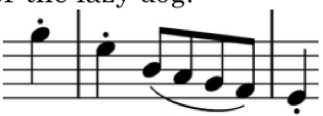
About Sequence Classification for Sound Event Detection and end-to-end ASR

Thomas Pellegrini

Training event 2: Speech Processing and Machine Learning Workshop, Lisbon, February 2019

Outline

Examples of sequence data

Speech recognition		→	"The quick brown fox jumped over the lazy dog."
Music generation	∅	→	
Sentiment classification	"There is nothing to like in this movie."	→	★☆☆☆☆
DNA sequence analysis	AGCCCCTGTGAGGAACTAG	→	AGCCCCTGTGAGGAACTAG
Machine translation	Voulez-vous chanter avec moi?	→	Do you want to sing with me?
Video activity recognition		→	Running
Name entity recognition	Yesterday, Harry Potter met Hermione Granger.	→	Yesterday, Harry Potter met Hermione Granger.

Andrew Ng

Outline

Bird's eye view on Recurrent Neural Networks

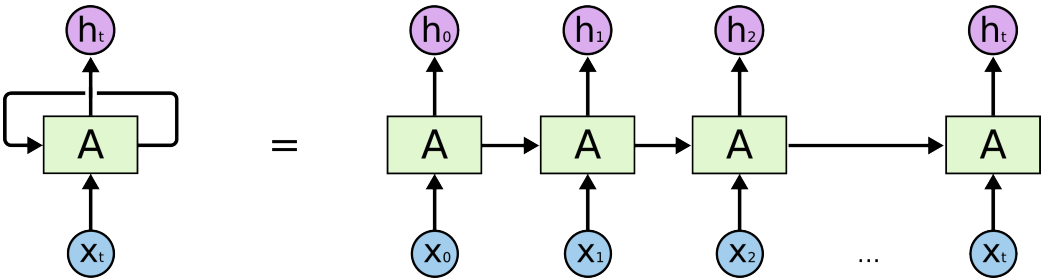
Case study: Weakly-labeled Sound Event Detection

Speech recognition

- Motivation for end-to-end systems
- From sounds to characters: CTC models
- Attention-based encoder-decoder models: Listen, Attend, Spell
- A few words on transfer learning

Generic RNNs

Unrolled RNN



RNN:

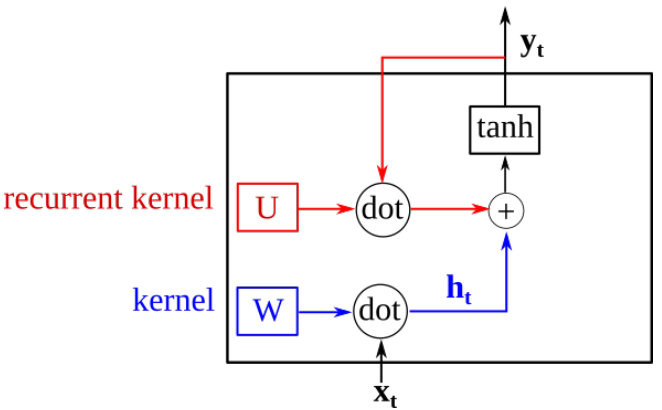
- a hidden state $\mathbf{h}$
- an optional output $\mathbf{y}$ which operates on a variable-length sequence $\mathbf{x} = (x_1, \ldots, x_T)$

At each time step t, $\mathbf{h}_t$ is updated by:

$$\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$$

where f is a non-linear function such as a sigmoid or an LSTM, or a GRU, etc

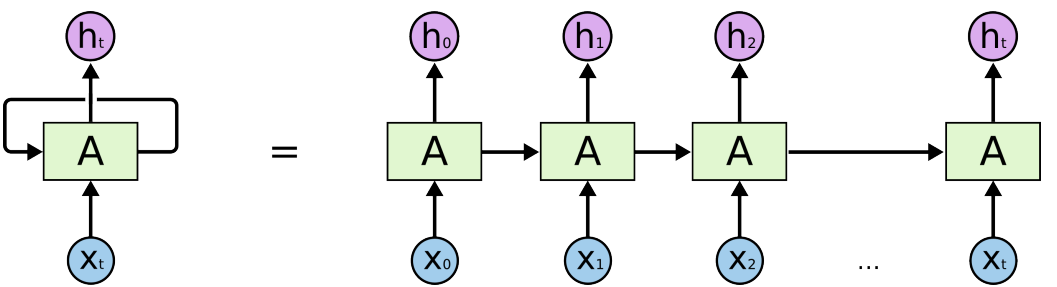
Simple RNN (keras version)



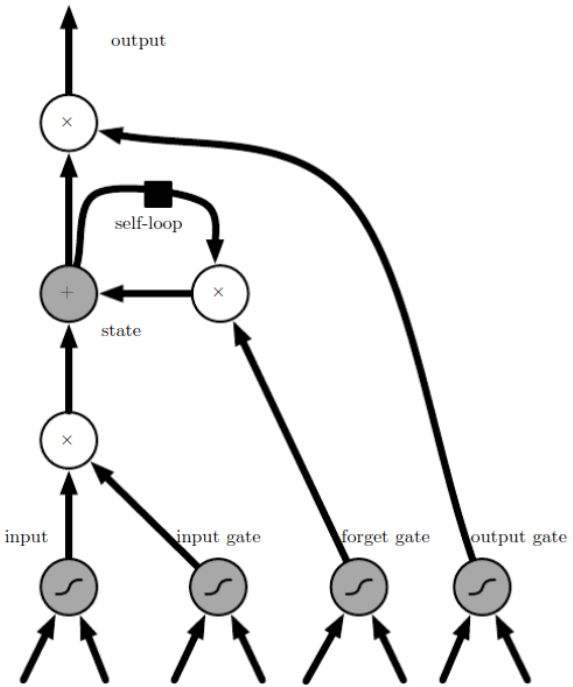
$$\mathbf{h_t} = \mathbf{x_t} \cdot W$$
$$\mathbf{y_t} = \tanh(\mathbf{h_t} + \mathbf{y_{t-1}} \cdot U)$$

Example: 10 cells, $\mathbf{x_t} \in \mathbb{R}^d \rightarrow W \in \mathbb{R}^{d \times 10}, U \in \mathbb{R}^{10 \times 10}, \mathbf{h_t}, \mathbf{y_t} \in \mathbb{R}^{10}$

Unrolled RNN



Long Short Term Memory (LSTM)



[Hochreiter and Schmidhuber, 1997], [Gers, 2000]
Image from `https://www.deeplearningbook.org/contents/rnn.html`

Gated Recurrent Units (GRU)

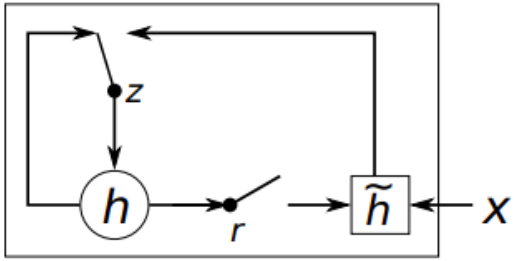


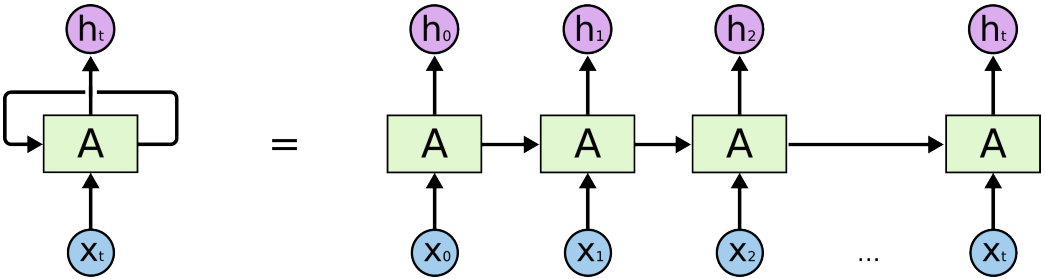
Figure 2: An illustration of the proposed hidden activation function. The update gate z selects whether the hidden state is to be updated with a new hidden state $\tilde{h}$. The reset gate r decides whether the previous hidden state is ignored. See

- Two gating units instead of three in LSTM
- shown to exhibit better performance on smaller datasets

[Cho et al, 2014], [Chung et al, 2014]

Generic RNNs

Unrolled RNN



- An RNN can be trained to predict the next symbol in a sequence
- In that case, the output at each timestep t is the conditional distribution $p(x_t|x_{t-1}, \ldots, x_1)$
- For language modeling for instance, we would use a softmax function:

$$p(x_{t,j}|x_{t-1}, \ldots, x_1) = \frac{\exp(\mathbf{w_j h_t})}{\sum_{i=1}^{K} \exp(\mathbf{w_i h_t})}$$

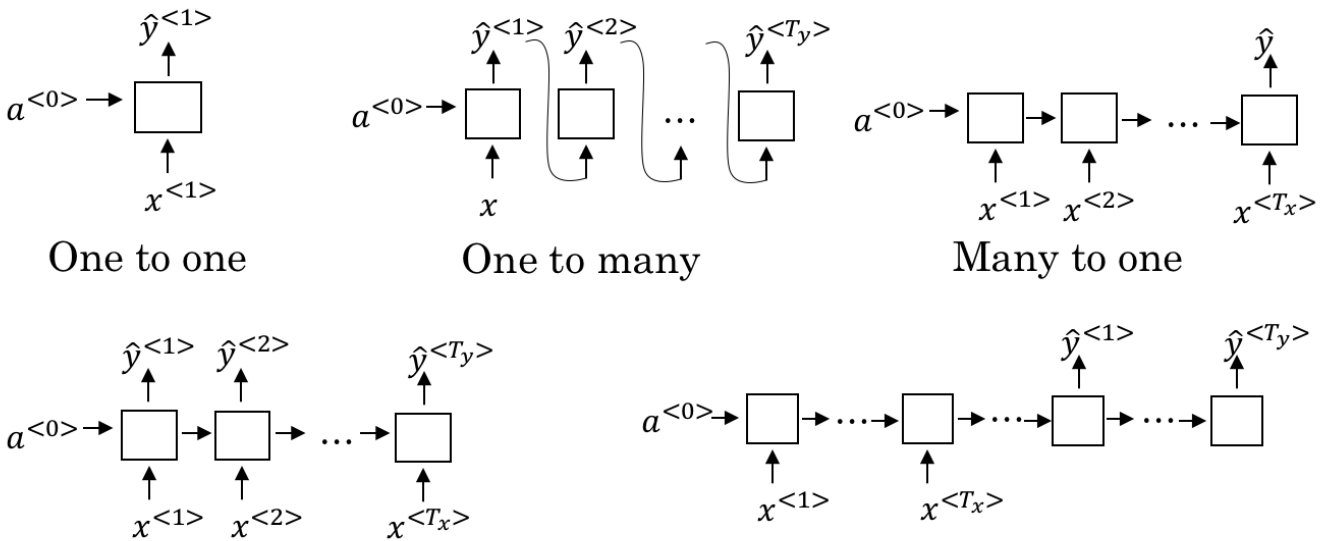
  for all possible words $j = 1 \ldots K$, where $\mathbf{w_j}$ are the rows of a weight matrix W
- By combining these probabilities, we can compute the probability of the sequence $\mathbf{x}$ using

$$p(\mathbf{x}) = \prod_{t=1}^{T} p(x_t|x_{t-1}, \ldots, x_1)$$

Forward pass and backpropagation

RNN trained with the backpropagation through time algorithm (BPTT)

Summary of RNN types



Andrew Ng

Outline

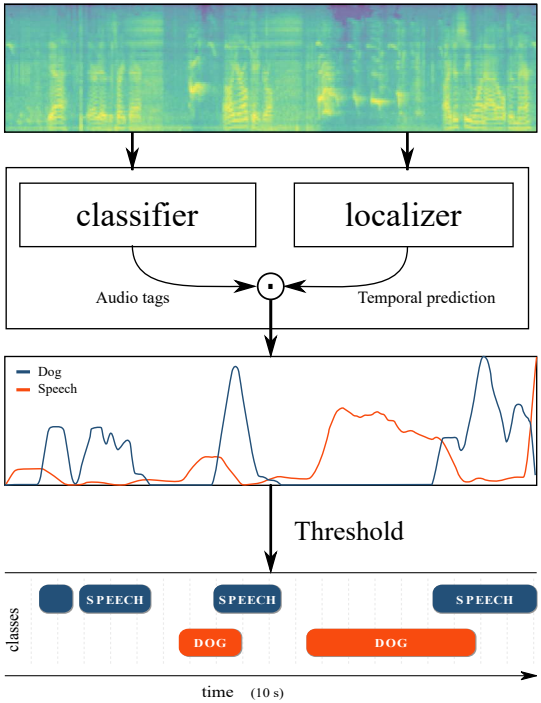
Bird's eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition

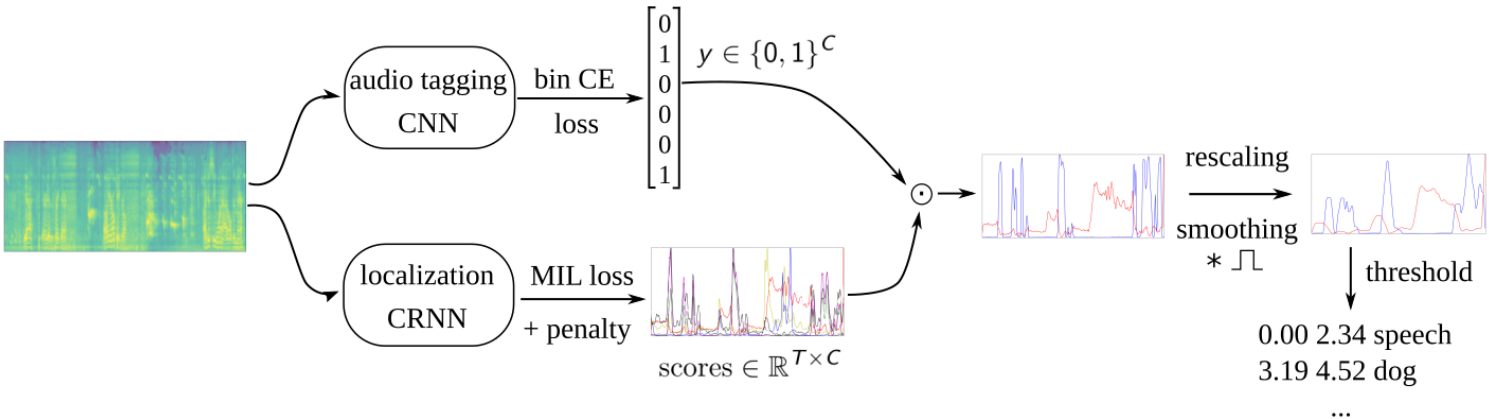
- Motivation for end-to-end systems
- From sounds to characters: CTC models
- Attention-based encoder-decoder models: Listen, Attend, Spell
- A few words on transfer learning

