level feature **h**

$$h_i^j = \text{BLSTM}(h_{i-1}^j, h_i^{j-1}) \quad \text{or} \quad h_i^j = \text{pBLSTM}(h_{i-1}^j, \left[h_{2i}^{j-1}, h_{2i+1}^{j-1}\right])$$

- Here, pBLSTM = pyramid BLSTM

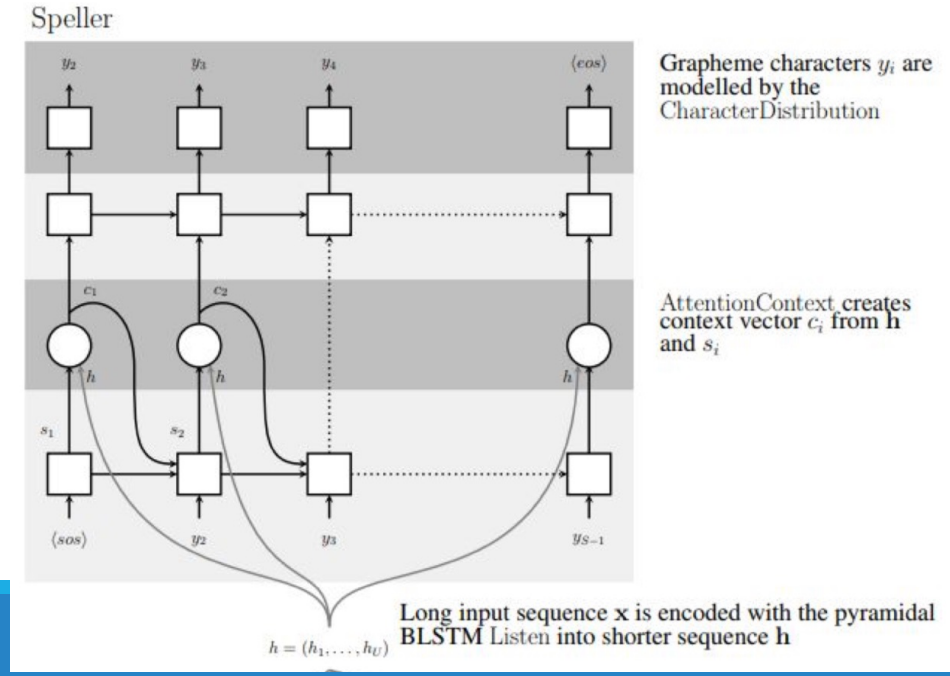




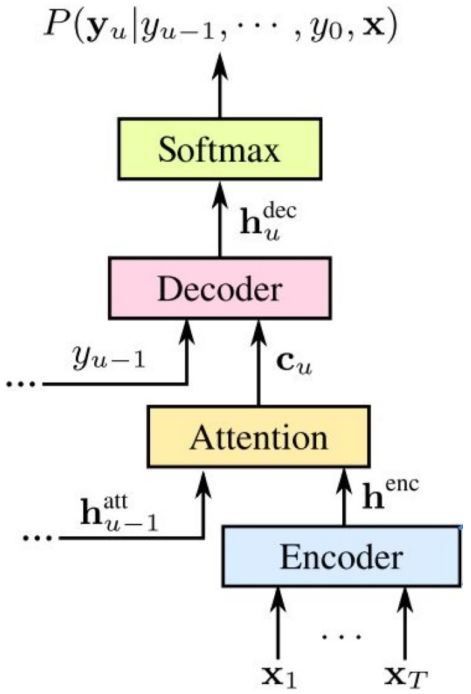
Attend and Spell

AttendAndSpell(**h**) is an attention-based RNN decoder

- content: $\boldsymbol{c}_i = \text{AttentionContext}(\boldsymbol{s}_i, \mathbf{h})$
- decoder state: $\boldsymbol{s}_i = \text{RNN}(\boldsymbol{s}_{i-1}, \boldsymbol{y}_{i-1}, \boldsymbol{c}_{i-1})$
- Softmax Ouput: $P(\boldsymbol{y}_i|\mathbf{x}, \boldsymbol{y}_{<i}) = \text{CharacterDistribution}(\boldsymbol{s}_i, \boldsymbol{c}_i)$



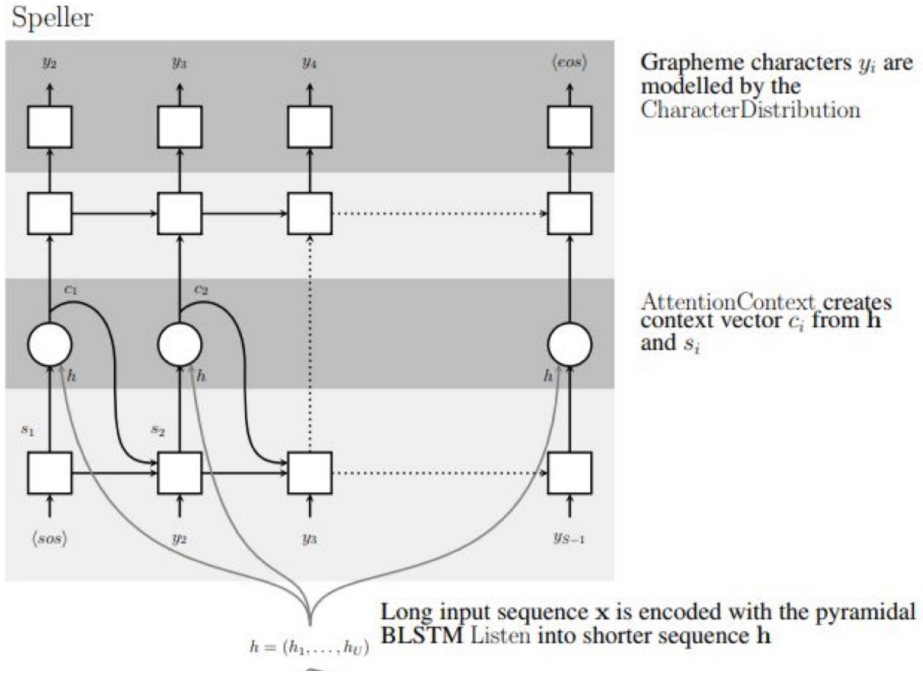
MLP with softmax outputs over characters



Attend

The AttentionContext(s_i,**h**) creates an alignment and context for each timestep

- at each decoder timestep ***i***, the AttentionContext function computes the scalar energy $e_{i,u}$ for each encoder timestep ***u***, using vector $h_u \in \boldsymbol{h}$ and s_i

