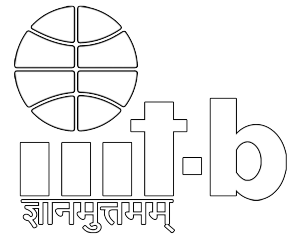


Attention-based end-to-end models for ASR

CTC, Attention and beyond

Shreekantha A Nadig [MS2016010]

August 13, 2019

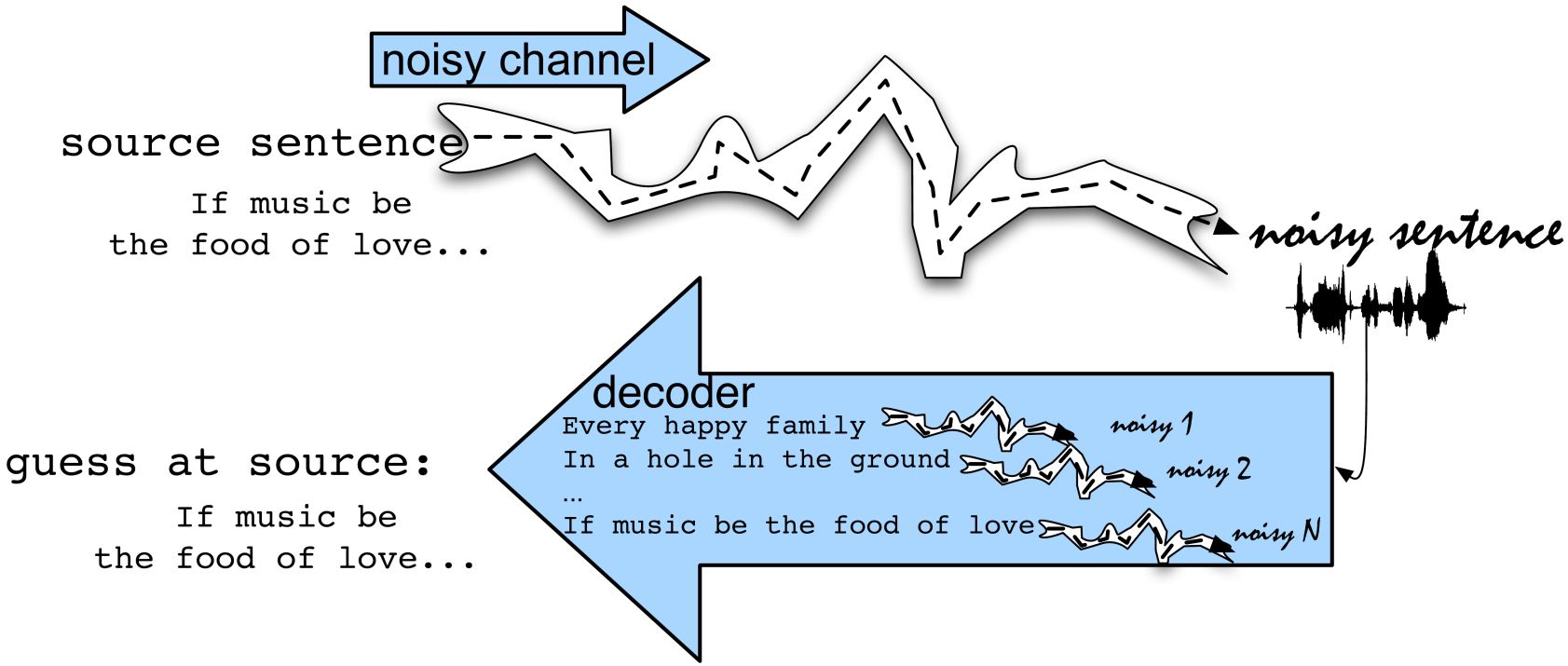
Under the guidance of
Prof. V.Ramasubramanian
Prof. Sachit Rao

Table of contents

ASR

The Noisy Channel Model

- Search through space of all possible sentences.
- Pick the one that is most probable given the waveform.