Case study: Sound Event Detection



- Multi-label classification problem
- *Weak labels*: audio tags, $\mathbf{y} \in \{0,1\}^C$: $\rightarrow$ many-to-one RNN
- *Strong labels*: frame-level tags, $Y \in \{0,1\}^{T \times C}$: $\rightarrow$ many-to-many RNN

Generic approach



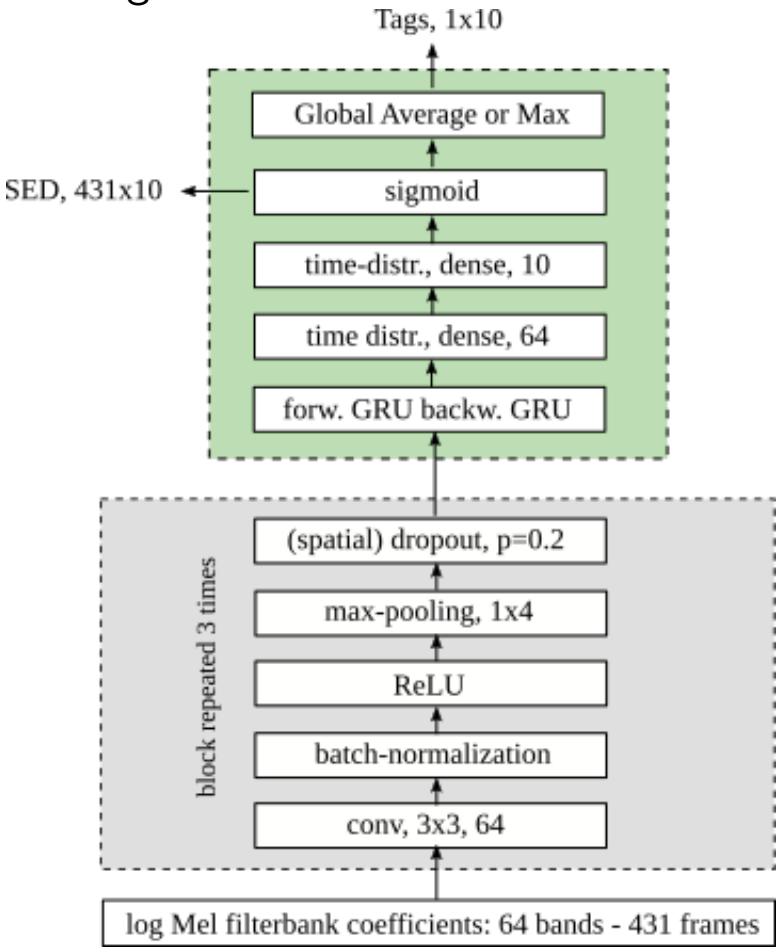
Weakly-labeled SED: problem statement

- We want to perform:
  - "audio tagging": infer *weak* labels, $\hat{\mathbf{y}}$
  - "localization" or SED: infer *strong* labels, $\hat{Y}$
- Input: a sequence of T.F. representation vectors, $X = \{\mathbf{x_1}, \ldots \mathbf{x_T}\}$
- **Only weak labels are available for learning**
- In a recording weakly labeled as *Dog*, some acoustic frames will comprise dog barking and others will not

Options:

1. False strong labeling and then refine the strong labels
2. Models with attention, e.g. with Gated Linear Units
3. Multiple Instance Learning setting (MIL)

Baseline model: Convolutional Recurrent Neural Network with "false strong labeling"



Loss function for classification: cross-entropy

"False strong labeling" (FSL): strong labels == weak label

$$\mathrm{loss}(\{X^k, y_c^k\}) = \mathrm{binCE}(y_c^k, \hat{y}_{tc}^k)$$

$$\mathrm{binCE}(y_c^k, \hat{y}_{tc}^k) = -y_c^k \log \hat{y}_{tc}^k - (1 - y_c^k) \log(1 - \hat{y}_{tc}^k)$$

- FSL achieves poor results for short duration events

Option 2: attention mechanisms

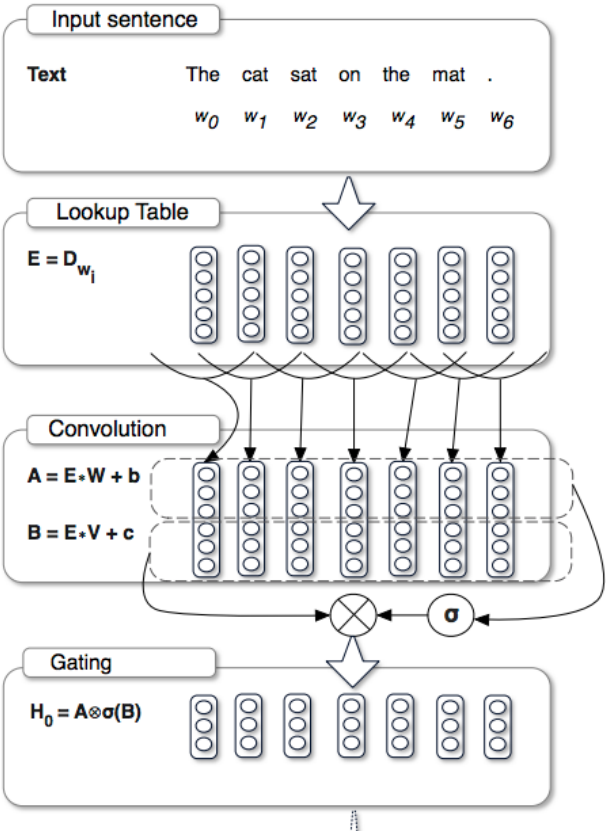
Language Modeling with Gated Convolutional Networks

Yann N. Dauphin [1] Angela Fan [1] Michael Auli [1] David Grangier [1]

- Pre-dominant approach to language modeling: recurrent neural networks
- Gated Linear Units (GLU): non-recurrent alternative with stacked convolutions, allows parallelization over sequential tokens

Gated Linear Units (GLU)

$$\mathbf{h_l}(X) = (X * W + \mathbf{b}) \odot \sigma(X * V + \mathbf{c})$$

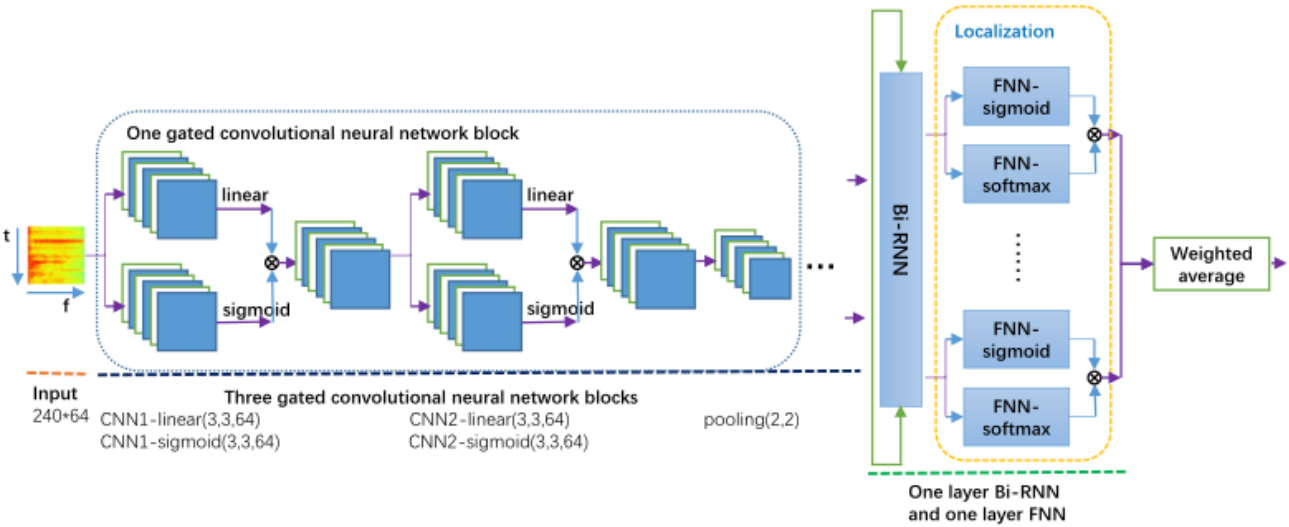


- $X \in \mathbb{R}^{N \times m}$ input of layer l
- learnable: $W, V \in \mathbb{R}^{k \times m \times n}$, $\mathbf{b}, \mathbf{c} \in \mathbb{R}^n$
- Similar to LSTM, the $\sigma(X * V + \mathbf{c})$ gates control the information pass
- The gates replace standard activation functions (such as ReLU)
- Linear activation to help gradients flow
- Input shift to prevent using information from future words
- GLUs wrapped in a residual block
- Output activation function: adaptive Softmax [Grave et al, 2016]

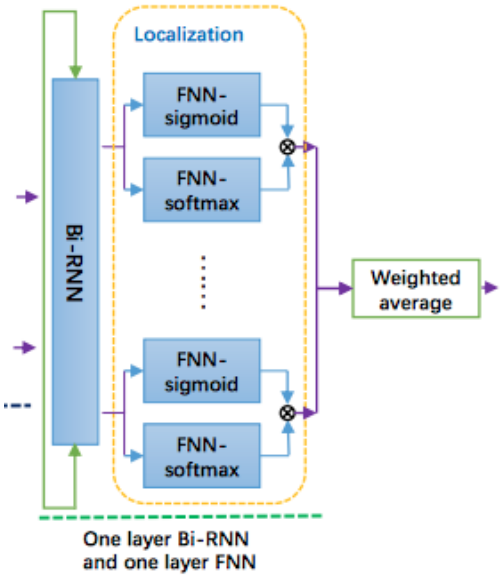
Gated CNN in SED

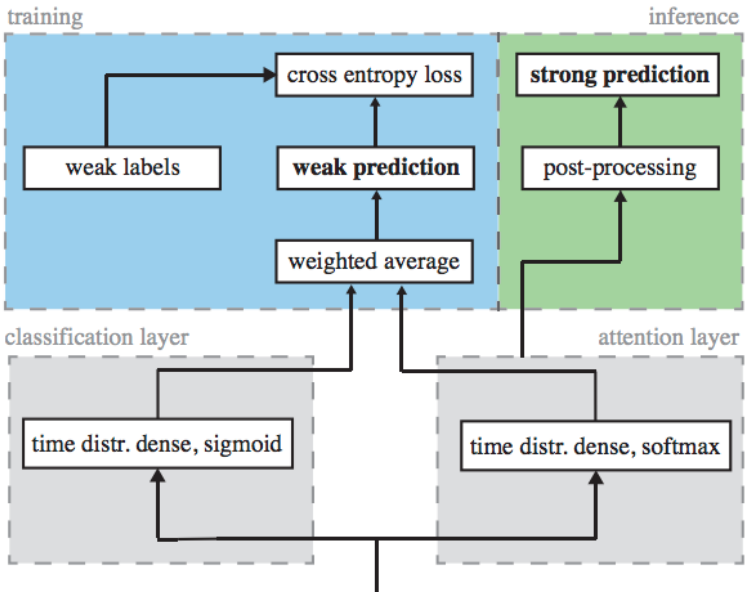
LARGE-SCALE WEAKLY SUPERVISED AUDIO CLASSIFICATION USING GATED CONVOLUTIONAL NEURAL NETWORK

Yong Xu, Qiuqiang Kong*, Wenwu Wang, Mark D. Plumbley*



Gated CNN in SED





Picture from [Robert Harb, Franz Pernkopf, DCASE-2018]

Final audio tag prediction:

$$y_c = \frac{\sum_t^T \mathbf{z}_c^{\mathrm{cla}}(t) \odot \mathbf{z}_c^{\mathrm{att}}(t)}{\sum_t^T \mathbf{z}_c^{\mathrm{att}}(t)}$$

Option 3: Multiple Instance Learning

- MIL terminology
  - $X = \{\mathbf{x_1}, \ldots, \mathbf{x_T}\}$: a bag or a set
  - $\mathbf{x_i}$: an instance
  - No dependency nor ordering among each other
  - A single binary label Y
  - Unknown individual labels: $y = \{y_1, \ldots, y_T\}$

MIL assumptions:

$$Y = \begin{cases} 0, & \text{iff } \sum_t y_t = 0, \\ 1, & \text{otherwise} \end{cases}$$