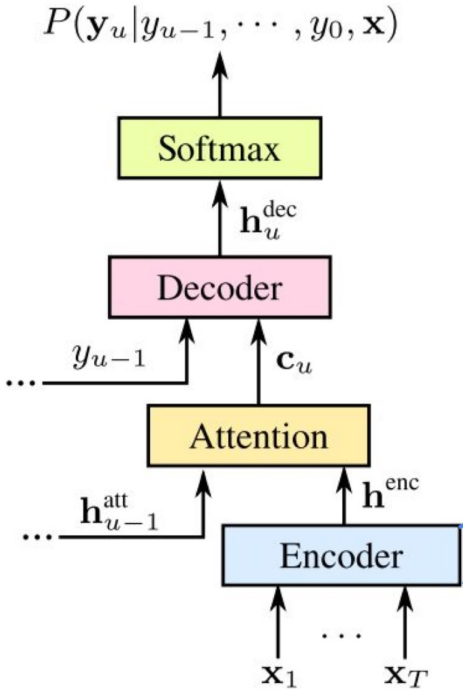


Dot-Product:
ϕ and ψ are MLP networks

$$e_{i,u} = \langle \phi(s_i), \psi(h_u) \rangle$$

$$\alpha_{i,u} = \frac{\exp(e_{i,u})}{\sum_{u'} \exp(e_{i,u'})}$$

$$c_i = \sum_u \alpha_{i,u} h_u$$

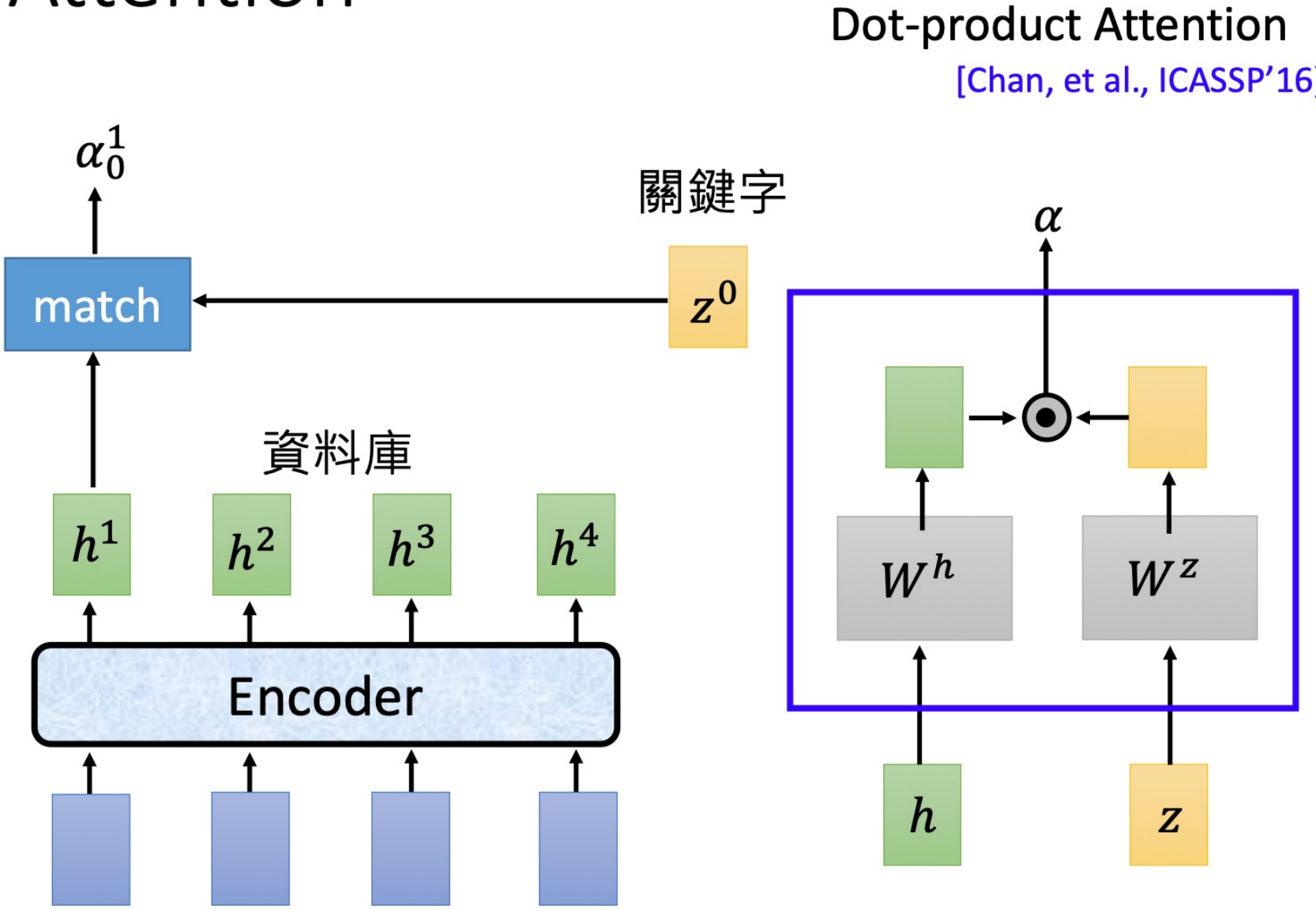


Popular attention mechanisms & their alignment score functions

Name	Alignment score function	Citation
Content-base attention	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \text{cosine}[\boldsymbol{s}_t, \boldsymbol{h}_i]$	Graves2014
Additive(*)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a[\boldsymbol{s}_t; \boldsymbol{h}_i])$	Bahdanau2015
Location-Base	$\alpha_{t,i} = \text{softmax}(\mathbf{W}_a \boldsymbol{s}_t)$ Note: This simplifies the softmax alignment to only depend on the target position.	Luong2015
General	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \mathbf{W}_a \boldsymbol{h}_i$ where $\mathbf{W}_a$ is a trainable weight matrix in the attention layer.	Luong2015
Dot-Product	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \boldsymbol{s}_t^\top \boldsymbol{h}_i$	Luong2015
Scaled Dot-Product(^)	$\text{score}(\boldsymbol{s}_t, \boldsymbol{h}_i) = \frac{\boldsymbol{s}_t^\top \boldsymbol{h}_i}{\sqrt{n}}$ Note: very similar to the dot-product attention except for a scaling factor; where n is the dimension of the source hidden state.	Vaswani2017

https://lilianweng.github.io/lil-log/2018/06/24/attention-attention.html

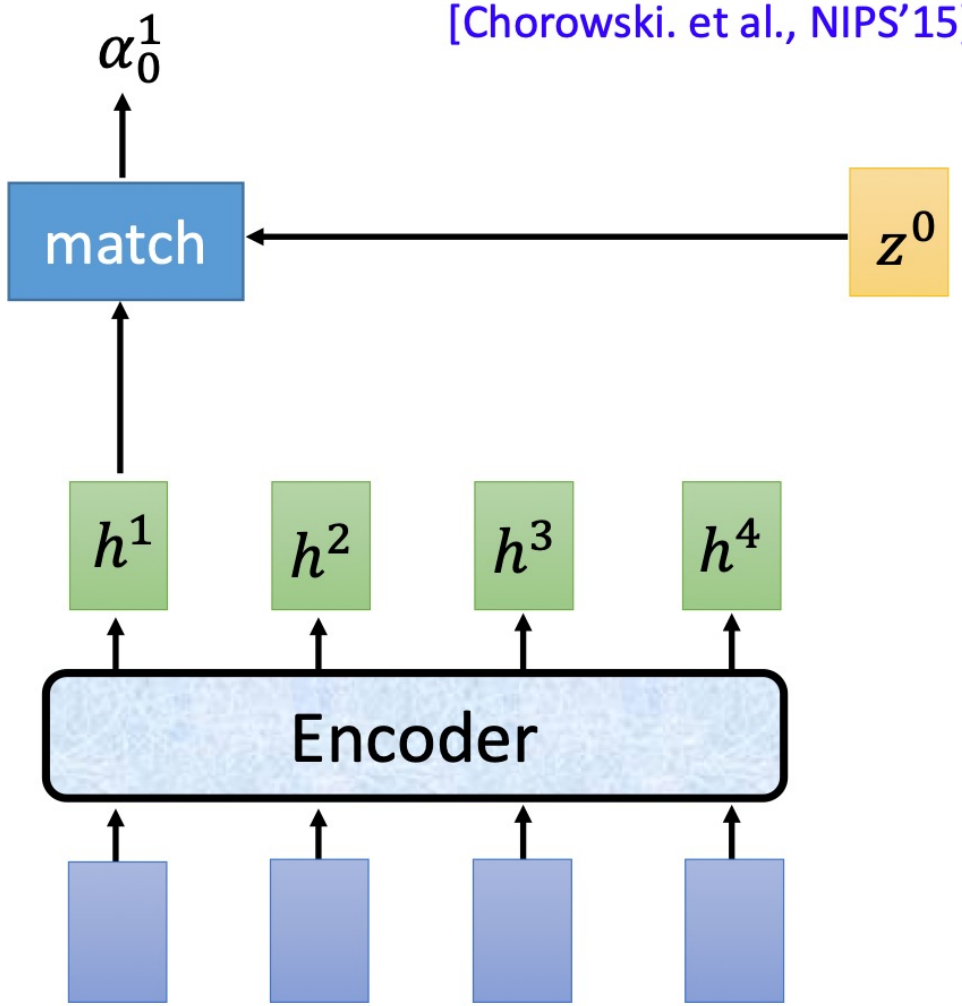
Attention

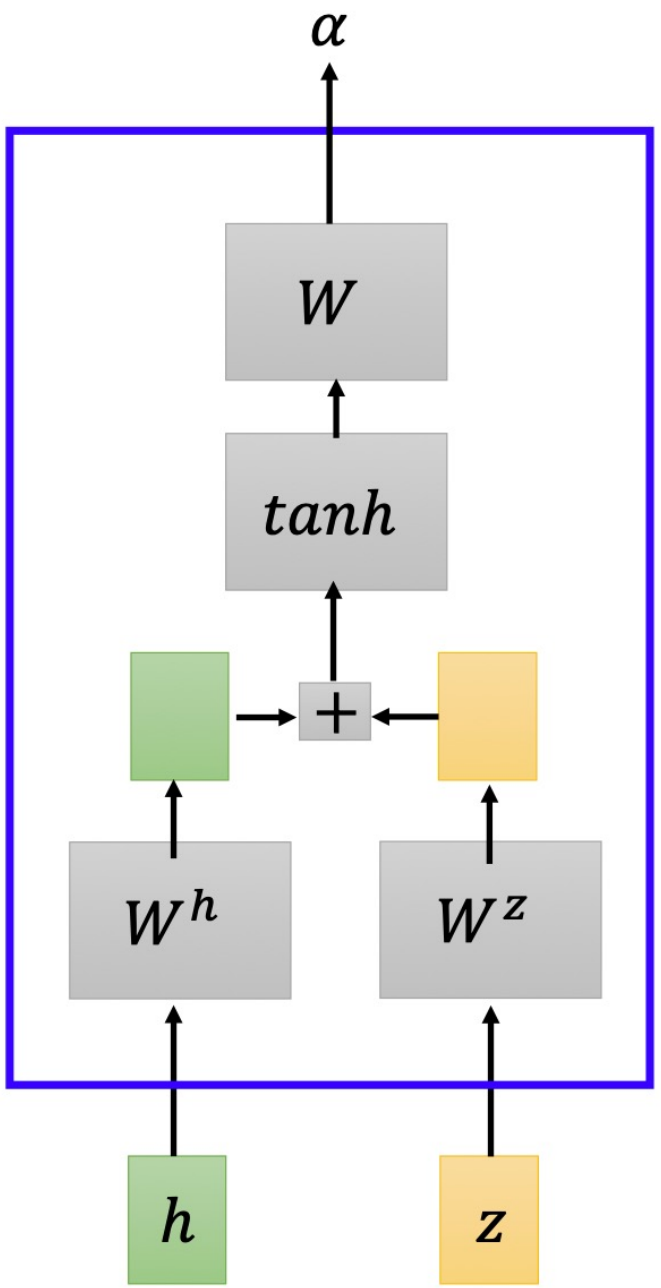

Dot-product Attention
[Chan, et al., ICASSP'16]
α_0^1
match
關鍵字
z^0
資料庫
h^1
h^2
h^3
h^4
Encoder
α
W^h
W^z
h
z

Attention

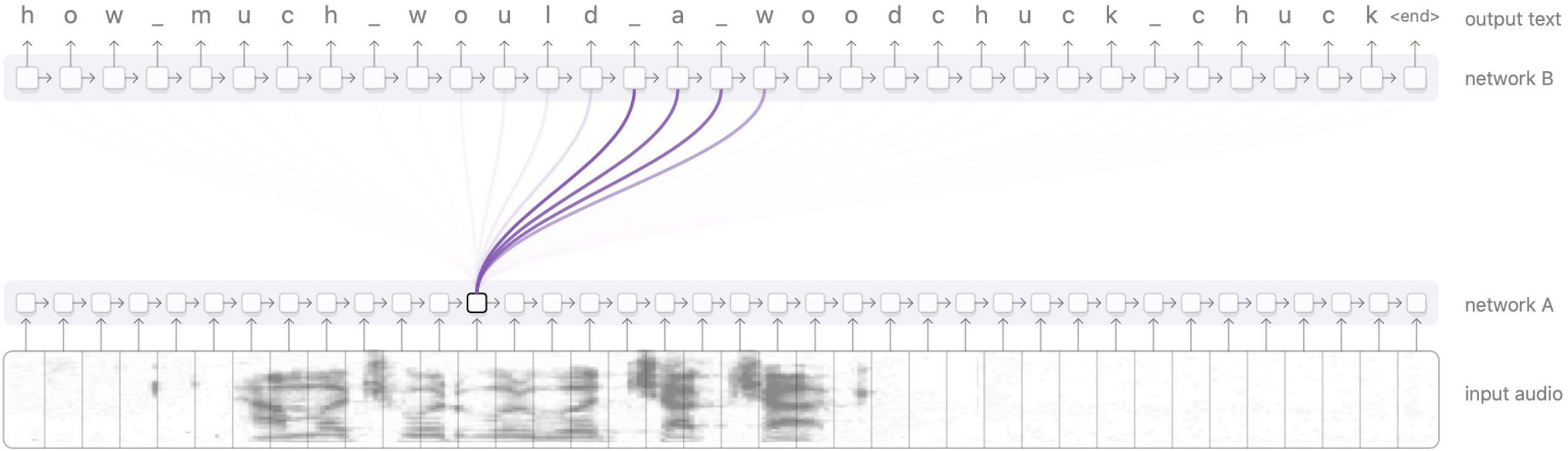
Additive Attention

[Chorowski. et al., NIPS'15]



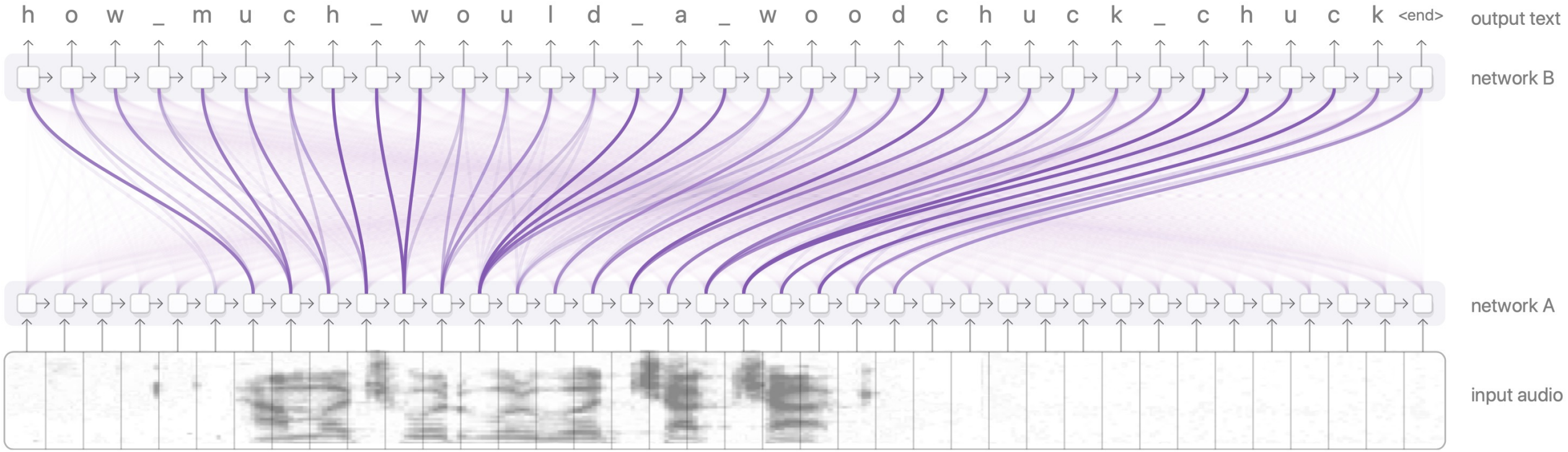


AttentionContext(s_i,h_u)



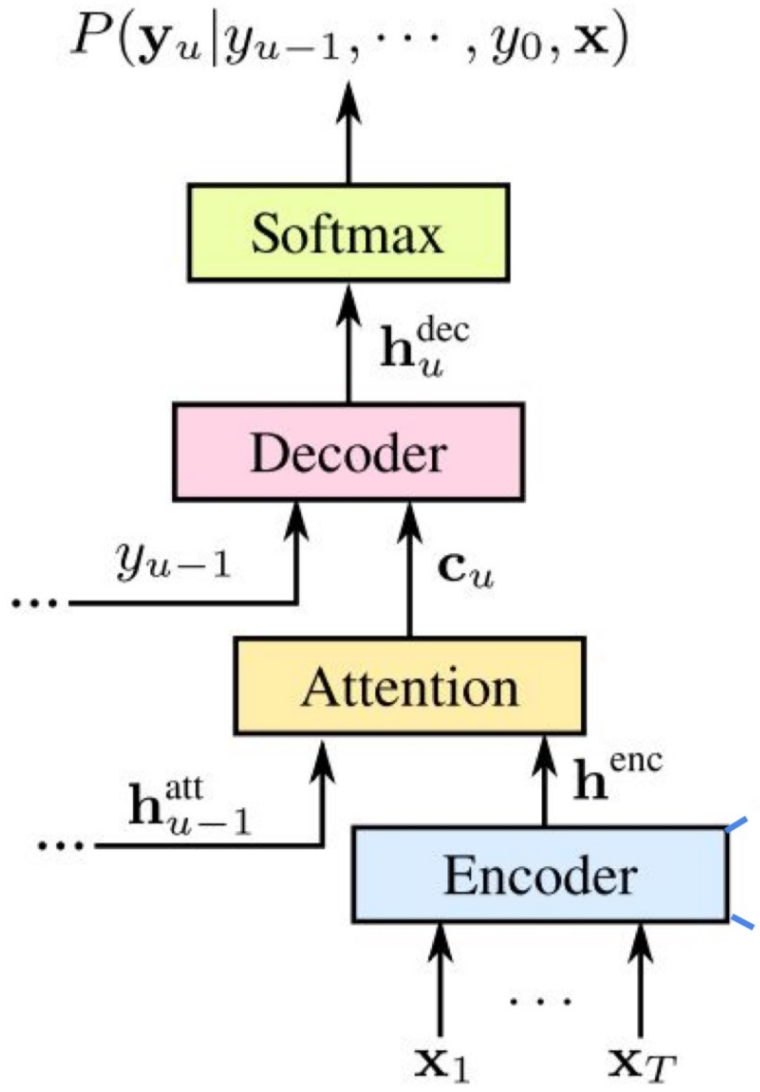
https://distill.pub/2016/augmented-rnns/

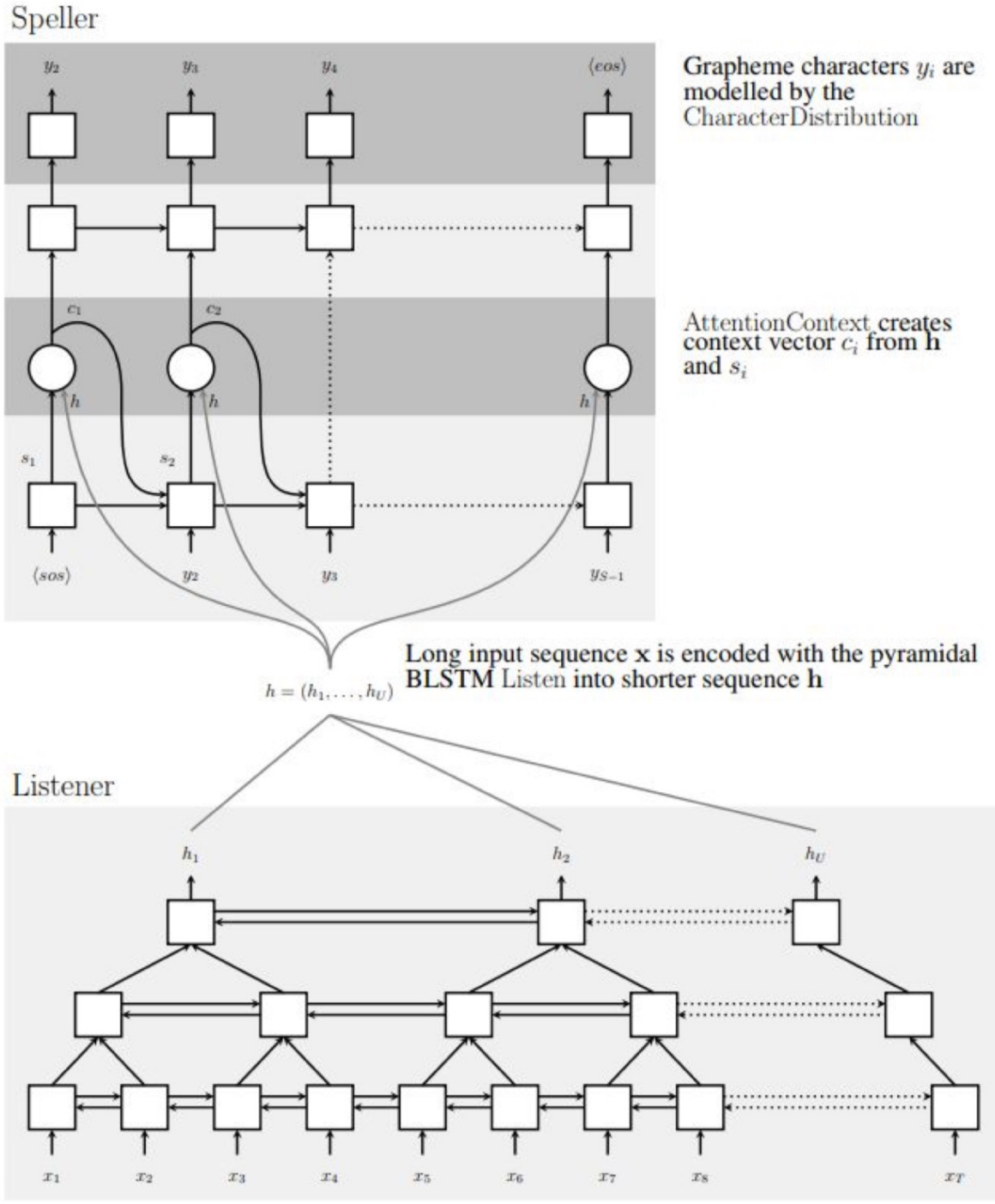
AttentionContext(s_i,h_u)