A MIL model must be **permutation invariant**
In a compact form:

$$Y = \max_t \{y_t\}$$

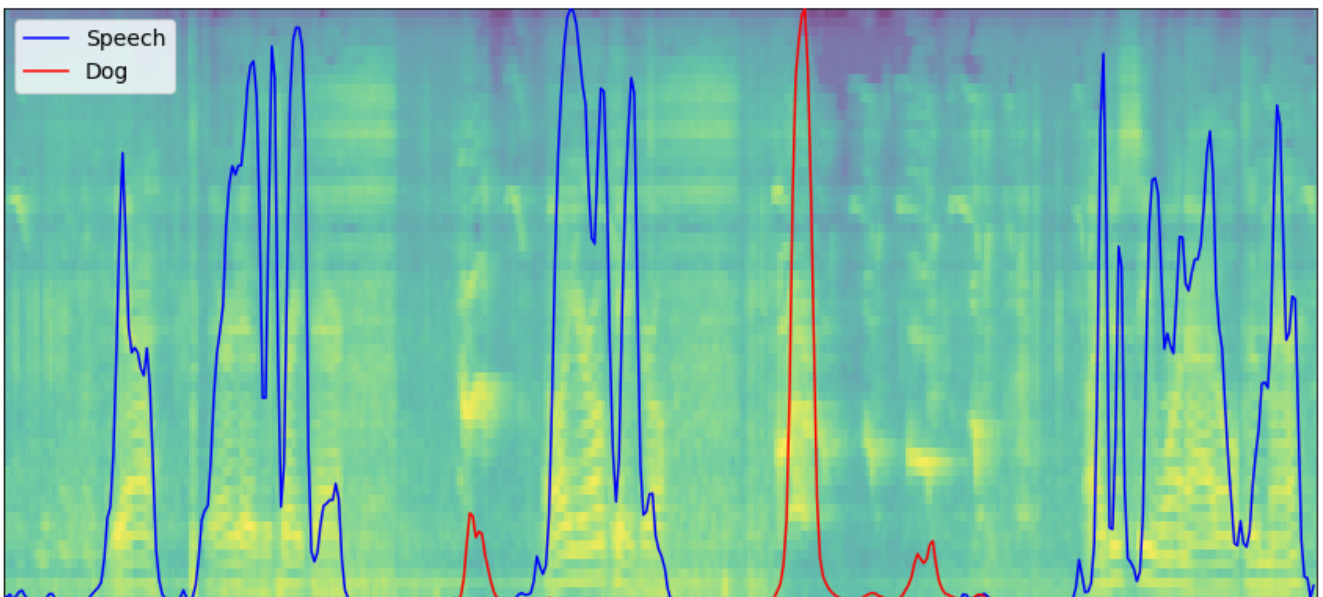
MIL loss for WL-SED

$$\mathrm{loss}(\{X, y_c\}) = \mathrm{binCE}(y_c, \max_t \hat{\mathbf{y}}_{\mathbf{tc}}) \tag{1}$$

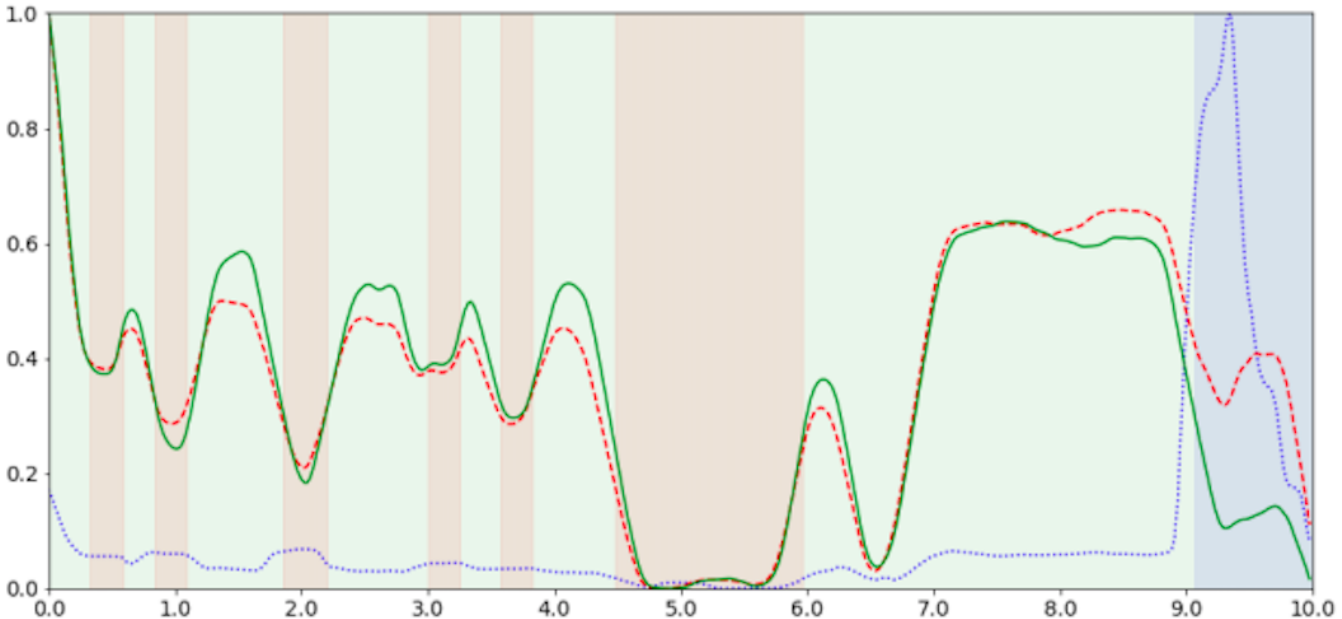
where

- $y_c \in \{0, 1\}$: ground truth label for class c
- $\hat{\mathbf{y}}_{\mathbf{tc}} \in [0, 1]^T$: temporal predictions for class c

Example



MIL loss for WL-SED

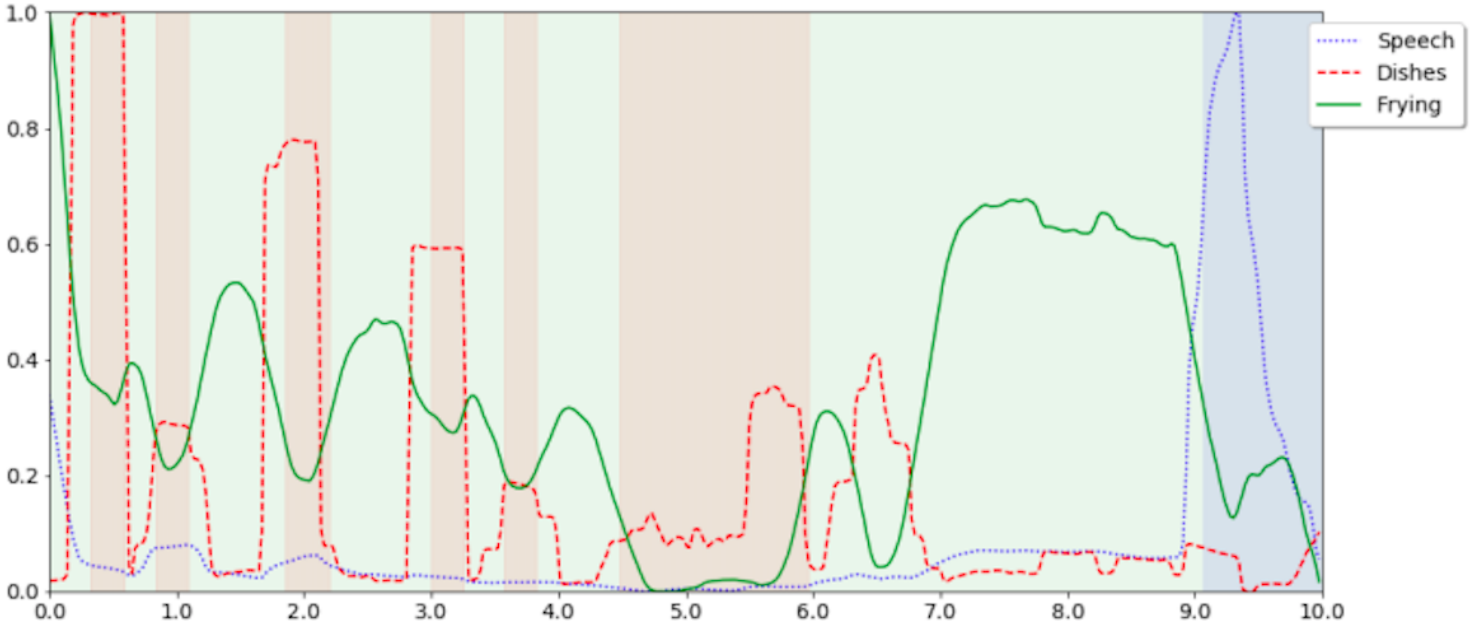


- Issue in distinguishing some classes

Adding a similarity penalty

$$\begin{aligned}\mathrm{loss}(\{X, y_c\}) &= \mathrm{binCE}(y_c, \max_t \hat{\mathbf{y}}_{\mathbf{tc}}) \\ &\quad + \alpha\, y_c \sum_{l \neq c}^{C} y_l \max(0, \cos(\hat{\mathbf{y}}_{:\mathbf{l}}, \hat{\mathbf{y}}_{:\mathbf{c}}))\end{aligned}$$

Adding a similarity penalty



Joint work with Léo Cances `arxiv.org/abs/1901.03146`
Audio demo?

Outline

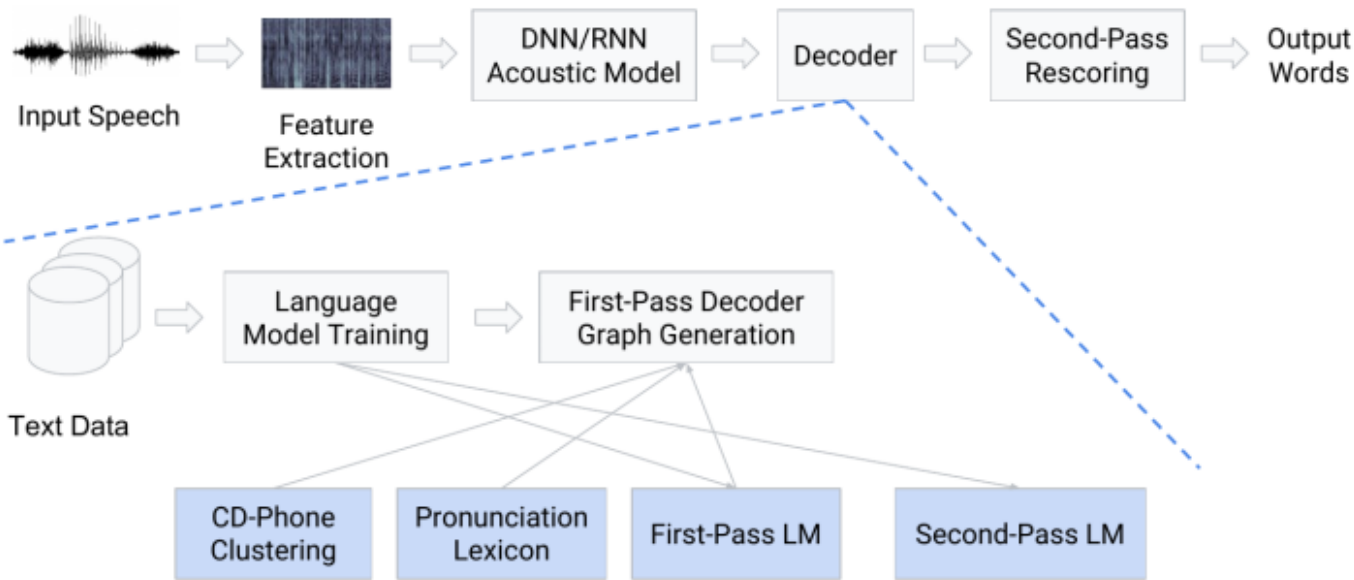
Bird's eye view on Recurrent Neural Networks

Case study: Weakly-labeled Sound Event Detection

Speech recognition

- Motivation for end-to-end systems
- From sounds to characters: CTC models
- Attention-based encoder-decoder models: Listen, Attend, Spell
- A few words on transfer learning

Conventional ASR pipeline



→ A single model instead of all these modules??

Slide from `http://iscslp2018.org/images/T4_Towards%20end-to-end%20speech%20recognition.pdf`

End-to-end speech recognition

"A system which directly maps a sequence of input acoustic features into a sequence of graphemes or words"

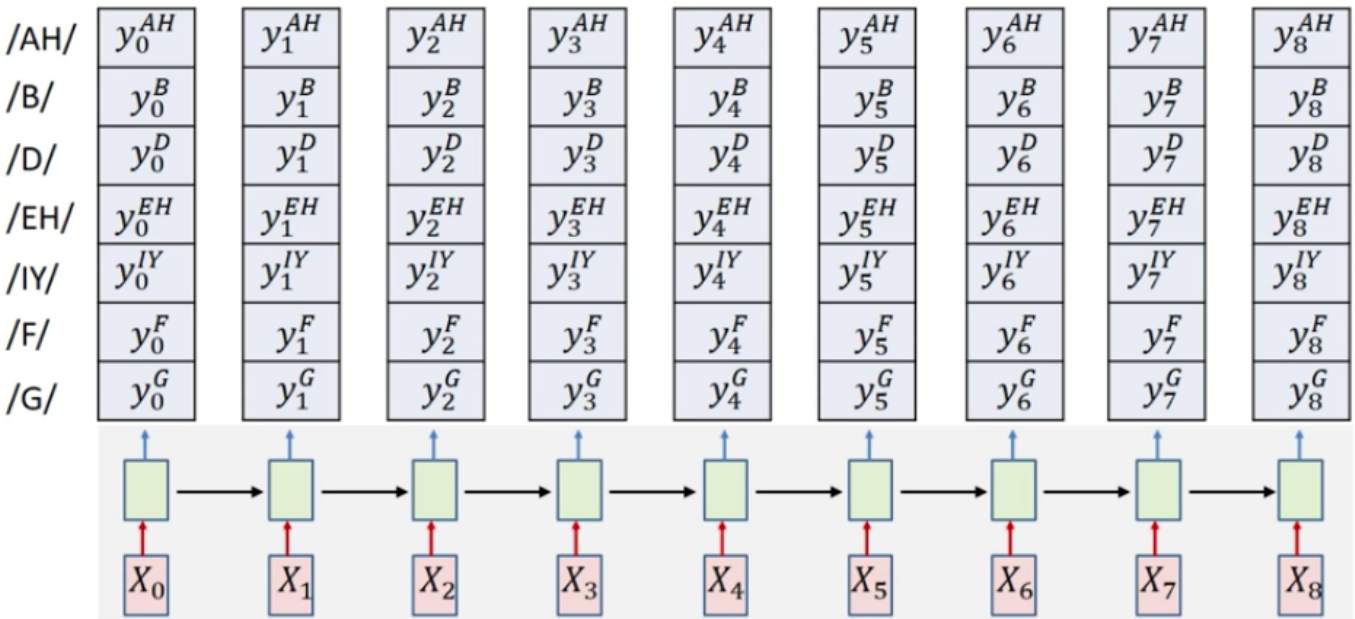
- Since about 2014, first attempt: CTC-based models
- Alternative: attention models

Connectionist Temporal Classification (CTC)

Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks

Alex Graves[1] ALEX@IDSIA.CH
Santiago Fernández[1] SANTIAGO@IDSIA.CH
Faustino Gomez[1] TINO@IDSIA.CH
Jürgen Schmidhuber[1,2] JUERGEN@IDSIA.CH

- CTC proposed by [Graves *et al*, 2006]
- A way to train acoustic models without frame-level alignments

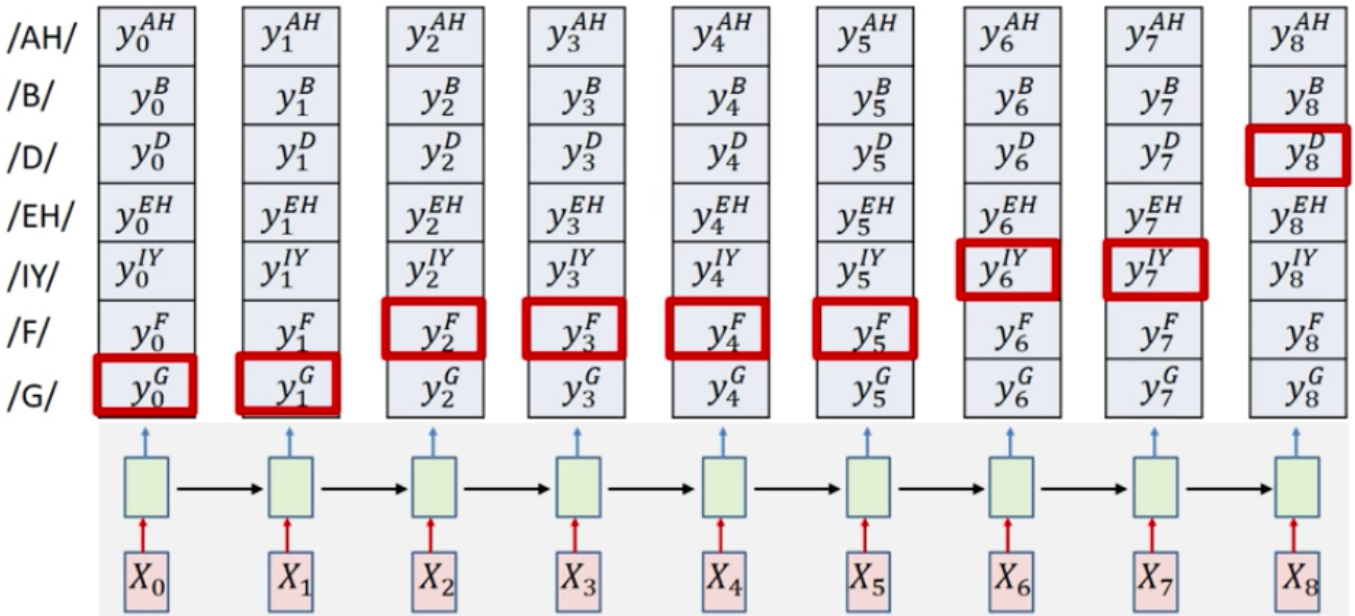


- Find most likely symbol sequence given inputs

$$S_0 \dots S_{K-1} = \underset{S'_0 \dots S'_{K-1}}{\operatorname{argmax}} \; prob(S'_0 \dots S'_{K-1} | X_0 \dots X_{N-1})$$

14

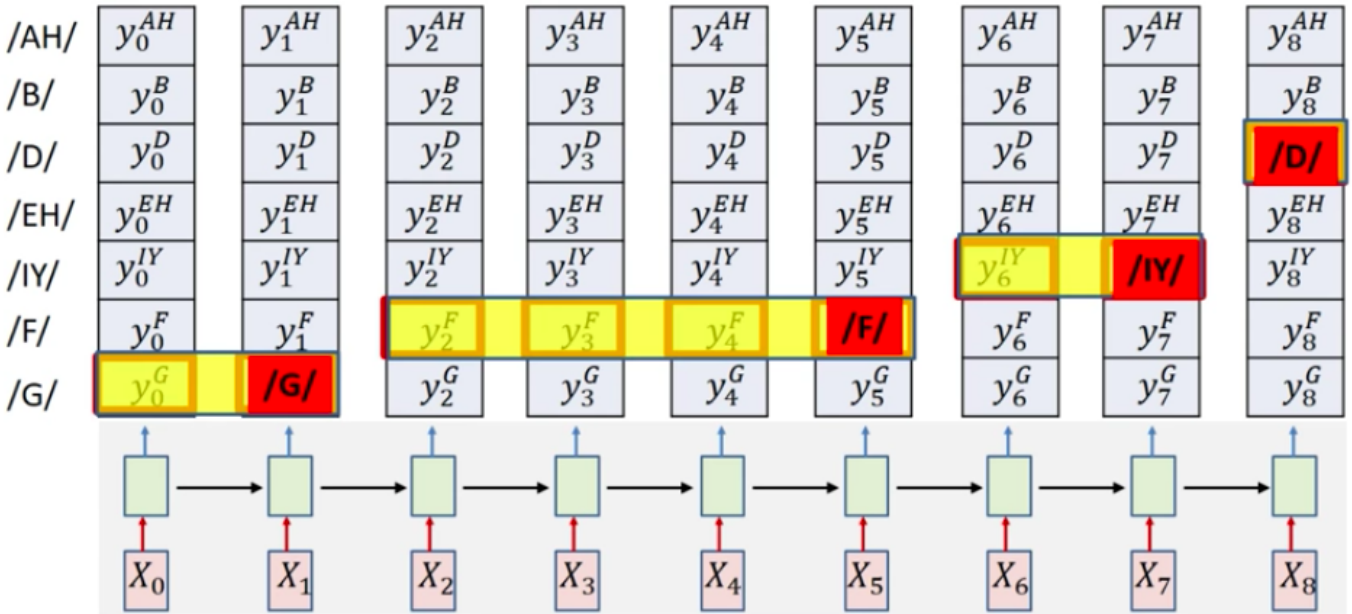
Slide from Bhiksha Raj



- Option 1: Simply select the most probable symbol at each time

15

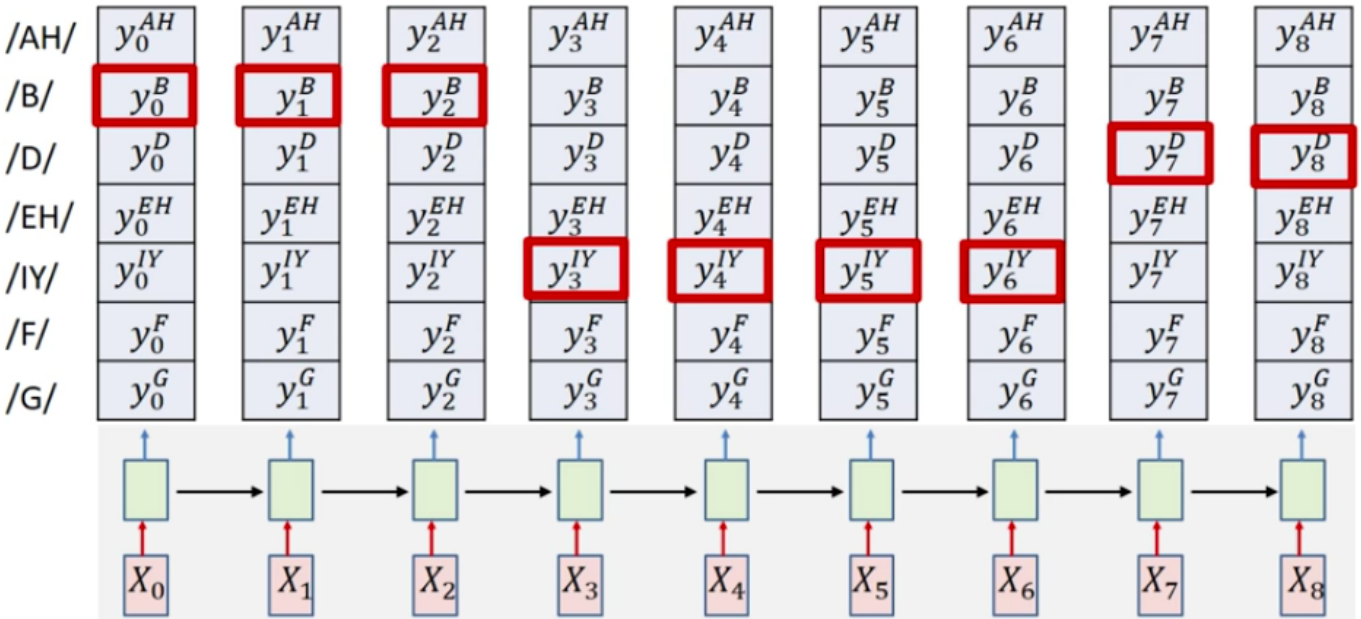
Slide from Bhiksha Raj



- Option 1: Simply select the most probable symbol at each time
  - *Merge* adjacent repeated symbols, and place the actual emission of the symbol in the final instant

16

Slide from Bhiksha Raj

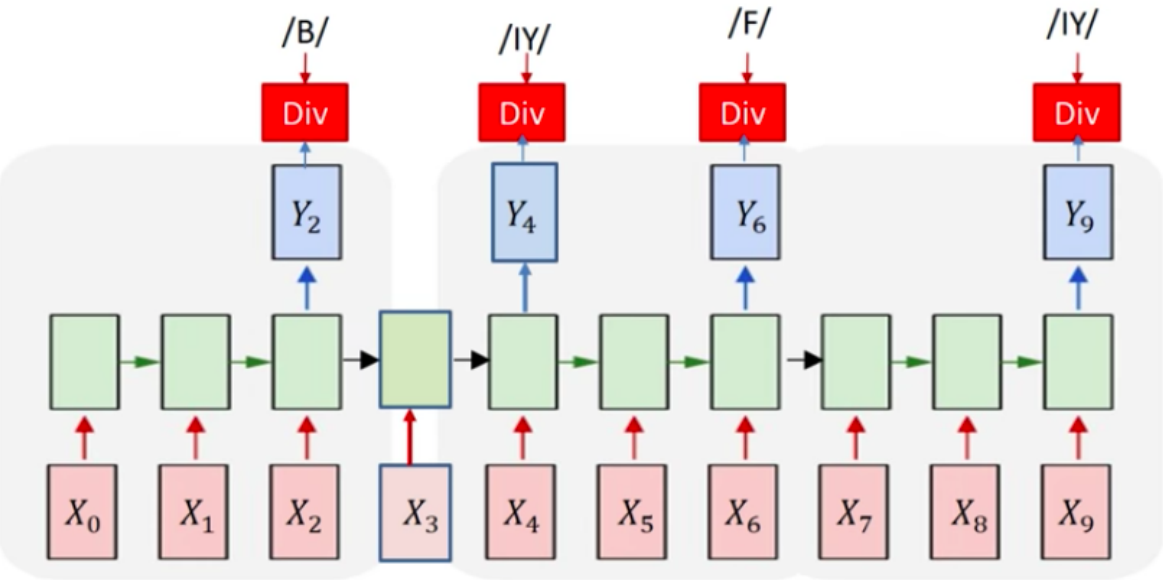


- Option 2: Impose external constraints on what sequences are allowed
  - *E.g.* only allow sequences corresponding to dictionary words
  - *E.g. Sub-symbol* units (like in HW1 – what were they?)

19

Slide from Bhiksha Raj

How do we train these models?

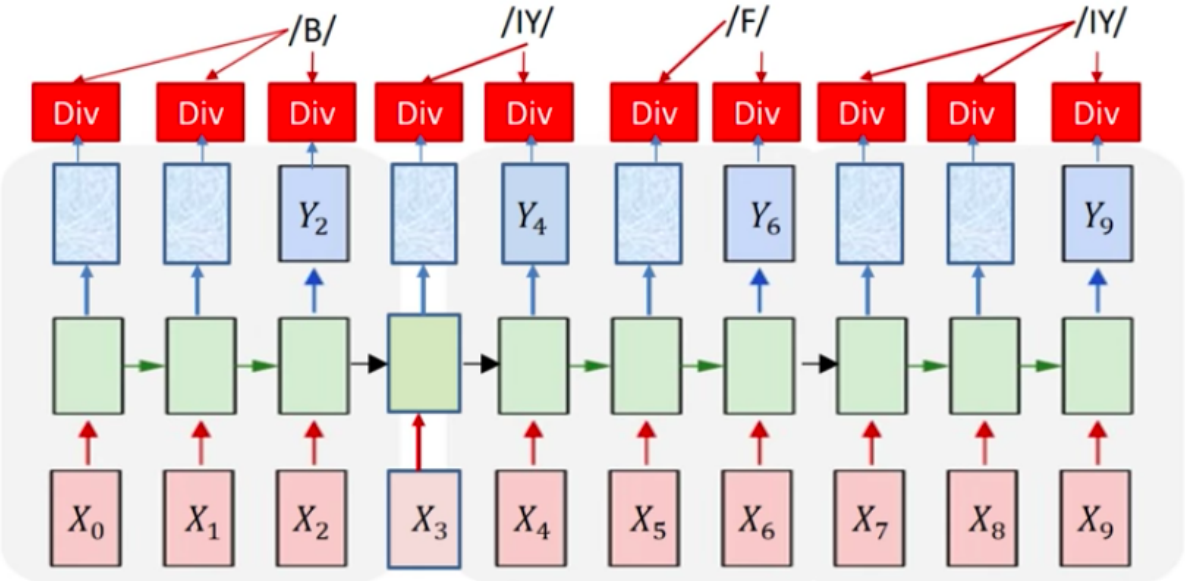


- Either just define Divergence as:

$$DIV = Xent(Y_2, B) + Xent(Y_4, IY) + Xent(Y_6, F) + Xent(Y_9, IY)$$

- Or..