https://distill.pub/2016/augmented-rnns/

Listen, Attend and Spell





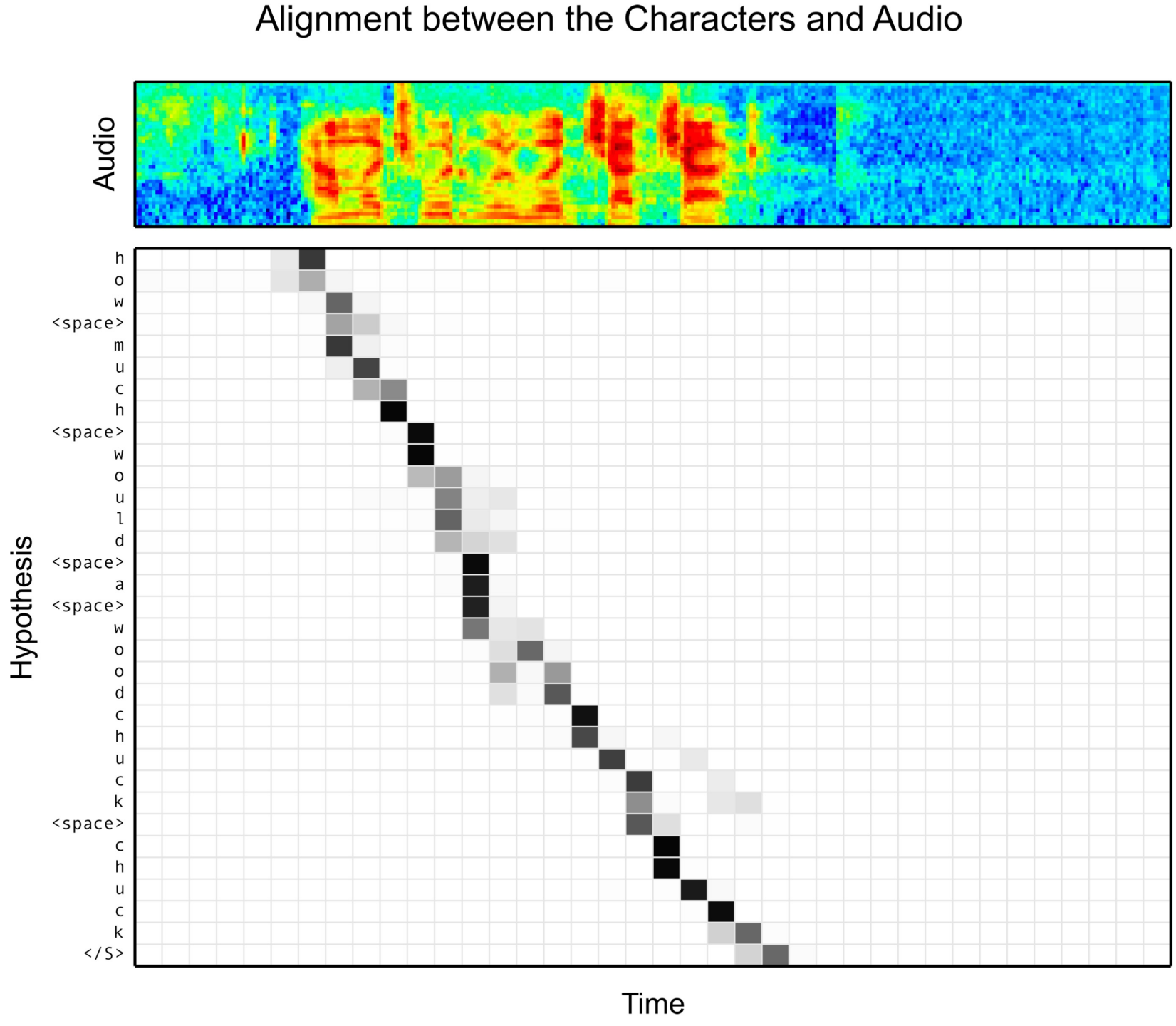
Listen, Attend and Spell

Attention mechanism creates a short circuit between each decoder's output and the acoustic

More efficient information/gradient flow!

Creates an explicit alignment between each character and the acoustic features

- CTC's alignment is latent

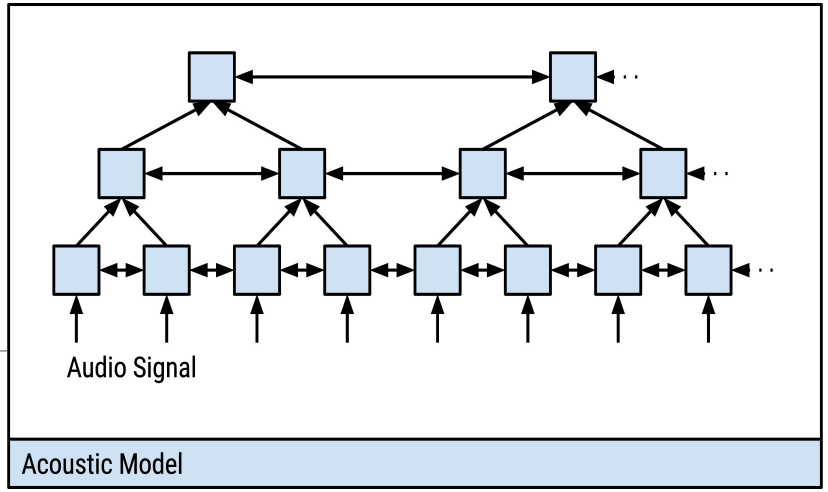

Alignment between the Characters and Audio
Audio
Hypothesis
h
o
w
<space>
m
u
c
h
<space>
w
o
u
l
d
<space>
a
<space>
w
o
o
d
c
h
u
c
k
<space>
c
h
u
c
k
</S>
Time

Issue #1

Model works but...

- Takes “forever” to train, after > 1 month model still not converged : (
- WERs in the >20s (CLDNN-HMM is 8ish)
- Attention mechanism must focus on a long range of frames

Pyramid



Build higher level features with each layer

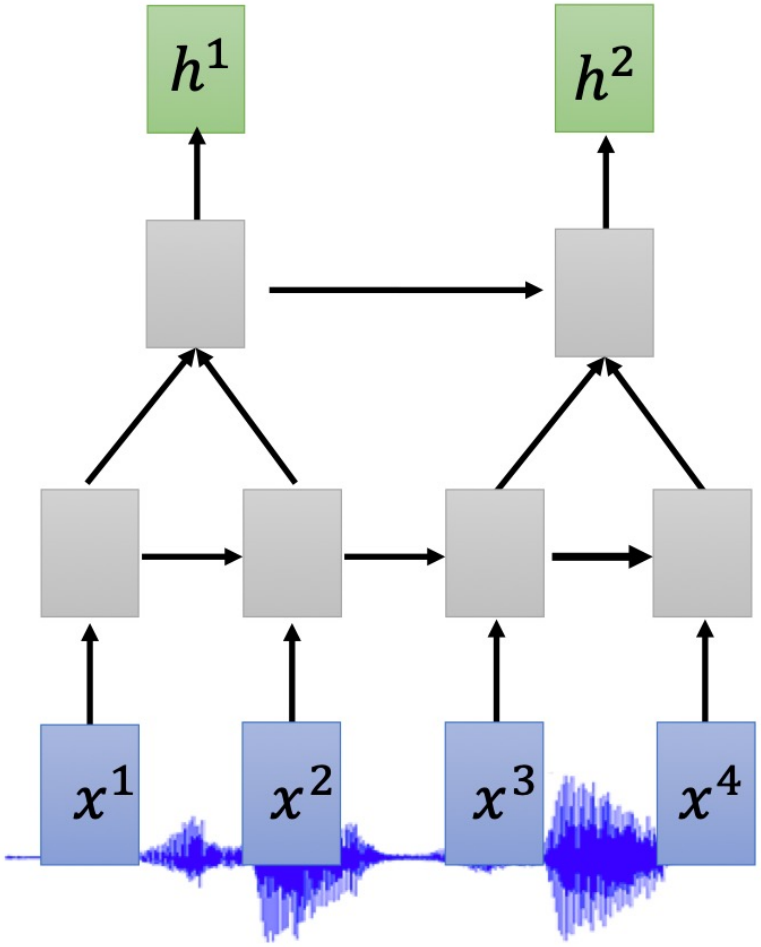
Reduce number of timesteps for attention to attend to

Computational efficiency

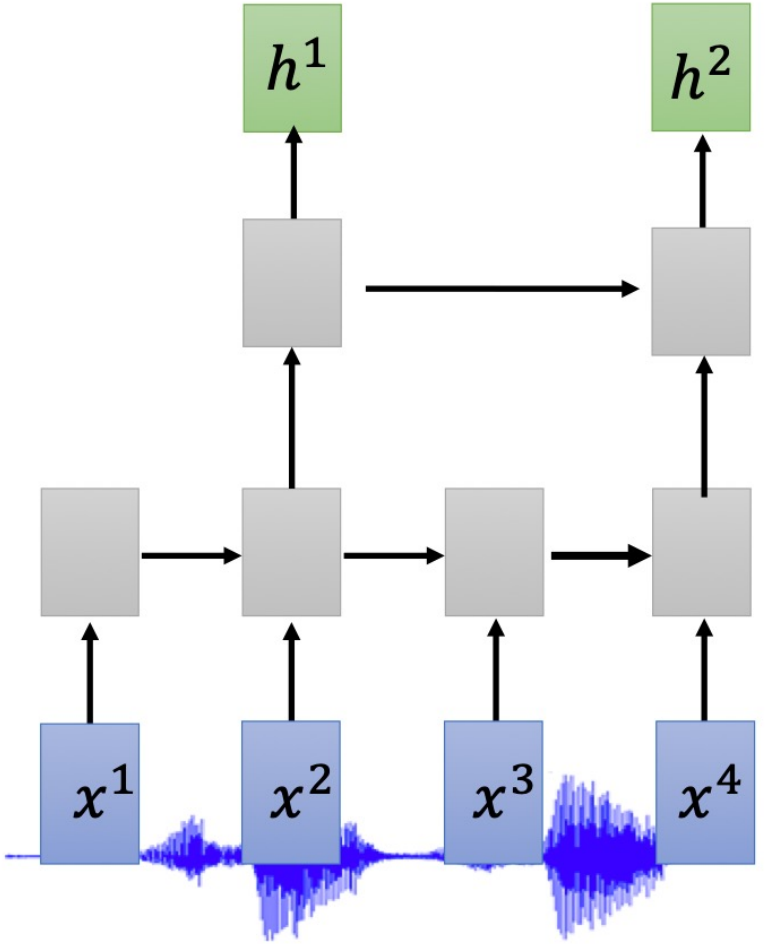
8 filterbank frames → 1 pyramid frame feature

$$h_i^j = \text{pBLSTM}(h^j_{i-1},\ h_{2i}^{j-1}, h_{2i+1}^{j-1}\ )$$

Listen – Down Sampling



Pyramid RNN [Chan, et al., ICASSP'16]



Pooling over time [Bahdanau. et al., ICASSP'16]

Issue #2

Model works better but...

- 16 to 20-ish WERs (w/o LM)
- Takes around 2-3 weeks to train, overfitting is a HUGE problem
- Mismatch between train and inference conditions

Sampling Trick - *Teacher Forcing*

Training is conditioned on ground truth y^*

- But we don't have access to ground truth during inference!

$$\max_\theta \sum_i \log P(y_i|\mathbf{x}, y^*_{>i}; \theta)$$

Sample from our model

- Condition on sample for next step prediction $\tilde{y}_i$

$$\tilde{y}_i \sim \text{CharacterDistribution}(s_i, c_i) \qquad \max_\theta \sum_i \log P(y_i|\mathbf{x}, \tilde{y}_{>i}; \theta)$$

Listen, Attend and Spell

$$P(\mathbf{y}|\mathbf{x}) = \prod_i P(y_i|\mathbf{x}, y_{<i})$$

$$\mathbf{h} = \text{Listen}(\mathbf{x})$$

$$P(\mathbf{y}|\mathbf{x}) = \text{AttendAndSpell}(\mathbf{h}, \mathbf{y})$$

Language Model Rescoring

Leverage on vast quantities of text!

Normalize our LAS model by number of characters in utterance → LAS has bias for short utterances

$$s(\mathbf{y}, \mathbf{x}) = \frac{\log P(\mathbf{y}|\mathbf{x})}{|\mathbf{y}|_c} + \lambda \log P_{\mathrm{LM}}(\mathbf{y})$$

Experiments

Dataset

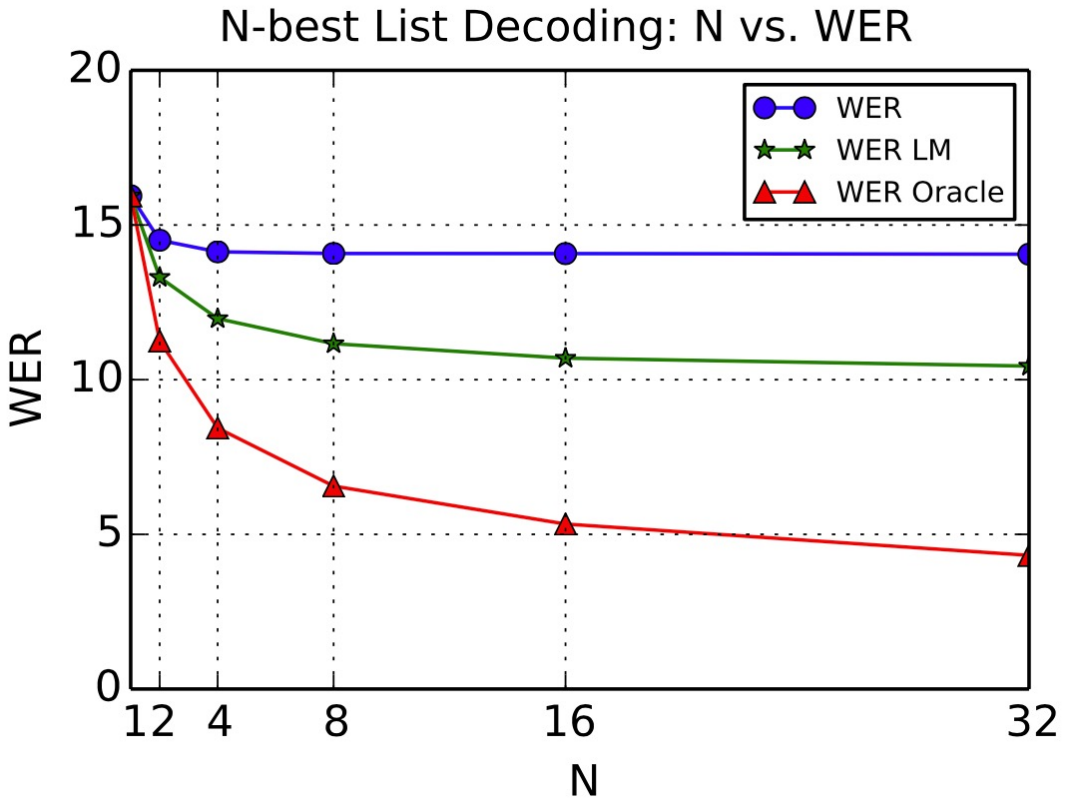
- Google voice search
  - 2000 hrs, 3M training utterances
  - 16 hrs, 22K test utterances
  - Mixed Room Simulator, artificially increase acoustic data by x20 (i.e., YouTube and environmental noise)
  - Clean and noisy test set

Training

- Stochastic Gradient Descent
- DistBelief 32 replicas, minibatch size of 32
- 2-3 weeks of training time

Experimental Results

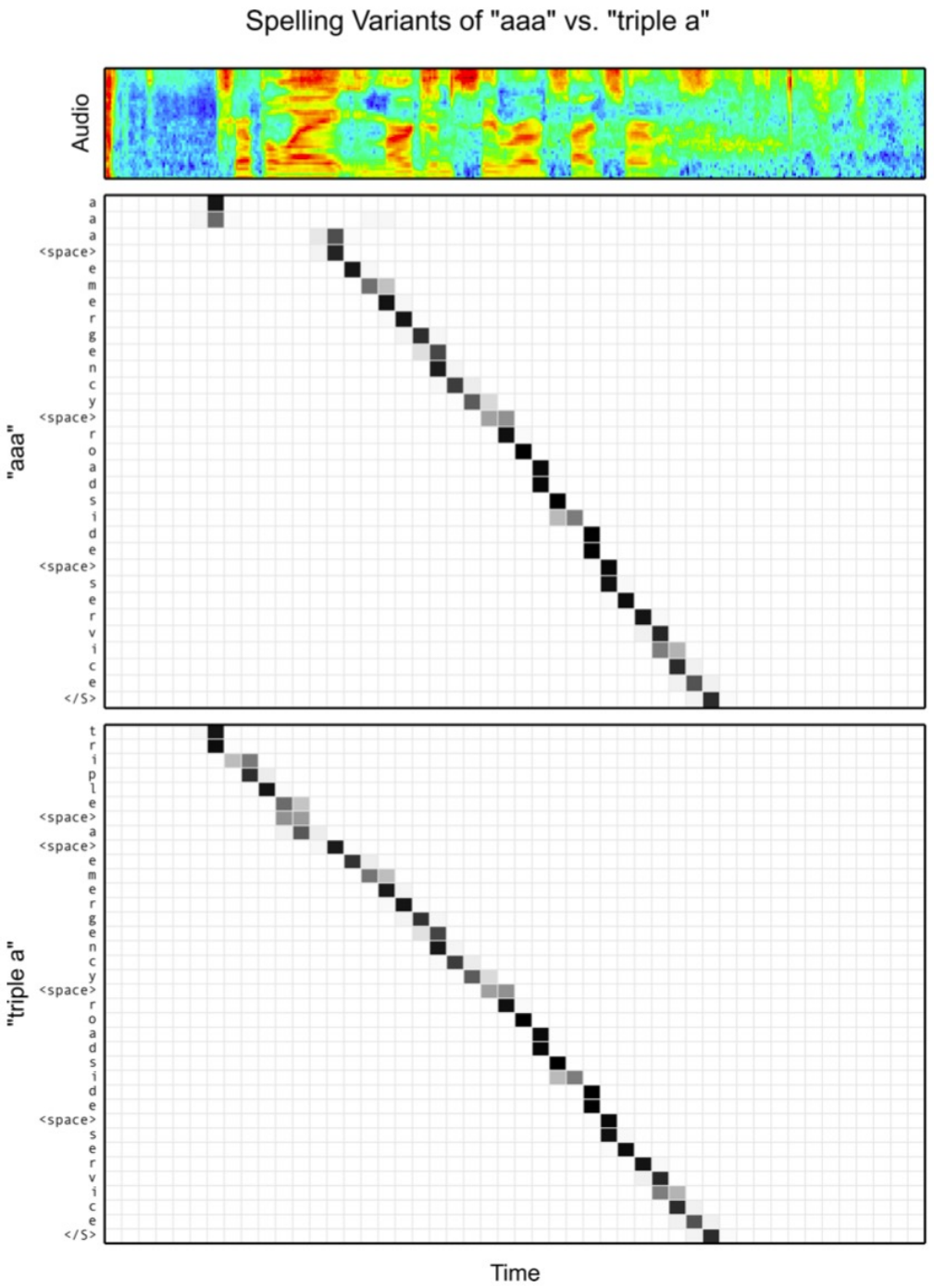
Model	Clean WER	Noisy WER
CLDNN-HMM (Tara et al., 2015)	8.0	8.9
LAS	16.2	19.0
LAS + LM	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM	10.3	12.0



- 16% WER without any searching (and LM) - just take the greedy path!
- Didn't decode with a dictionary!
  - LAS implicitly learnt the dictionary during training
  - Rare spelling mistakes!
- Didn't decode with a LM! (only rescored)
  - n-best list decoding where n = 32

Results: AAA vs. Triple A

N	Text	$\log P$	WER
Truth	call aaa roadside assistance	-	-
1	call aaa roadside assistance	-0.57	0.0
2	call triple a roadside assistance	-1.54	50.0
3	call trip way roadside assistance	-3.50	50.0
4	call xxx roadside assistance	-4.44	25.0



Conclusion

Listen, Attend and Spell (LAS)

- End-to-end speech recognition model
- No conditional independence, Markovian assumptions, or proxy problems
- Sequence-to-Sequence + Attention + Pyramid
- One model: integrate all traditional components of an ASR system into one model (acoustic, pronunciation, language, etc...)

Competitive to state-of-the-art CLDNN-HMM system

- 10.3 vs. 8.0 WER

Time to throw away HMM and phonemes!