Slide from Bhiksha Raj



- Either just define Divergence as:

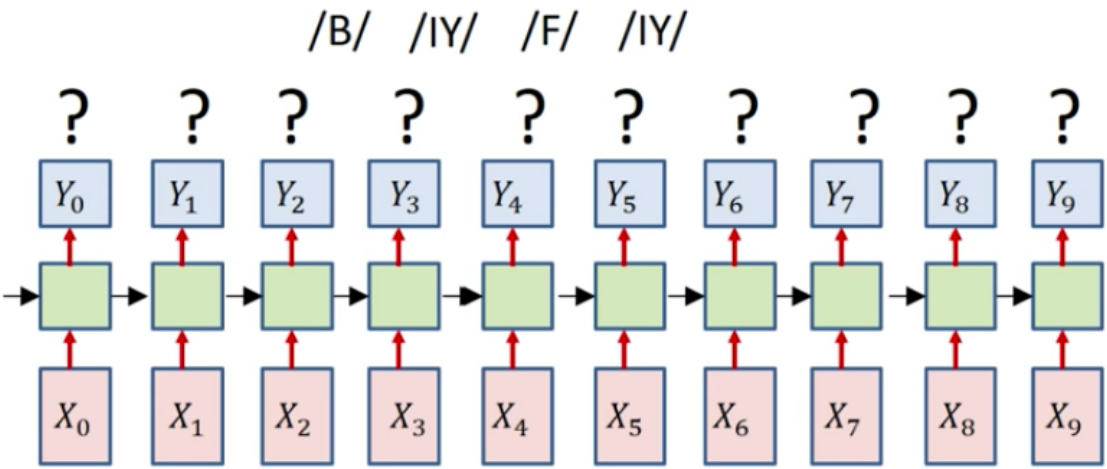
$$DIV = Xent(Y_2, B) + Xent(Y_4, IY) + Xent(Y_6, F) + Xent(Y_9, IY)$$

- Or *repeat the symbols over their duration*

$$DIV = \sum_t Xent(Y_t, symbol_t) = -\sum_t \log Y(t, symbol_t)$$

24

Problem: No timing information provided



- Only the sequence of output symbols is provided for the training data
  - But no indication of which one occurs where
- How do we compute the divergence?
  - And how do we compute its gradient w.r.t. Y_t

26

Slide from Bhiksha Rai

Constraining the alignment: Try 1

/B/	y_0^B		y_1^B		y_2^B		y_3^B		y_4^B		y_5^B		y_6^B		y_7^B		y_8^B
/IY/	y_0^{IY}		y_1^{IY}		y_2^{IY}		y_3^{IY}		y_4^{IY}		y_5^{IY}		y_6^{IY}		y_7^{IY}		y_8^{IY}
/F/	y_0^F		y_1^F		y_2^F		y_3^F		y_4^F		y_5^F		y_6^F		y_7^F		y_8^F

- Block out all rows that do not include symbols from the target sequence
  - E.g. Block out rows that are not /B/ /IY/ or /F/

Slide from Bhiksha Raj

Blocking out unnecessary outputs

Compute the entire output (for all symbols)

Copy the output values for the target symbols into the secondary reduced structure

33

Slide from Bhiksha Raj

Constraining the alignment: Try 1

/B/	y_0^B	y_1^B	y_2^B	y_3^B	y_4^B	y_5^B	y_6^B	y_7^B	y_8^B
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}
/F/	y_0^F	y_1^F	y_2^F	y_3^F	y_4^F	y_5^F	y_6^F	y_7^F	y_8^F

- Only decode on reduced grid
  - We are now assured that only the appropriate symbols will be hypothesized
- Problem: This still doesn't assure that the decode sequence correctly expands the target symbol sequence
  - E.g. the above decode is not an expansion of /B//IY//F//IY/
- Still needs additional constraints

Slide from Bhiksha Raj

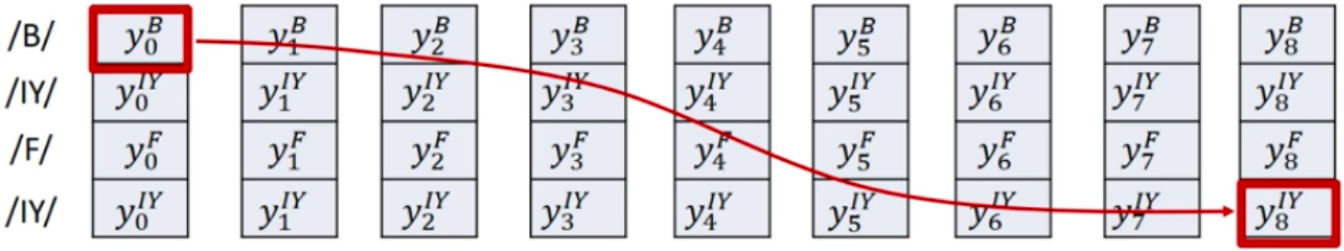
Try 2: Explicitly arrange the constructed table

/B/	y_0^B	y_1^B	y_2^B	y_3^B	y_4^B	y_5^B	y_6^B	y_7^B	y_8^B
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}
/F/	y_0^F	y_1^F	y_2^F	y_3^F	y_4^F	y_5^F	y_6^F	y_7^F	y_8^F
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}

/AH/	y_0^{AH}	y_1^{AH}	y_2^{AH}	y_3^{AH}	y_4^{AH}	y_5^{AH}	y_6^{AH}	y_7^{AH}	y_8^{AH}
/B/	y_0^B	y_1^B	y_2^B	y_3^B	y_4^B	y_5^B	y_6^B	y_7^B	y_8^B
/D/	y_0^D	y_1^D	y_2^D	y_3^D	y_4^D	y_5^D	y_6^D	y_7^D	y_8^D
/EH/	y_0^{EH}	y_1^{EH}	y_2^{EH}	y_3^{EH}	y_4^{EH}	y_5^{EH}	y_6^{EH}	y_7^{EH}	y_8^{EH}
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}
/F/	y_0^F	y_1^F	y_2^F	y_3^F	y_4^F	y_5^F	y_6^F	y_7^F	y_8^F
/G/	y_0^G	y_1^G	y_2^G	y_3^G	y_4^G	y_5^G	y_6^G	y_7^G	y_8^G

Arrange the constructed table so that from top to bottom it has the exact sequence of symbols required

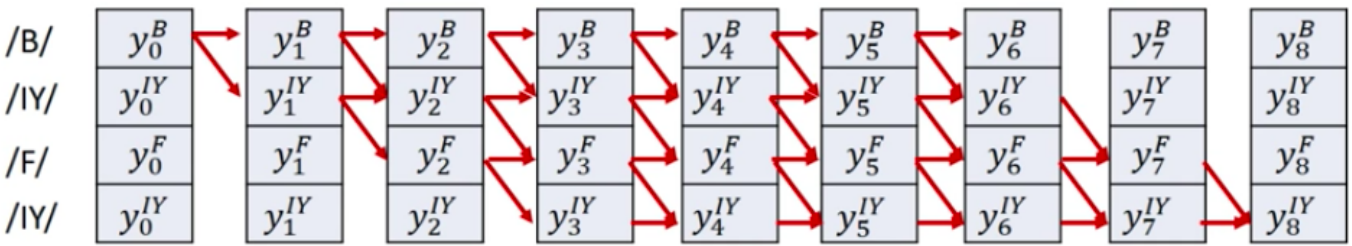
Explicitly constrain alignment



- Constrain that the first symbol in the decode *must* be the top left block
- The last symbol *must* be the bottom right
- The rest of the symbols must follow a sequence that *monotonically* travels down from top left to bottom right
  - I.e. never goes up
- This guarantees that the sequence *is* an expansion of the target sequence
  - /B/ /IY/ /F/ /IY/ in this case

Slide from Bhiksha Raj

Explicitly constrain alignment



- Compose a graph such that every path in the graph from source to sink represents a valid alignment
  - Which maps on to the target symbol sequence (/B//AH//T/)
- Edge scores are 1
- Node scores are the probabilities assigned to the symbols by the neural network
- The "score" of a path is the product of the probabilities of all nodes along the path

- Find the most probable path from source to sink using any dynamic programming algorithm
  - E.g. The Viterbi algorithm

Iterative update

- Option 1:
  - Determine alignments for every training instance
  - Train model (using SGD or your favorite approach) on the entire training set
  - Iterate
- Option 2:
  - During SGD, for each training instance, find the alignment during the forward pass
  - Use in backward pass

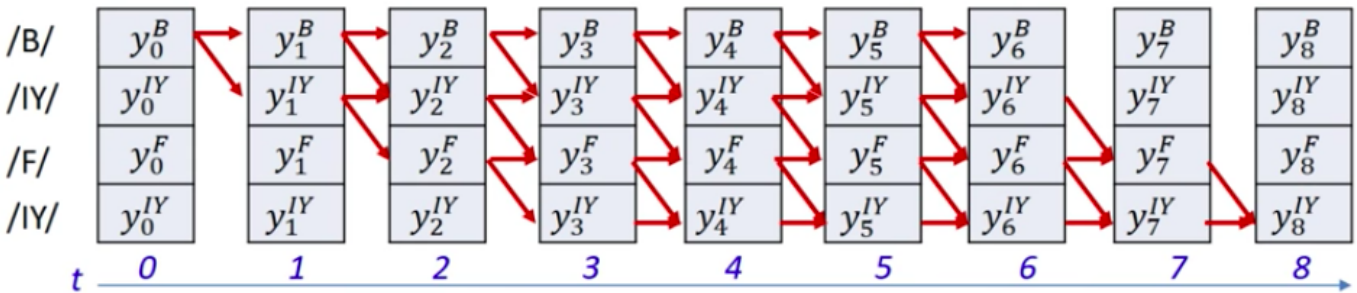
Slide from Bhiksha Raj

Iterative update: Problem

- Approach heavily dependent on initial alignment
- Prone to poor local optima
- Alternate solution: Do not commit to an alignment during any pass..

Slide from Bhiksha Raj

Averaging over *all* alignments



- Instead of only selecting the most likely alignment, use the statistical expectation over *all* possible alignments

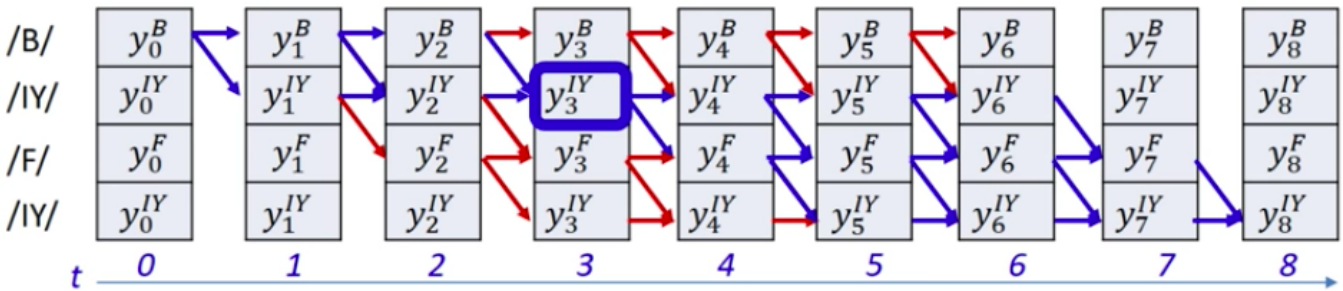
$$DIV = E\left[-\sum_t \log Y(t, s_t)\right]$$

  - Use the *entire distribution of alignments*
  - This will mitigate the issue of suboptimal selection of alignment

61

Slide from Bhiksha Raj

A posteriori probabilities of symbols



$$P(s_t = S_r|\mathbf{S},\mathbf{X}) \propto P(s_t = S_r,\mathbf{S}|\mathbf{X})$$

- $P(s_t = S_r,\mathbf{S}|\mathbf{X})$ is the total probability of all valid paths *in the graph for target sequence* **S** that go through the symbol S_r (the rth symbol in the sequence $S_1 \ldots S_K$) at time t

- We will compute this using the "forward-backward" algorithm

64

Slide from Bhiksha Raj

A key *decoding* problem

- Consider a problem where the output symbols are characters
- We have a decode: R R R O O O O O D
- Is this the symbol sequence ROD or ROOD?

Slide from Bhiksha Raj

A key *decoding* problem

- Consider a problem where the output symbols are characters
- We have a decode: R R R O O O O O D
- Is this the symbol sequence ROD or ROOD?

- Note: This problem does not always occur, e.g. when symbols have sub symbols
  - E.g. If O is produced as O1 and O2
    - A single O would be of the form O1 O1 .. O2 → O
    - Multiple Os would have the decode O1 .. O2.. O1..O2.. → OO

Slide from Bhiksha Raj

A key *decoding* problem

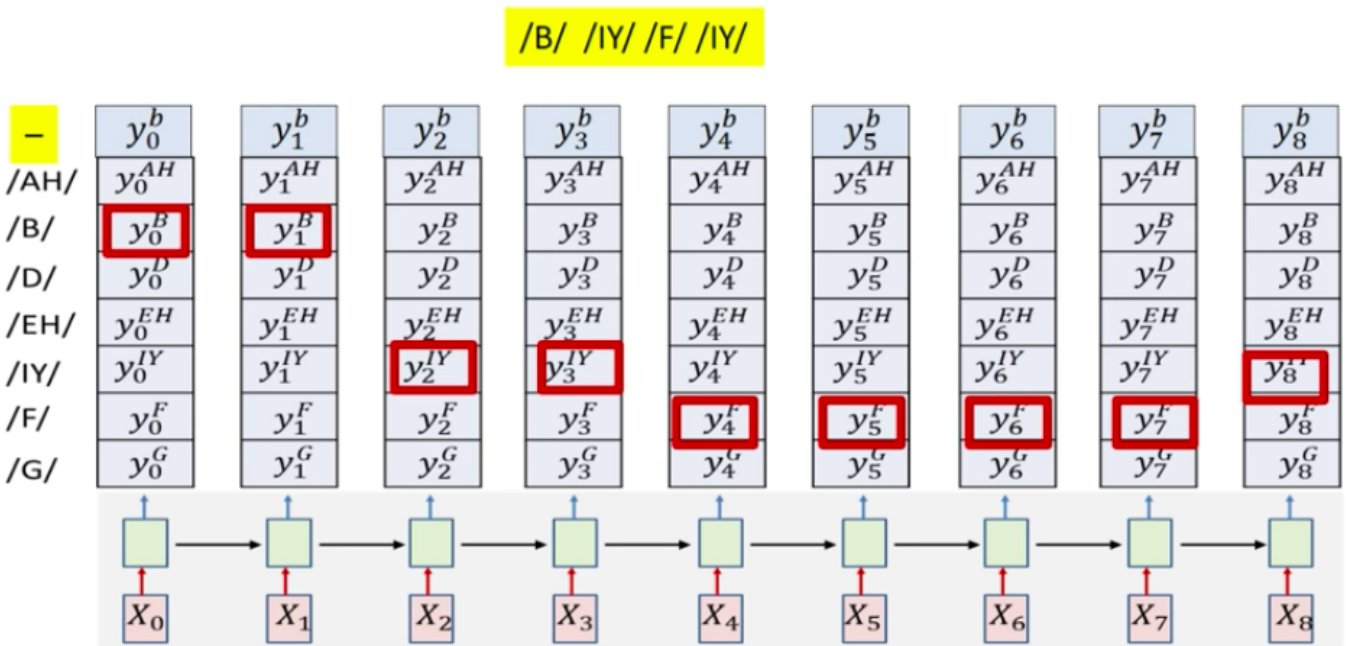
- We have a decode: R R R O O O O O D
- Is this the symbol sequence ROD or ROOD?