Model	Clean WER	Noisy WER
CLDNN-HMM [22]	8.0	8.9
LAS	14.1	16.5
LAS + LM Rescoring	10.3	12.0

2000 hours

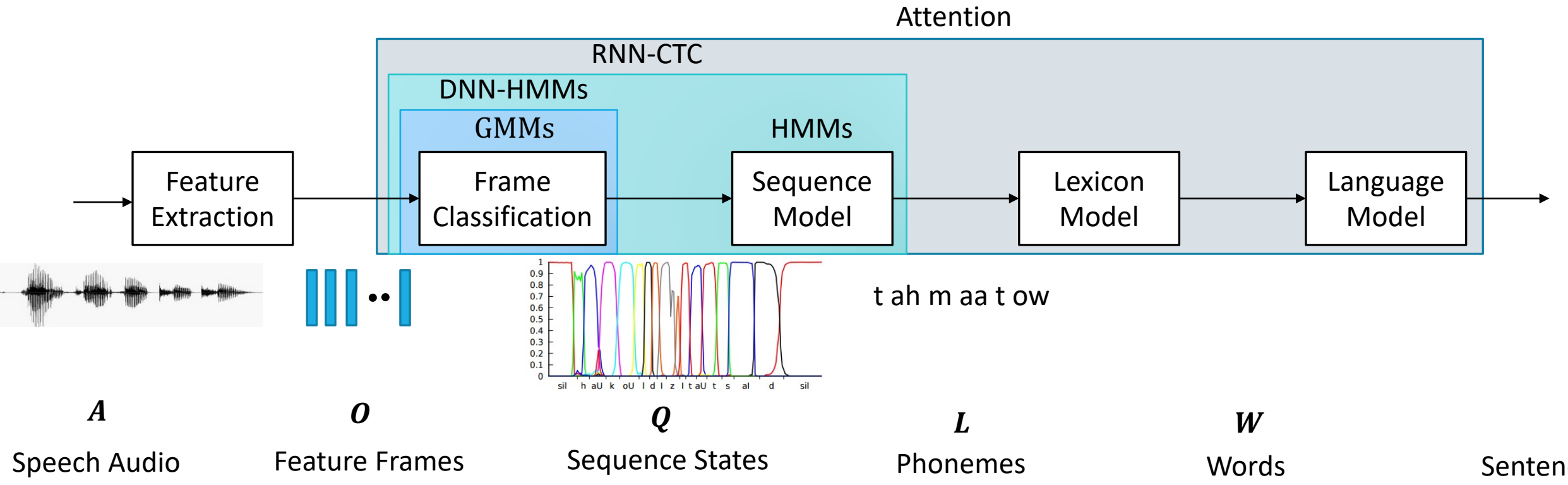
[Chan, et al., ICASSP'16]

Exp-ID	Model	VS/D	1st pass Model Size
E8	Proposed	**5.6/4.1**	**0.4 GB**
E9	Conventional LFR system	6.7/5.0	0.1 GB (AM) + 2.2 GB (PM) + 4.9 GB (LM) = 7.2GB

12500 hours

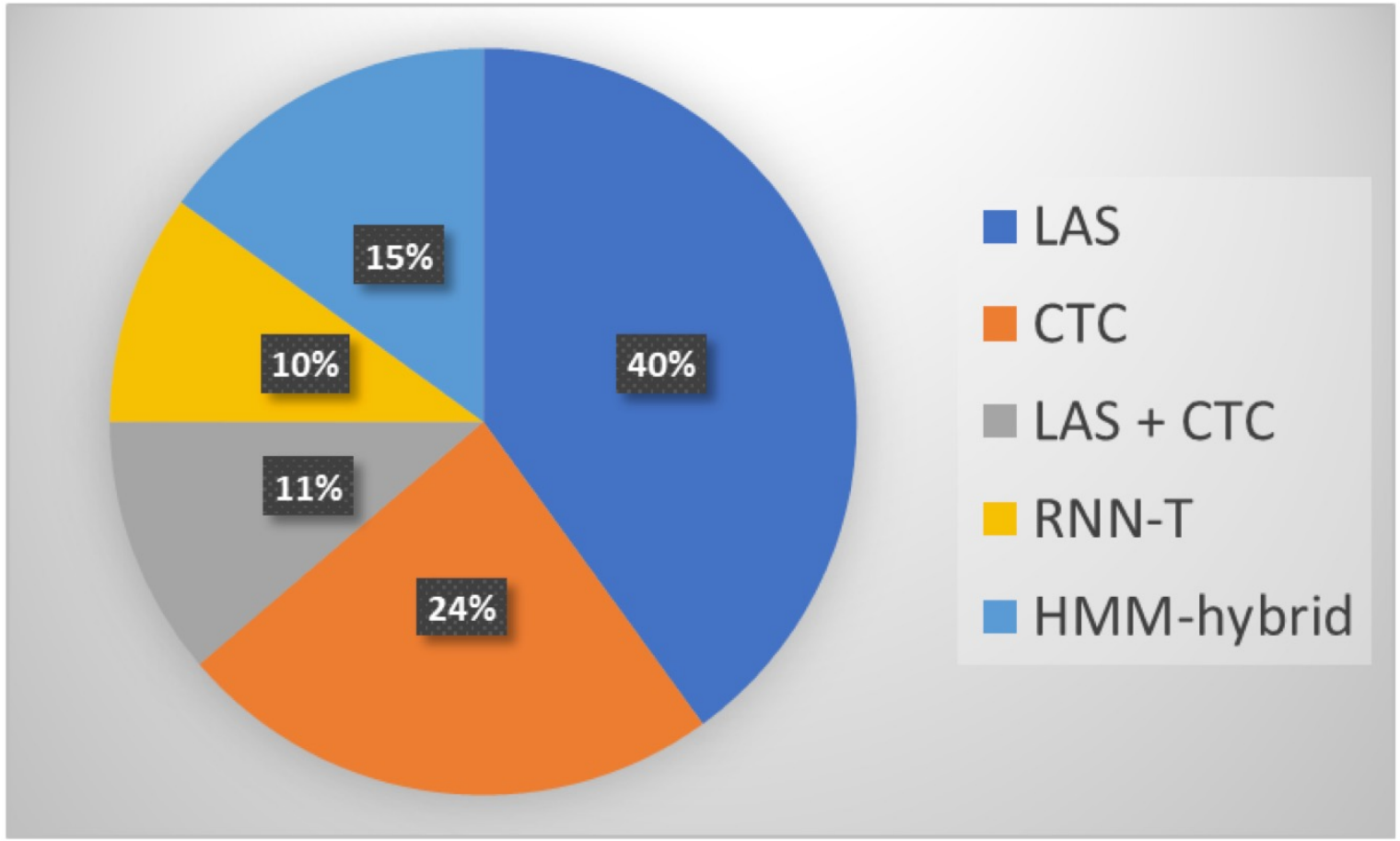
[Chiu, et al., ICASSP, 2018]

Deep learning in Speech Recognition

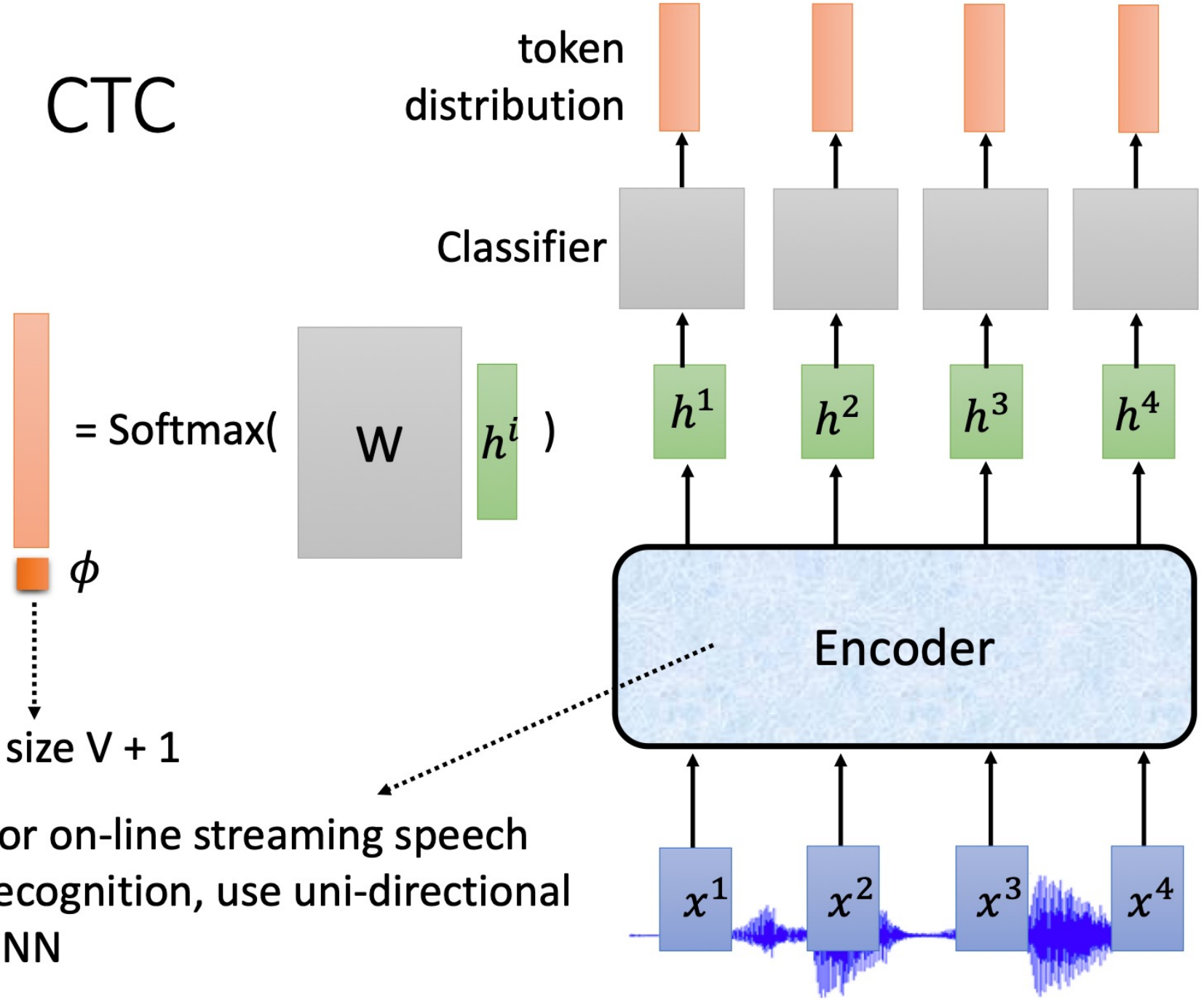


ASR Models

Go through more than 100 papers in INTERSPEECH'19, ICASSP'19, ASRU'19

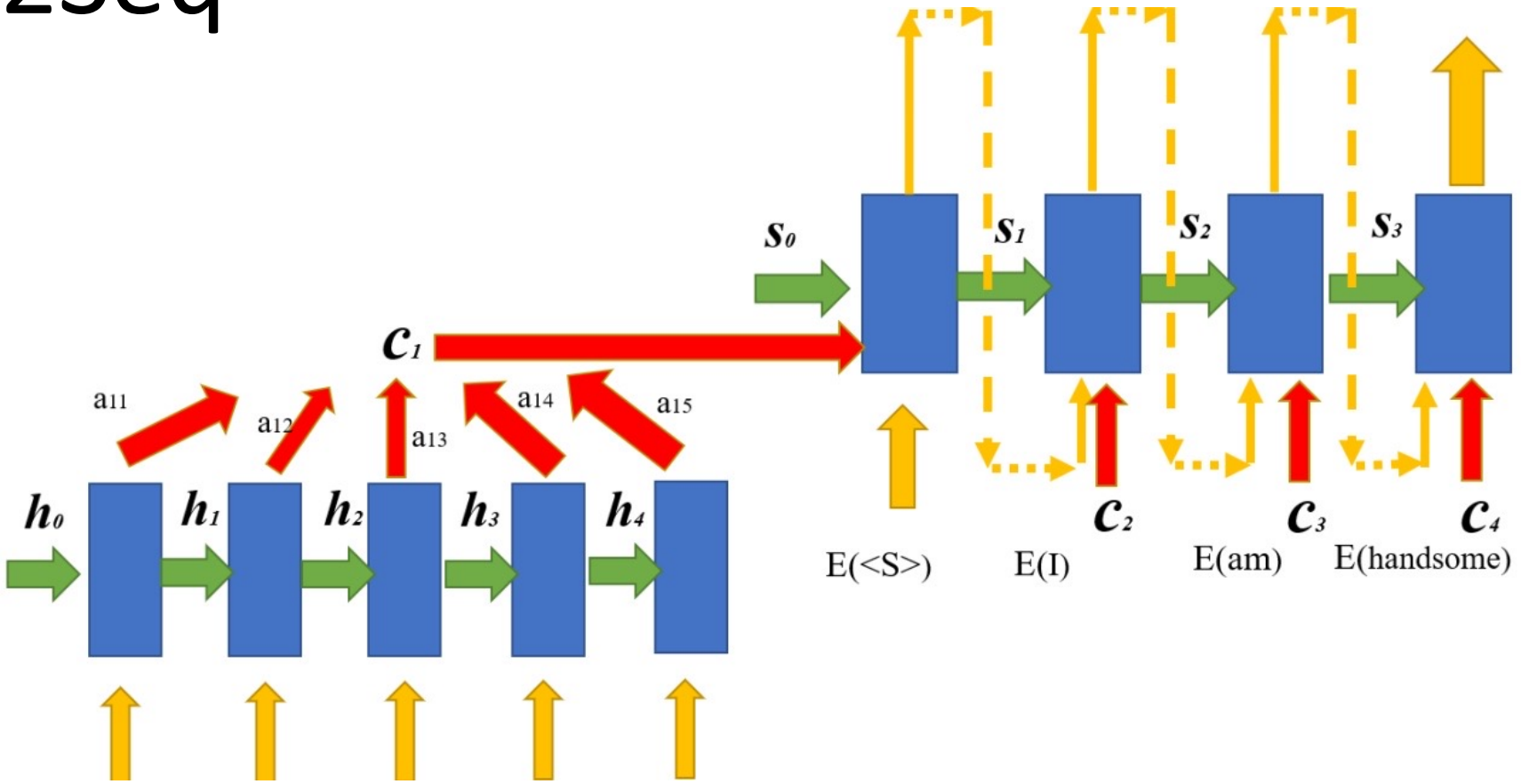


CTC



For on-line streaming speech recognition, use uni-directional RNN

Seq2Seq


s_0
s_1
s_2
s_3
c_1
a_{11}
a_{12}
a_{13}
a_{14}
a_{15}
h_0
h_1
h_2
h_3
h_4
c_2
c_3
c_4
E(<S>)
E(I)
E(am)
E(handsome)

RNA

Recurrent Neural Aligner

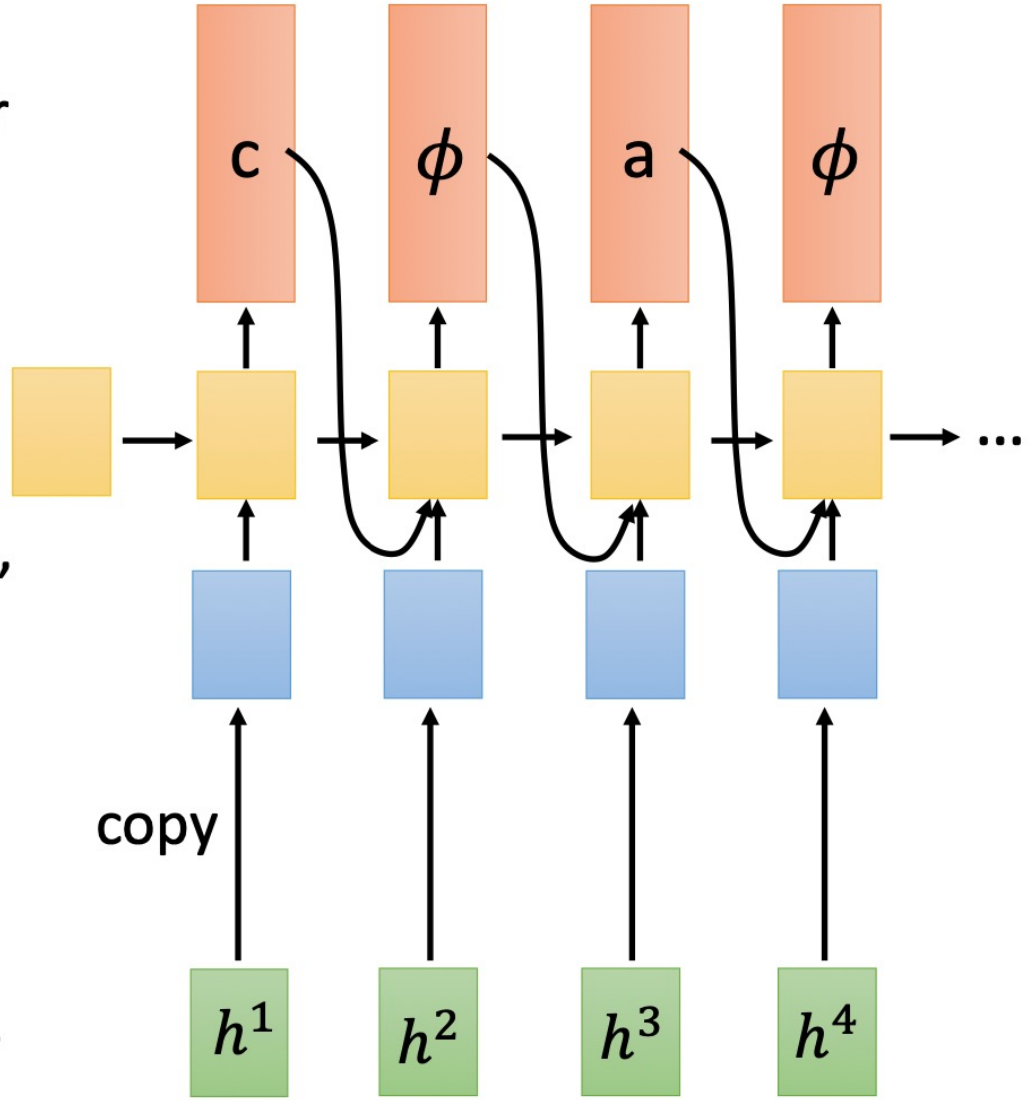
[Sak, et al., INTERSPEECH'17]

CTC Decoder:
take one vector as input,
output one token

RNA adds dependency

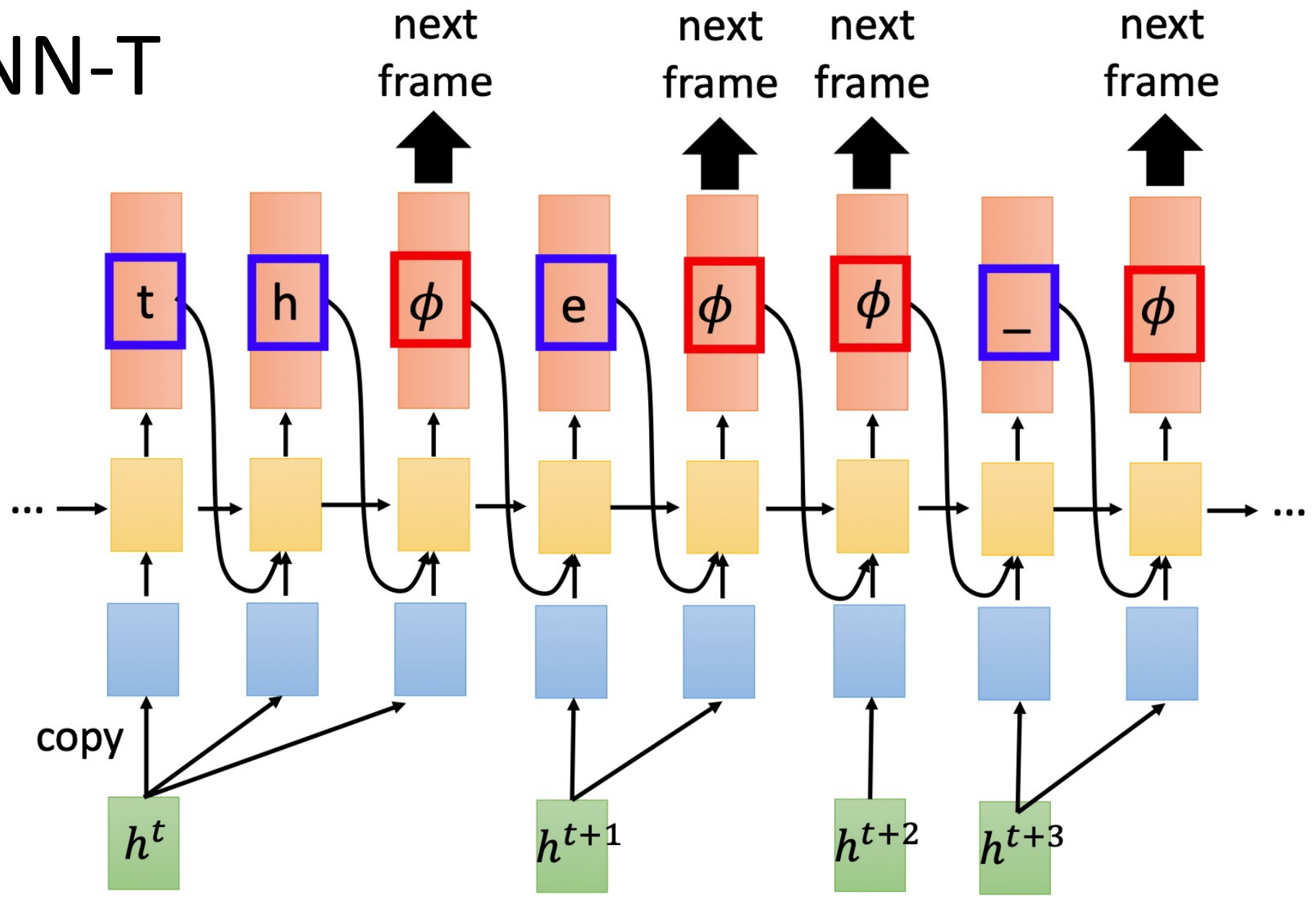
Can one vector map to multiple tokens?