- Solution: Introduce an explicit extra symbol which serves to separate discrete versions of a symbol
  - A "blank" (represented by "-")
  - RRR---OO---DDD = ROD
  - RR-R---OO---D-DD = RRODD
  - R-R-R---O-O--DD-DDDD-D = RRROODDD
    - The next symbol at the end of a sequence of blanks is always a new character
    - When a symbol repeats, there must be at least one blank between the repetitions

- The symbol set recognized by the network must now include the extra blank symbol
  - Which too must be trained

The modified forward output

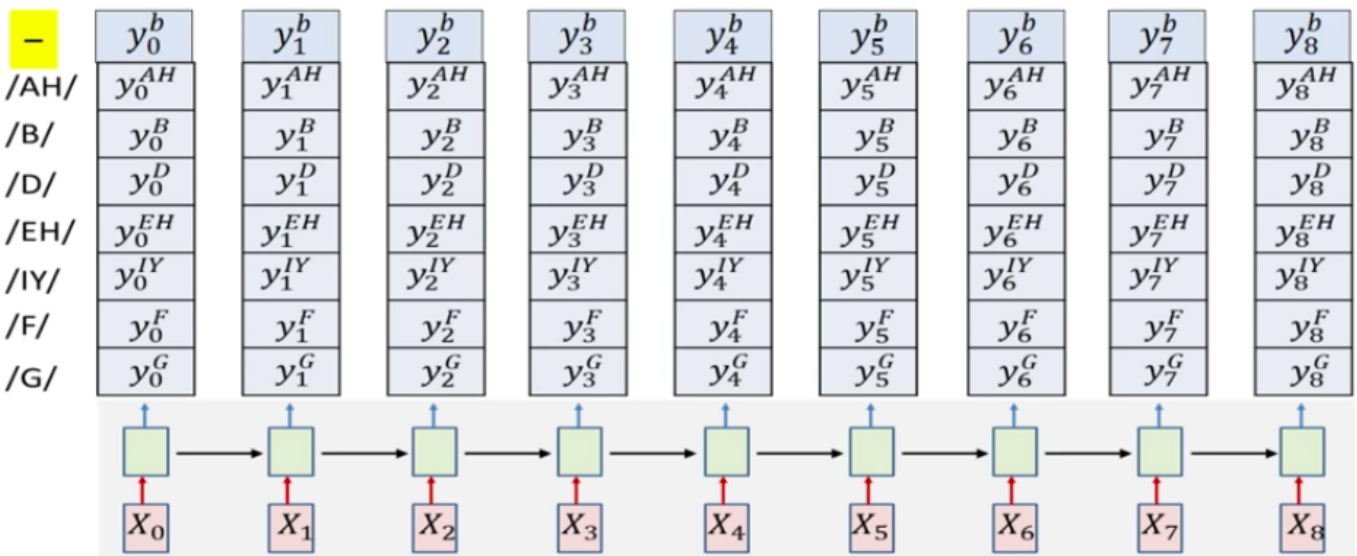
- Note the extra "blank" at the output



Slide from Bhiksha Raj

The modified forward output

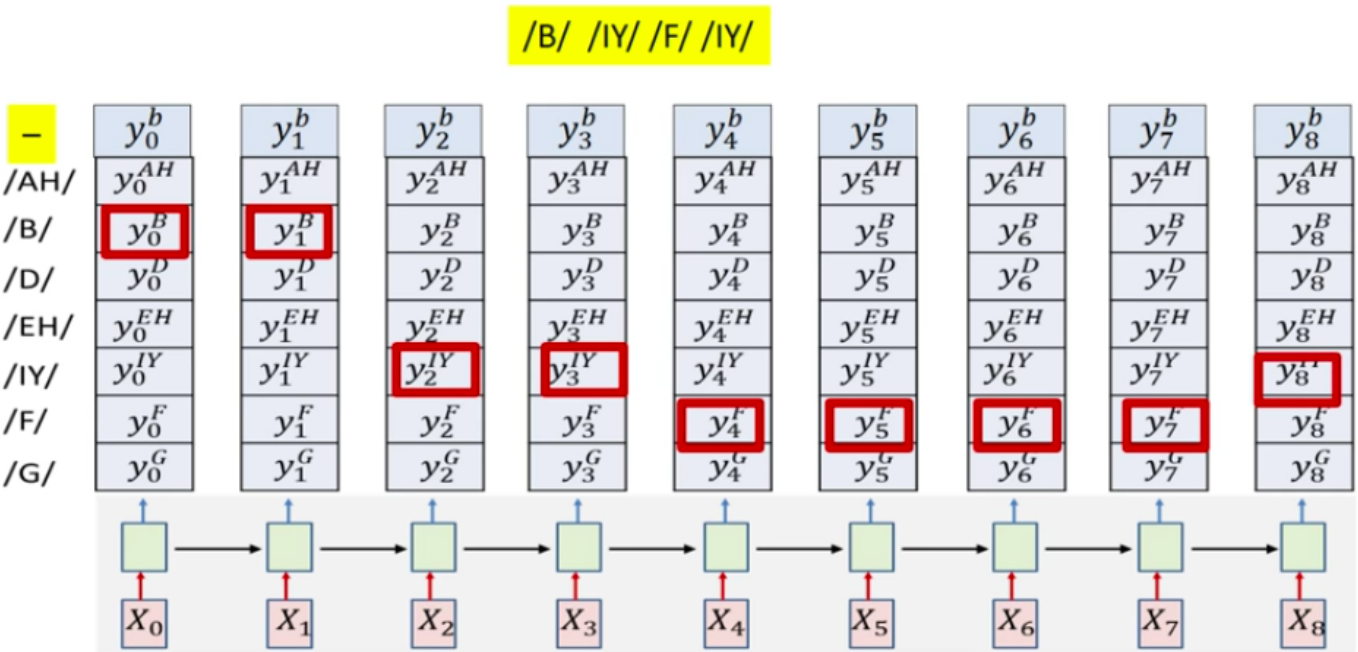
- Note the extra "blank" at the output



Slide from Bhiksha Raj

The modified forward output

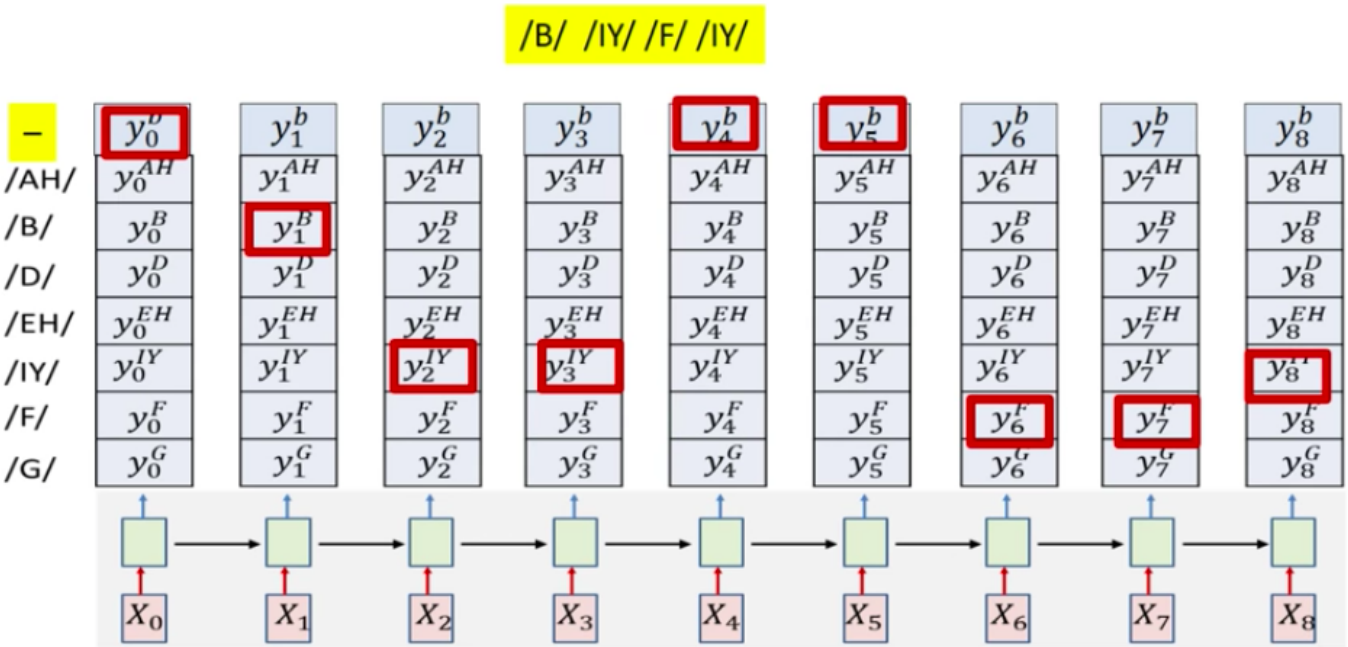
- Note the extra "blank" at the output



Slide from Bhiksha Raj

The modified forward output

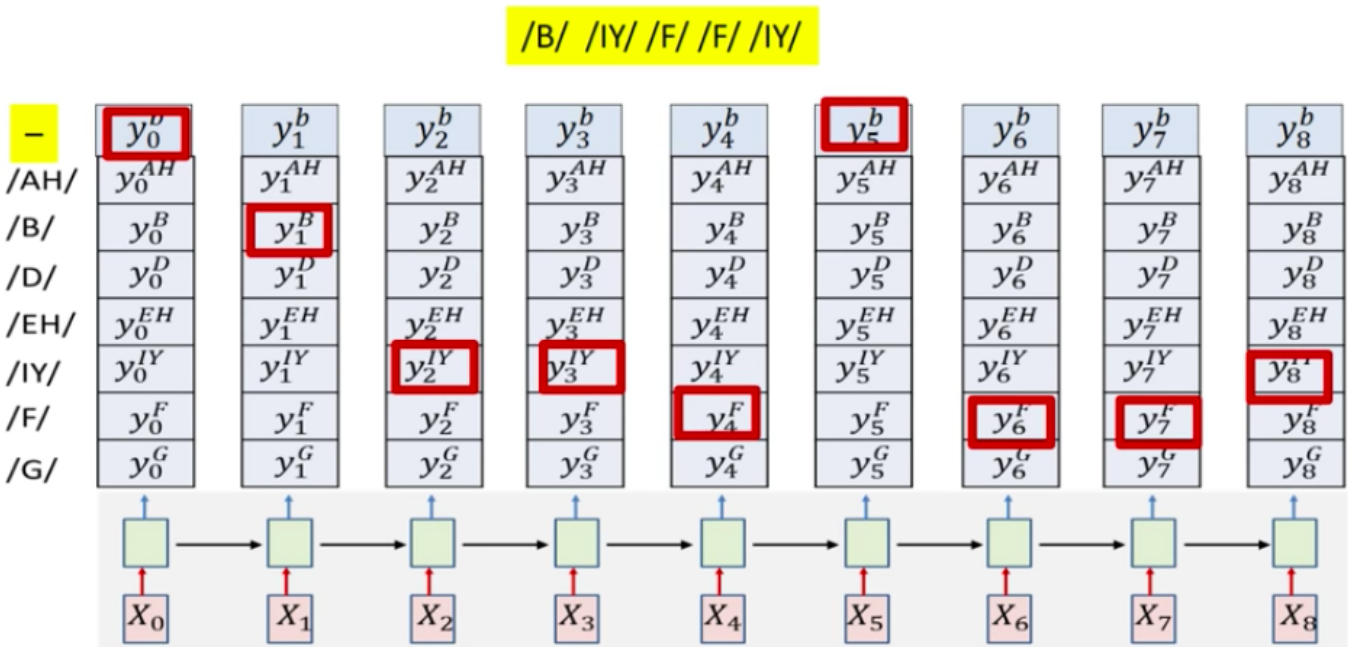
- Note the extra "blank" at the output



Slide from Bhiksha Raj

The modified forward output

- Note the extra "blank" at the output



Slide from Bhiksha Raj

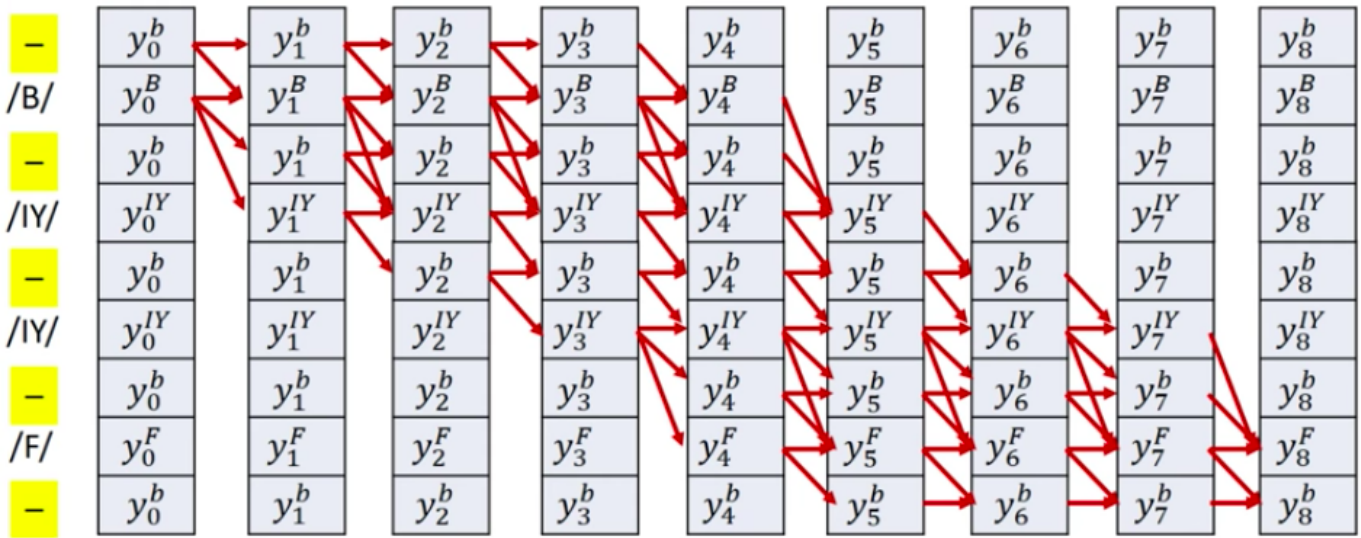
Composing the graph for training

–	y_0^b	y_1^b	y_2^b	y_3^b	y_4^b	y_5^b	y_6^b	y_7^b	y_8^b
/B/	y_0^B	y_1^B	y_2^B	y_3^B	y_4^B	y_5^B	y_6^B	y_7^B	y_8^B
–	y_0^b	y_1^b	y_2^b	y_3^b	y_4^b	y_5^b	y_6^b	y_7^b	y_8^b
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}
–	y_0^b	y_1^b	y_2^b	y_3^b	y_4^b	y_5^b	y_6^b	y_7^b	y_8^b
/IY/	y_0^{IY}	y_1^{IY}	y_2^{IY}	y_3^{IY}	y_4^{IY}	y_5^{IY}	y_6^{IY}	y_7^{IY}	y_8^{IY}
–	y_0^b	y_1^b	y_2^b	y_3^b	y_4^b	y_5^b	y_6^b	y_7^b	y_8^b
/F/	y_0^F	y_1^F	y_2^F	y_3^F	y_4^F	y_5^F	y_6^F	y_7^F	y_8^F
–	y_0^b	y_1^b	y_2^b	y_3^b	y_4^b	y_5^b	y_6^b	y_7^b	y_8^b

- With blanks
- Note: a row of blanks between any two symbols
- Also blanks at the very beginning and the very end

Slide from Bhiksha Raj

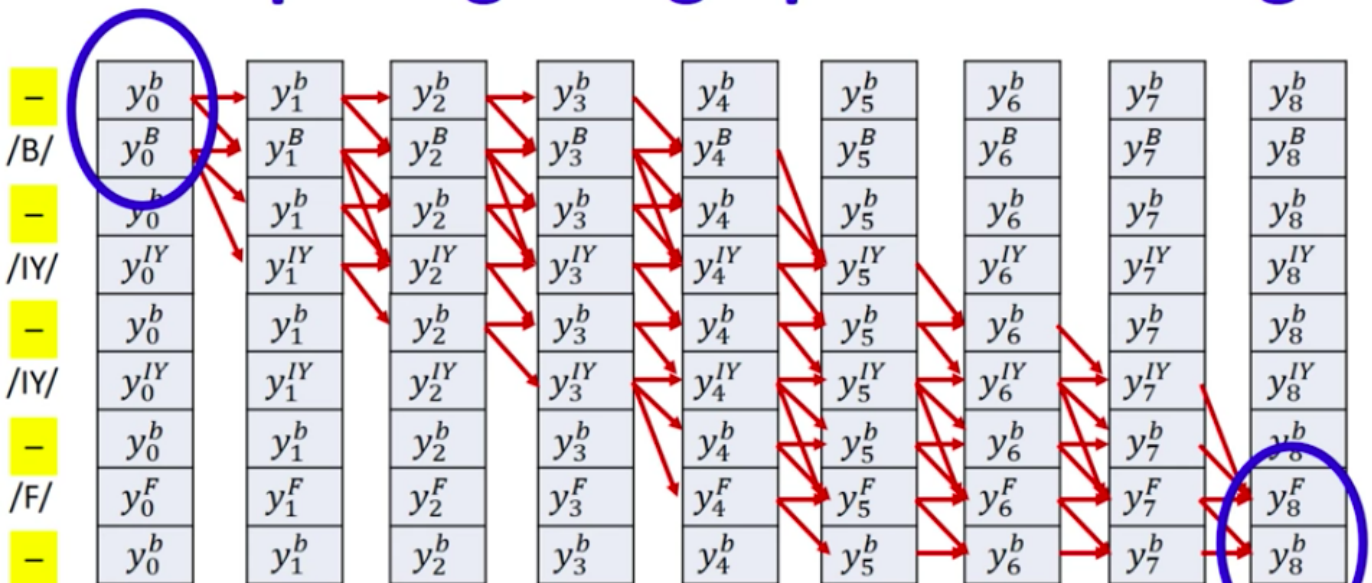
Composing the graph for training



- Add edges such that all paths from initial node(s) to final node(s) unambiguously represent the target symbol sequence

Slide from Bhiksha Raj

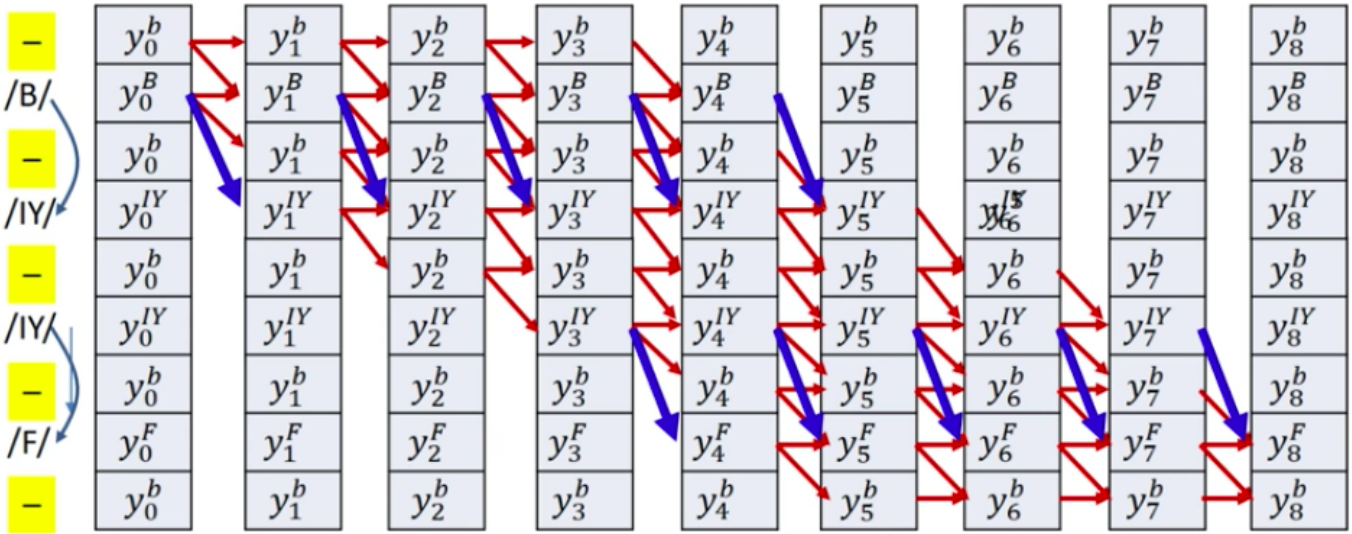
Composing the graph for training



- The first and last column are allowed to also end at initial and final blanks

Slide from Bhiksha Raj

Composing the graph for training



- The first and last column are allowed to also end at initial and final blanks
- Skips are permitted across a blank, but only if the symbols on either side are different
  - Because a blank is *mandatory between repetitions of a symbol* but *not required between distinct symbols*

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

The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.

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

The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.