for example, "*th*"

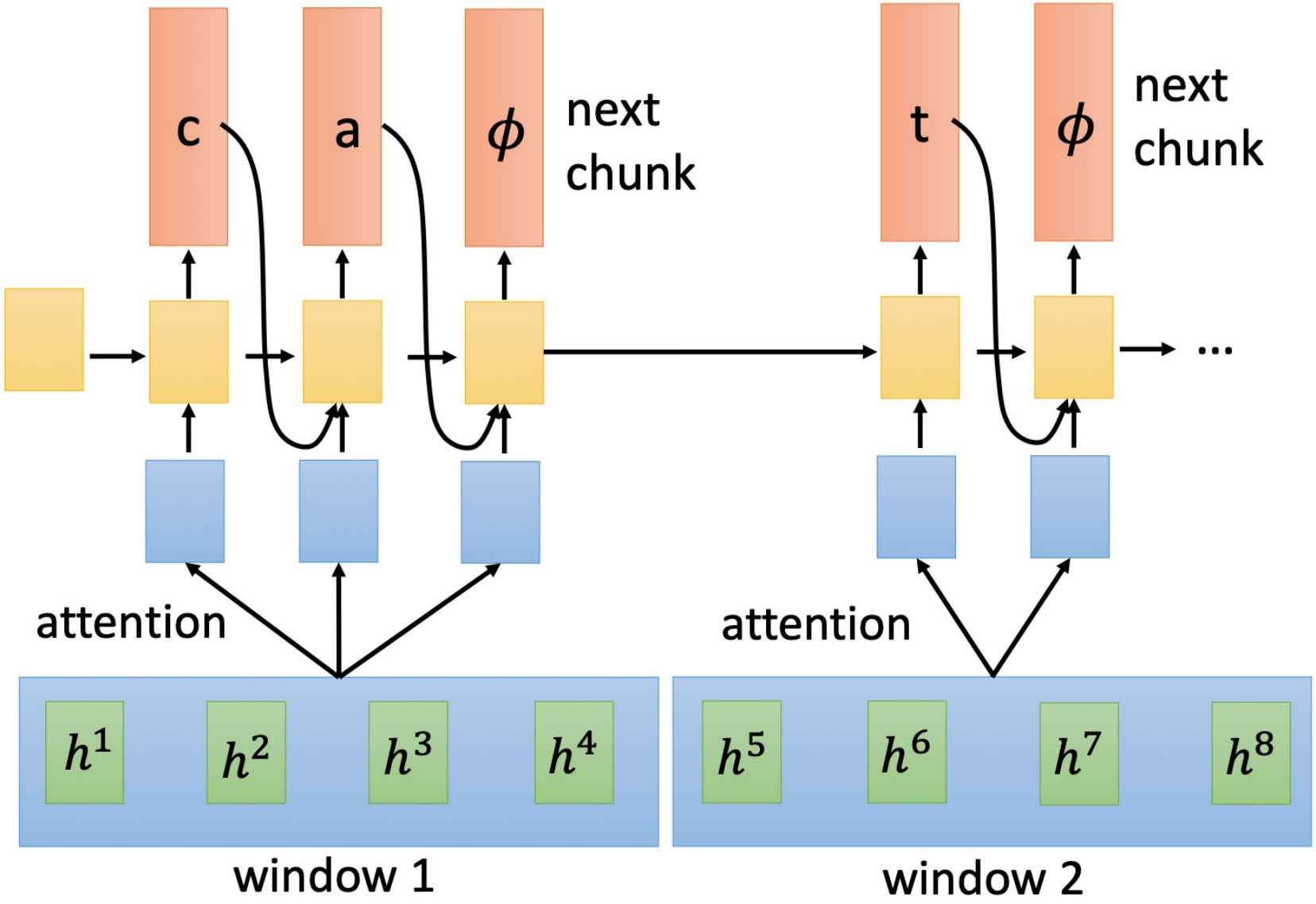


There are T "ϕ" in the output.

RNN-T

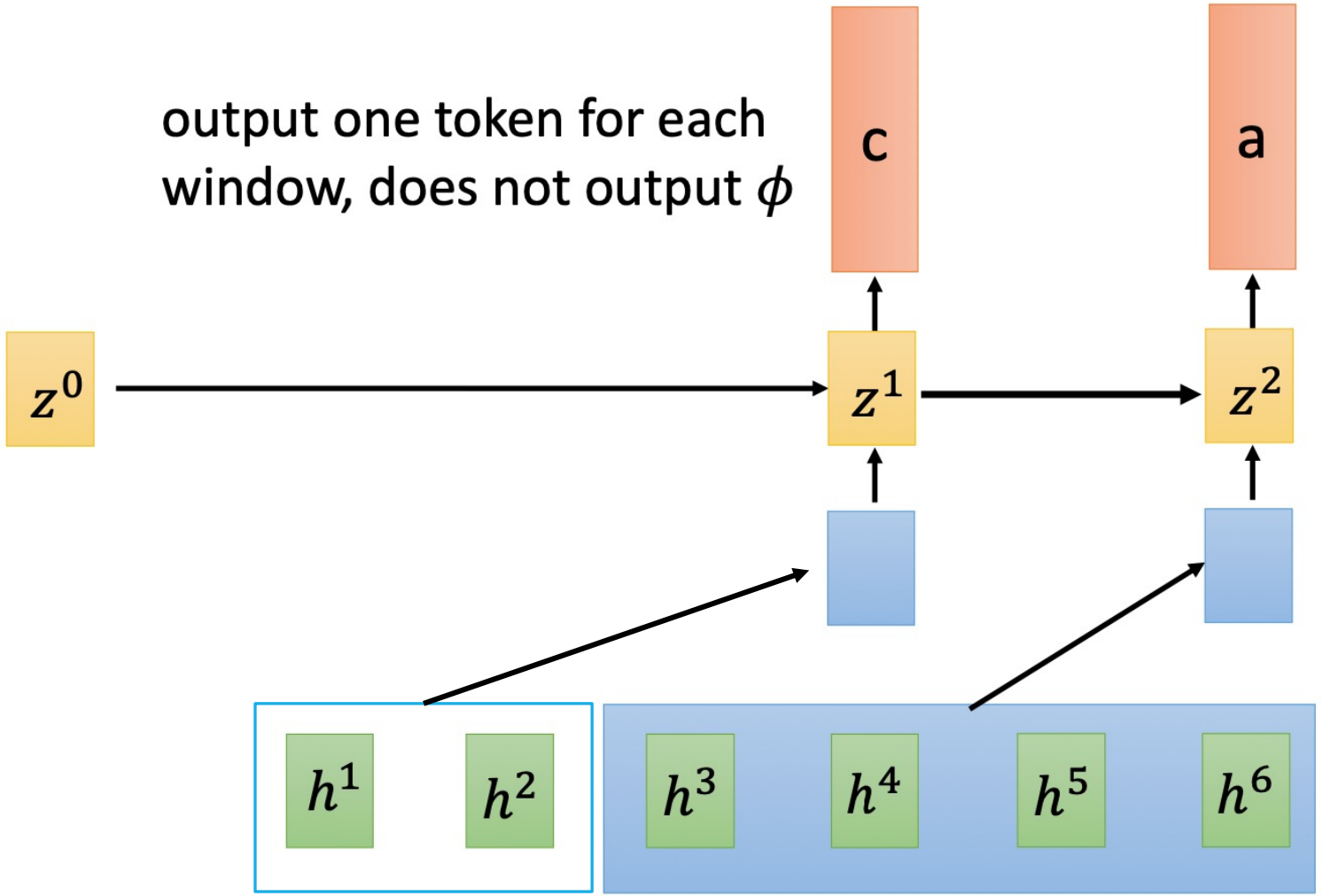


Neural Transducer



MoChA

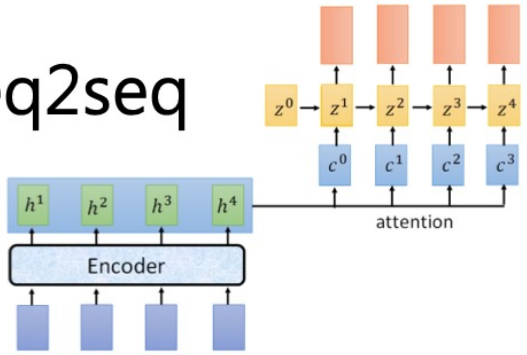
Monotonic Chunkwise Attention



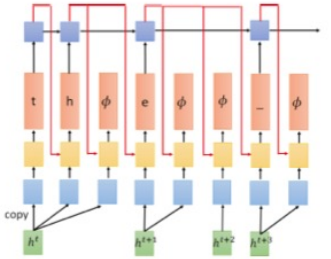
dynamically shift the window

Summary

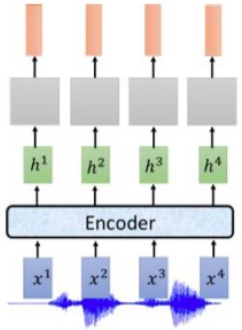
LAS: 就是 seq2seq



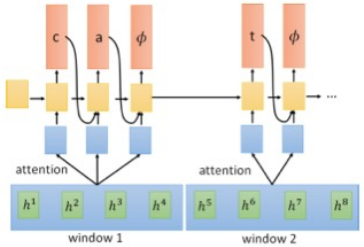
RNN-T: 輸入一個東西可以輸出多個東西的 seq2seq



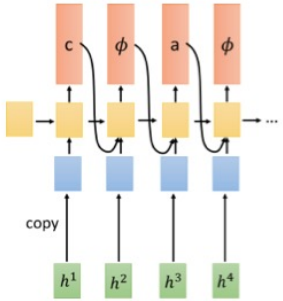
CTC: decoder 是 linear classifier 的 seq2seq



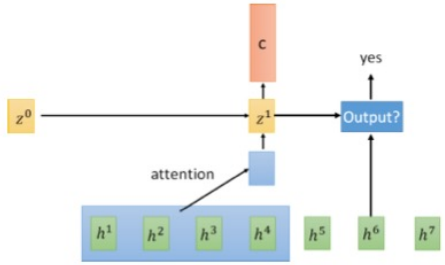
Neural Transducer: 每次輸入一個 window 的 RNN-T



RNA: 輸入一個東西就要輸出一個東西的 seq2seq



MoCha: window 移動伸縮自如的 Neural Transducer



Summary

Automatic Speech Recognition (ASR)

- Hybrid HMM/DNN
- Connectionist Temporal Classification (CTC)
- Listen, Attend and Spell