- Labels at each time index are conditionally independent (like HMMs)

Final learning objective: maximizing probabilities of true labels:

$$\mathrm{CTC(X)} = -\log \mathrm{P(Y^*|X)}$$

How to decode with a CTC model?

- Greedy search ("max decoding"): sub-optimal
- Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)

Table 1. **Label Error Rate (LER) on TIMIT**. CTC and hybrid results are means over 5 runs, ± standard error. All differences were significant ($p < 0.01$), except between weighted error BLSTM/HMM and CTC (best path).

System	LER
Context-independent HMM	38.85 %
Context-dependent HMM	35.21 %
BLSTM/HMM	33.84 ± 0.06 %
Weighted error BLSTM/HMM	31.57 ± 0.06 %
CTC (best path)	31.47 ± 0.21 %
CTC (prefix search)	30.51 ± 0.19 %

- Greedy search ("max decoding"): sub-optimal
- Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)
- + N-best rescoring with an LM [Graves & Jaitly, 2014]

Table 1. **Wall Street Journal Results.** All scores are word error rate/character error rate (where known) on the evaluation set. 'LM' is the Language model used for decoding. '14 Hr' and '81 Hr' refer to the amount of data used for training.

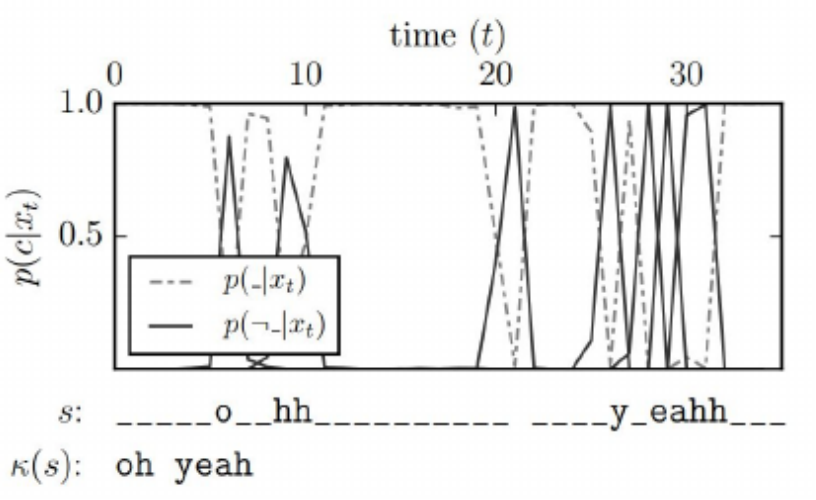
SYSTEM	LM	14 HR	81 HR
RNN-CTC	NONE	74.2/30.9	30.1/9.2
RNN-CTC	DICTIONARY	69.2/30.0	24.0/8.0
RNN-CTC	MONOGRAM	25.8	15.8
RNN-CTC	BIGRAM	15.5	10.4
RNN-CTC	TRIGRAM	13.5	8.7
RNN-WER	NONE	74.5/31.3	27.3/8.4
RNN-WER	DICTIONARY	69.7/31.0	21.9/7.3
RNN-WER	MONOGRAM	26.0	15.2
RNN-WER	BIGRAM	15.3	9.8
RNN-WER	TRIGRAM	13.5	8.2
BASELINE	NONE	—	—
BASELINE	DICTIONARY	56.1	51.1
BASELINE	MONOGRAM	23.4	19.9
BASELINE	BIGRAM	11.6	9.4
BASELINE	TRIGRAM	9.4	7.8
COMBINATION	TRIGRAM	—	6.7

- Greedy search ("max decoding"): sub-optimal
- Prefix beam search (PBS) proposed in [Graves, 2006] uses the forward-backward algo (also used to compute the gradients during training)
- + N-best rescoring with an LM [Graves & Jaitly, 2014]
- PBS with char/word LM incorporated [Hannun et al, 2014], [Maas et al, 2015]
- PBS + WFST in Eesen open-source toolkit (Kaldi based) [Miao, 2015]

Table 3. *Performance of the character-based Eesen system using different vocabularies and language models, and its comparison with results presented in previous work.*

System	Vocabulary	Language Model	WER%
Eesen	Original	Standard	9.07
Eesen	Expanded	Re-trained	7.34
Graves *et al.* [6]	Expanded	Re-trained	8.7
Hannun *et al.* [8]	Original	Unknown	14.1

CTC produces sparse and spiky activations

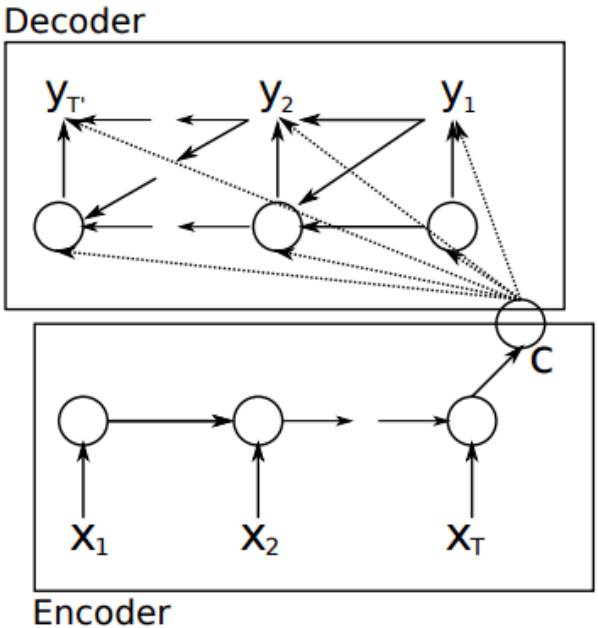


$\rightarrow$ not a good tool for speech analysts?

Conclusions on CTC

- Alignment-agnostic end-to-end ASR approach
- CTC assumption: for efficiency, network outputs at different frames are conditionally independent
- CTC models need an external LM, greedy decoding does not perform well
- Requires several tens of hours of data to work well
- Open question: is CTC adapted to small scale speech tasks and to speech analysis studies?

RNN encoder-decoder for Machine Translation (Cho, 2014)



- Recall the generic RNN update eq: $\mathbf{h}_t = f(\mathbf{h}_{t-1}, x_t)$
- With the RNN encoder-decoder, it becomes: $\mathbf{h}_t = f(\mathbf{h}_{t-1}, y_{t-1}, \mathbf{c})$
- $\mathbf{c}$: a summary of the input sequence

Attention models

- Issue with Cho's 2014 model: long sentences
- Where the model should focus its attention given a sequence of inputs?
- Badhanau's proposal: an encoder-decoder that learns to align and translate jointly

[Badhanau et al, 2014]

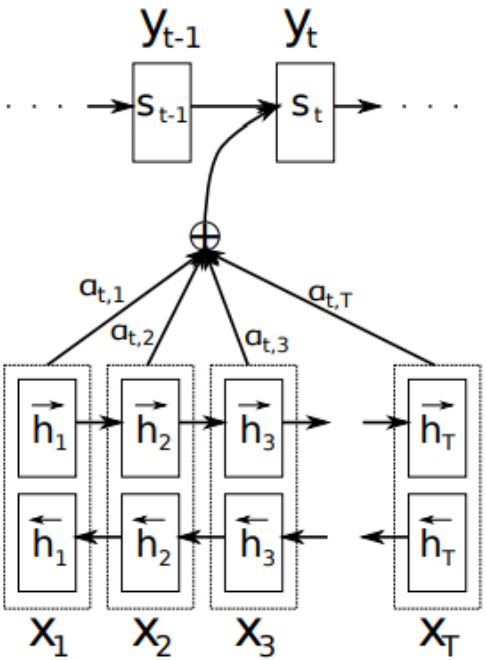


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention models

- $\mathbf{s}_t = f(\mathbf{s}_{t-1}, y_{t-1}, \mathbf{c_t})$
- The context vector $\mathbf{c_t}$ is computed at each time step using:

$$\mathbf{c_t} = \sum_{j=1}^{T_x} \alpha_{tj} \mathbf{h_j}$$

- α_{tj}: amount of attention y_t should pay to $\mathbf{h_j}$
- How to compute α_{tj}?

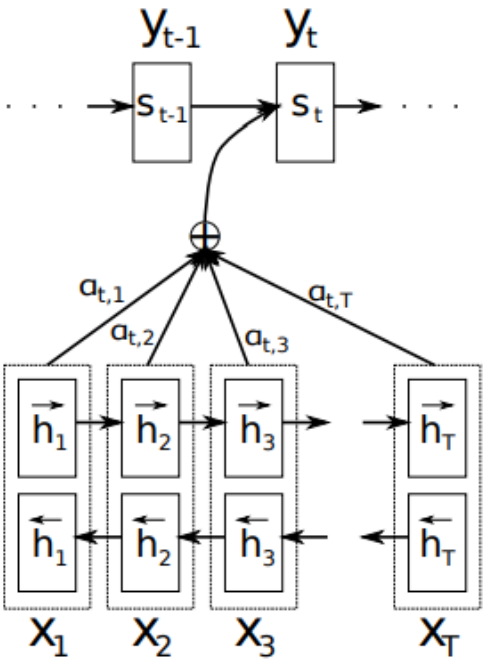


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

- α_{tj}: amount of attention y_t should pay to $\mathbf{h_j}$
- $\mathbf{c_t} = \sum_{j=1}^{T_x} \alpha_{tj} \mathbf{h_j}$
- How to compute α_{tj}?
- $\alpha_{tj} = \frac{\exp(e_{tj})}{\sum_{k=1}^{T_x} \exp(e_{tk})}$
- $e_{tj} = a(\mathbf{s_{t-1}}, \mathbf{h_j})$
- a: alignment model scoring how well the inputs around position j and the output at position i match
- In practice, what is a?

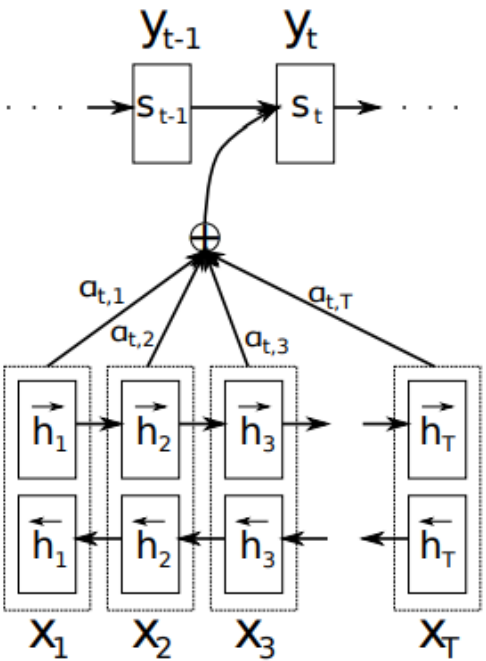
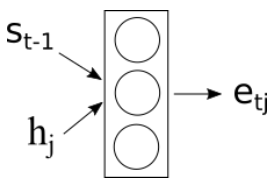


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

- $e_{tj} = a(\mathbf{s_{t-1}}, \mathbf{h_j})$
- a: alignment model scoring how well the inputs around position j and the output at position i match
- In practice, what is a?
- a is a small neural network



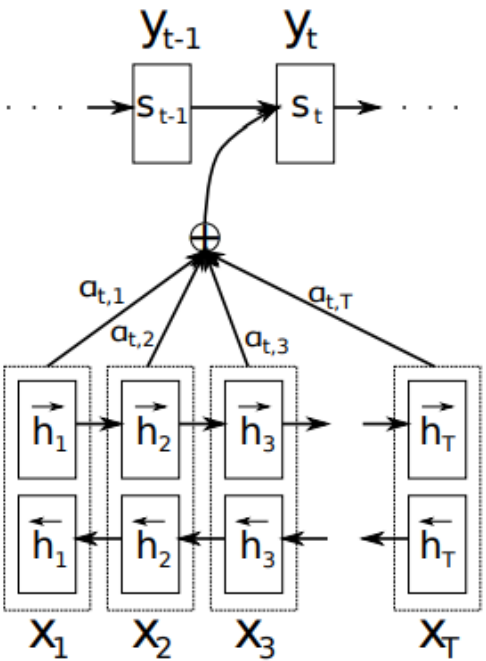


Figure 1: The graphical illustration of the proposed model trying to generate the t-th target word y_t given a source sentence $(x_1, x_2, \ldots, x_T)$.

Attention models

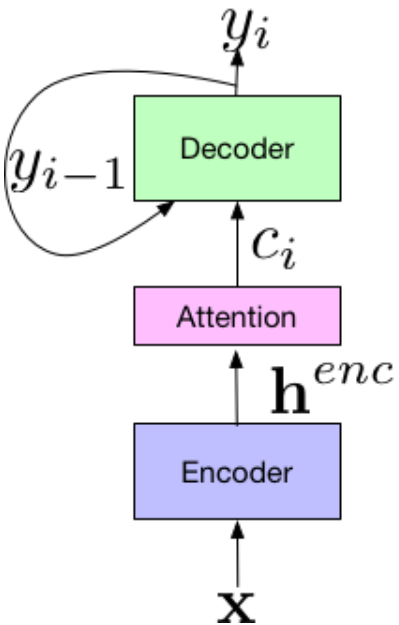


Image from https://ai.googleblog.com/2017/12/improving-end-to-end-models-for-speech.html

Listen, Attend and Spell (LAS), [Chan et al, 2015]

Listen, Attend and Spell

William Chan
Carnegie Mellon University
williamchan@cmu.edu

Navdeep Jaitly, Quoc V. Le, Oriol Vinyals
Google Brain
{ndjaitly,qvl,vinyals}@google.com

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

LAS is a character-level RNN encoder-decoder with attention

Listen, Attend and Spell (LAS, 2015)

- Listen:

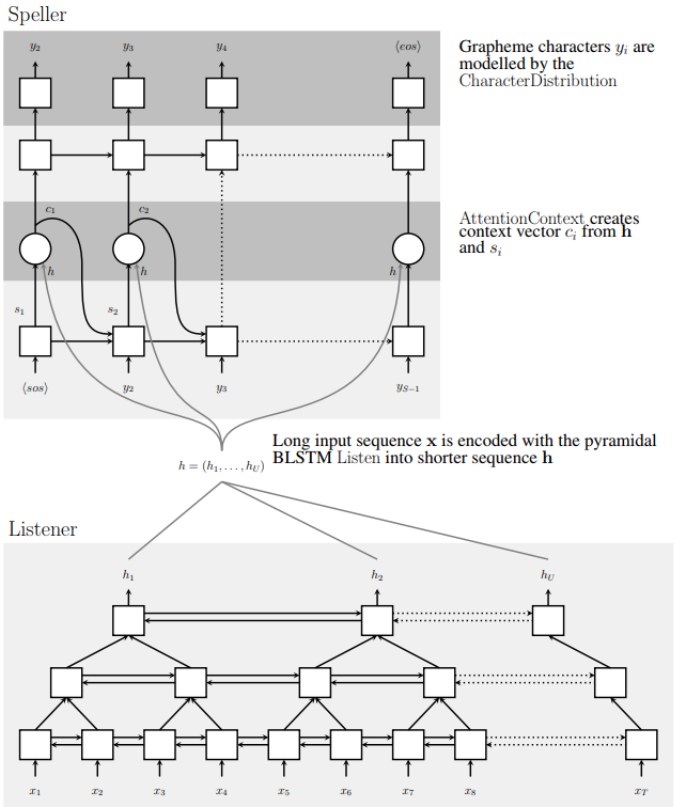
  $\mathbf{h} = \mathrm{BLSTM}(x)$

- Attend:

  $c_i = \mathrm{AttentionContext}(s_i, \mathbf{h})$

- Spell:

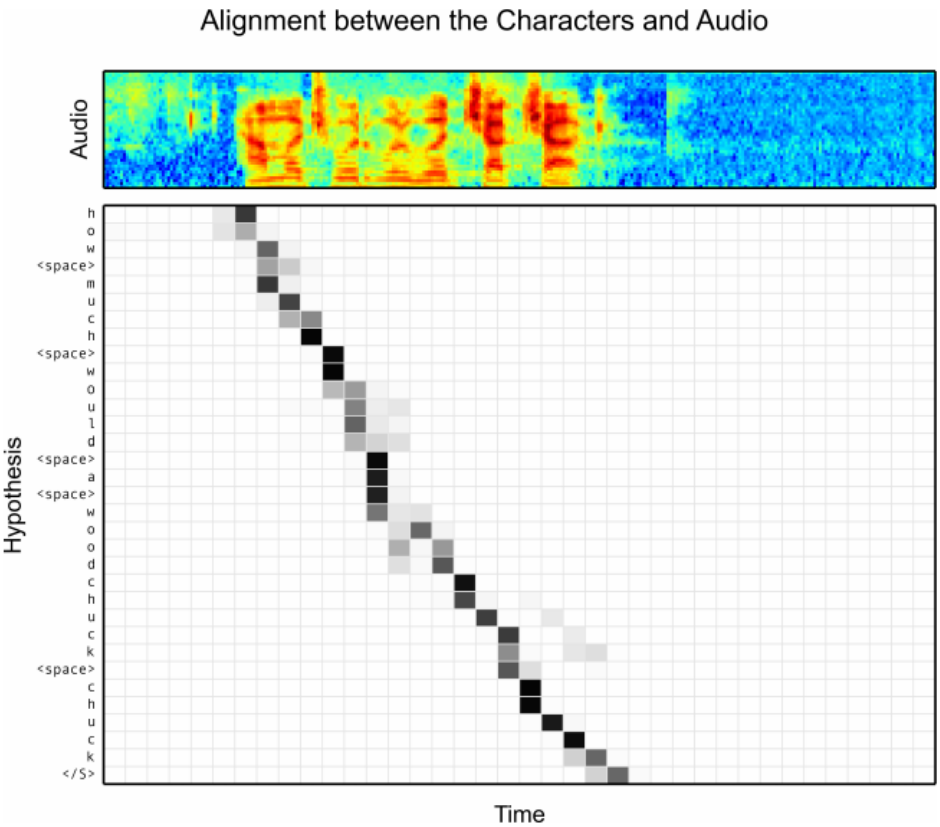
  $s_i = \mathrm{RNN}(s_{i-1}, y_{i-1}, c_{i-1})$



Decoding and results

Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0

- Training set: Google voice search task, 2000 h, test set: 10 h
- Decoding: character-level beam search
- No need of a lexicon to constraint the search space to valid words, LAS spells real words



How much would a woodchuck chuck

And the winner is...

STATE-OF-THE-ART SPEECH RECOGNITION WITH SEQUENCE-TO-SEQUENCE MODELS

Chung-Cheng Chiu, Tara N. Sainath, Yonghui Wu, Rohit Prabhavalkar, Patrick Nguyen, Zhifeng Chen, Anjuli Kannan, Ron J. Weiss, Kanishka Rao, Ekaterina Gonina, Navdeep Jaitly, Bo Li, Jan Chorowski, Michiel Bacchiani

Google, USA

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

Table 5: Resulting WER on voice search (VS)/dictation (D). The improved LAS outperforms the conventional LFR system while being more compact. Both models use second-pass rescoring.

Training set: 12,500 hours, 15M English sentences

Conclusions on end-to-end ASR

- Sequence-to-sequence (seq2seq) ASR is the new SOTA
- More precisely attention-based encoder-decoder architectures such as LAS are the new SOTA
- A single NN subsumes the acoustic, pronunciation and language model components
- requires massive data and compute resources
- $\rightarrow$ of any interest in our low-resource case?

A few words on transfer learning

MULTILINGUAL SEQUENCE-TO-SEQUENCE SPEECH RECOGNITION: ARCHITECTURE, TRANSFER LEARNING, AND LANGUAGE MODELING

Jaejin Cho[1,‡] , *Murali Karthick Baskar*[2,‡], *Ruizhi Li*[1,‡], *Matthew Wiesner*[1], *Sri Harish Mallidi*[3], *Nelson Yalta*[4], *Martin Karafiát*[2], *Shinji Watanabe*[1], *Takaaki Hori*[5]

[1]Johns Hopkins University, [2]Brno University of Technology, [3]Amazon, [4]Waseda University, [5]Mitsubishi Electric Research Laboratories (MERL)

- Multilingual hybrid CTC-attention model as seed model
- Performance drops when finetuning only the decoder
- Gains by finetuning both the encoder and decoder