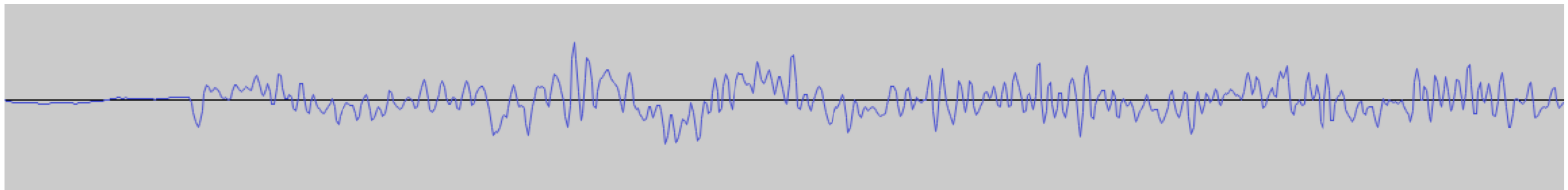
[1] **4767370**.

The Noisy Channel Model

- What is the most likely sentence out of all sentences in the language L given some acoustic input O?
- Treat acoustic input O as sequence of individual observations $O = o_1, o_2, o_3, ..., o_T$
- Define a sentence as a sequence of words: $W = w_1, w_2, w_3, ..., w_U$

Raw audio

- Simple 1D signal:

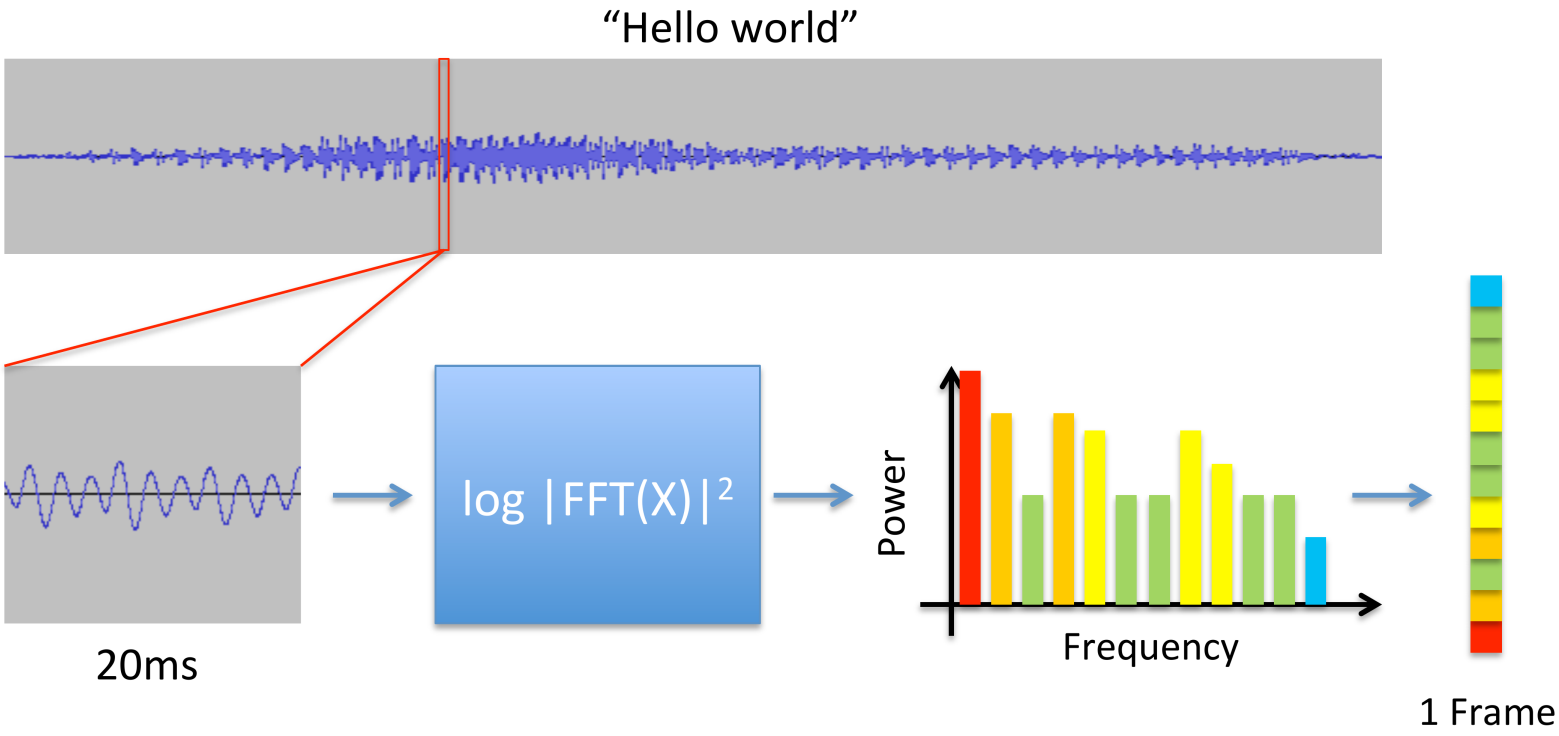


- Typical sample rates for speech: 8KHz, 16KHz.
- Each sample typically 8-bit or 16-bit
- 1D vector:

$$\mathbf{X} = [x_1, x_2, x_3, ...]$$

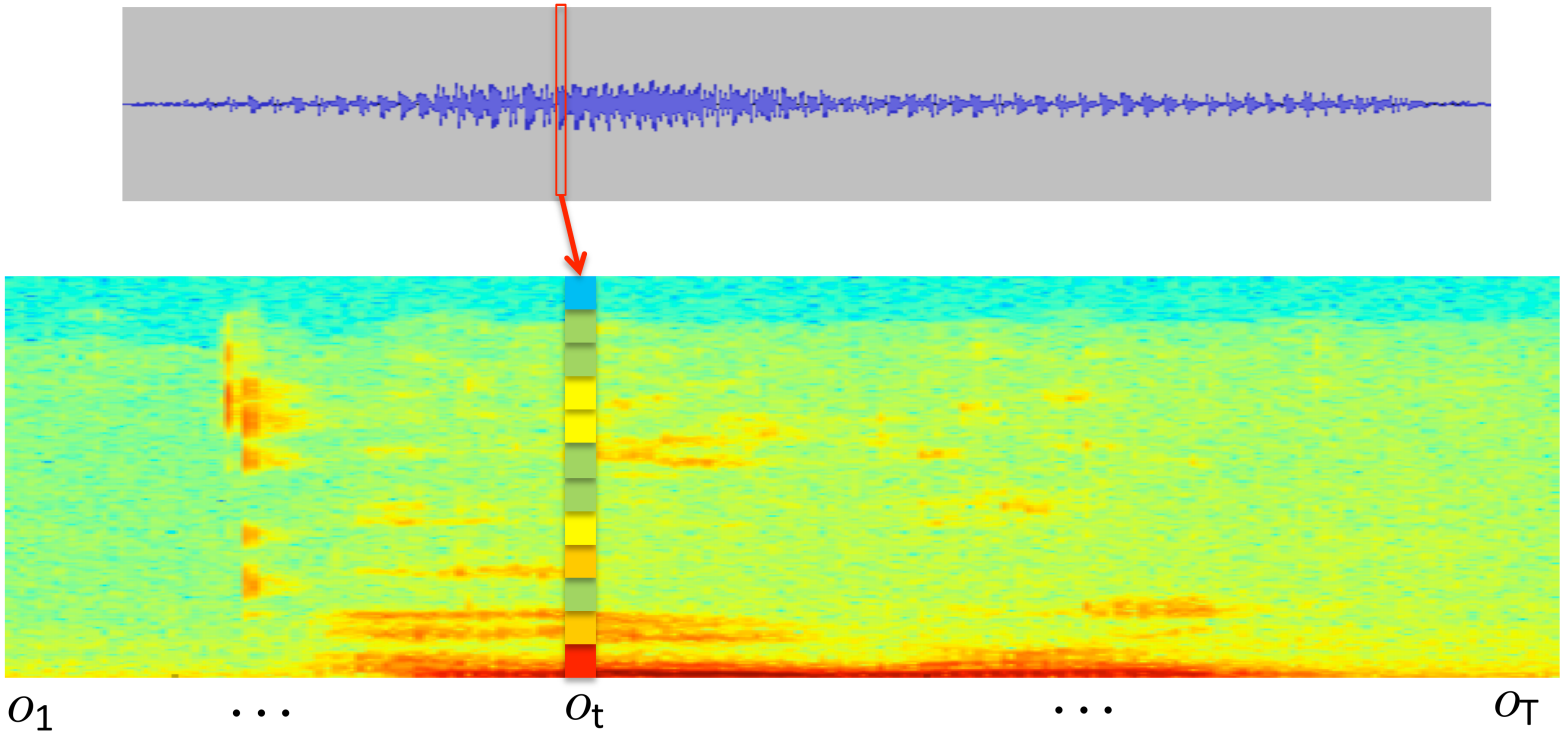
Spectrogram

- Take a small window (e.g., 20ms) of waveform
- Compute FFT and take magnitude. (i.e., power)
- Describes frequency content in local window.



Spectrogram

- Concatenate frames from adjacent windows to form "spectrogram"



The Noisy Channel Model - ASR equation

- Probabilistic implication: Pick the highest prob S:

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} P(W|O)$$

- We can use Bayes rule to rewrite this:

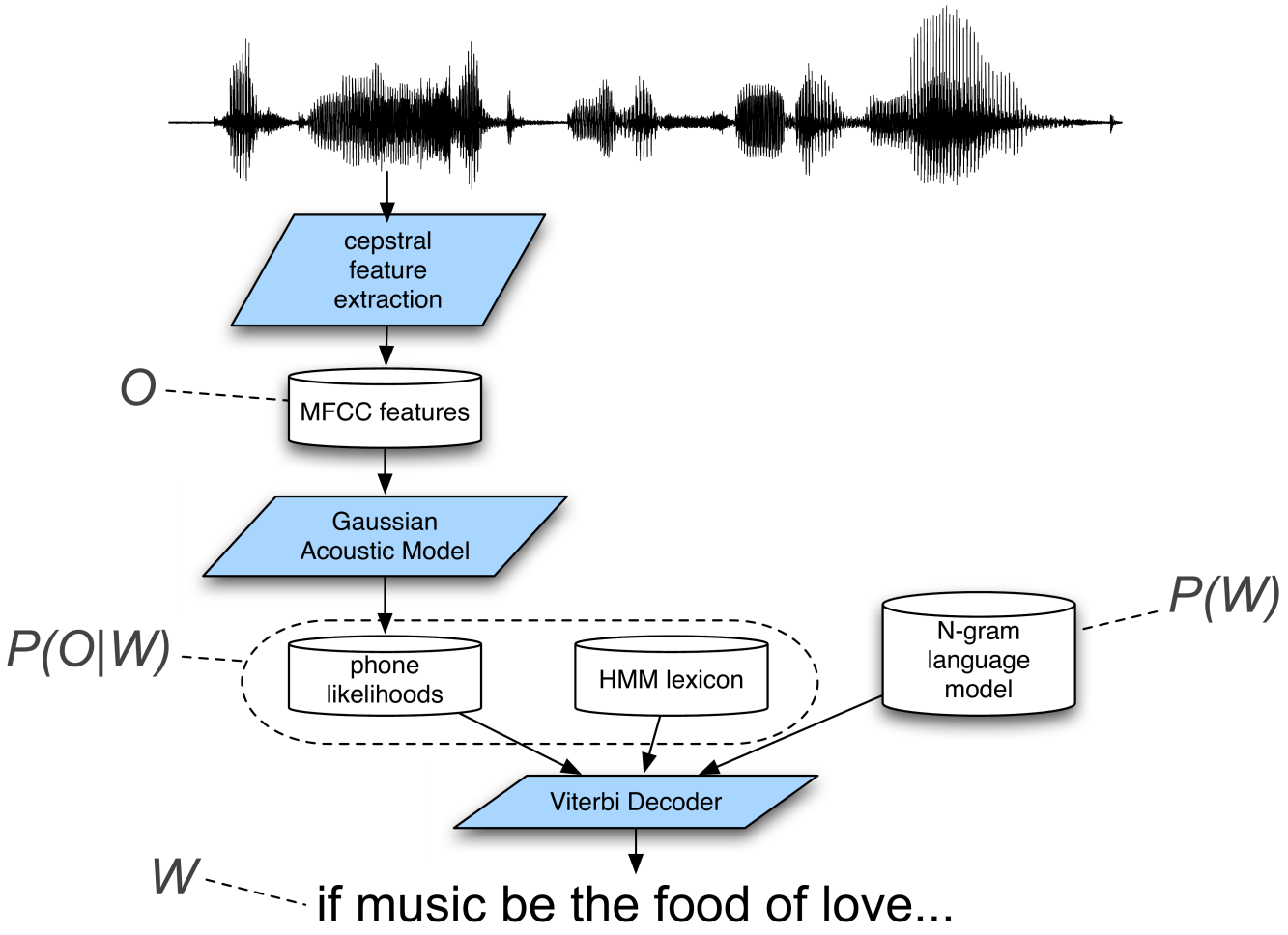
$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} \frac{P(O|W)P(W)}{P(O)}$$

- Since denominator is the same for each candidate sentence W, we can ignore it for the argmax:

$$\hat{W} = \underset{W \in L}{\operatorname{argmax}} P(O|W)P(W)$$

Before end-to-end

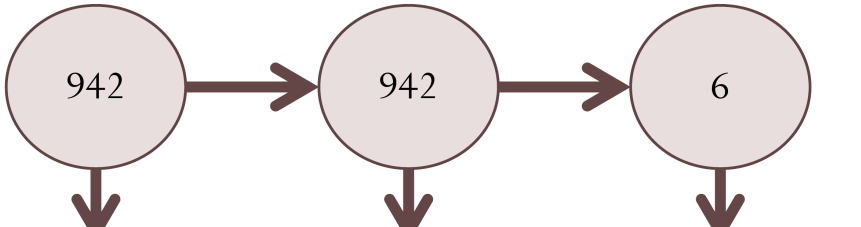
Speech Recognition Architecture



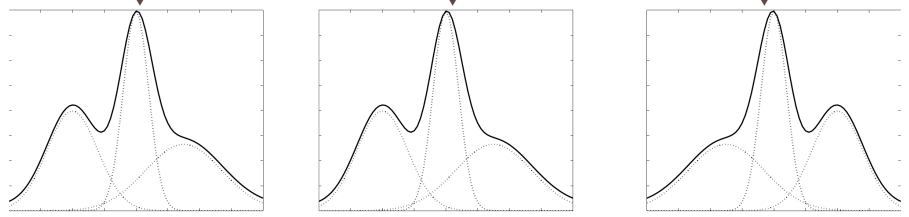
Acoustic Modeling with GMMs

Transcription: Samson
Pronunciation: S – AE – M – S –AH – N
Sub-phones : 942 – 6 – 37 – 8006 – 4422 …

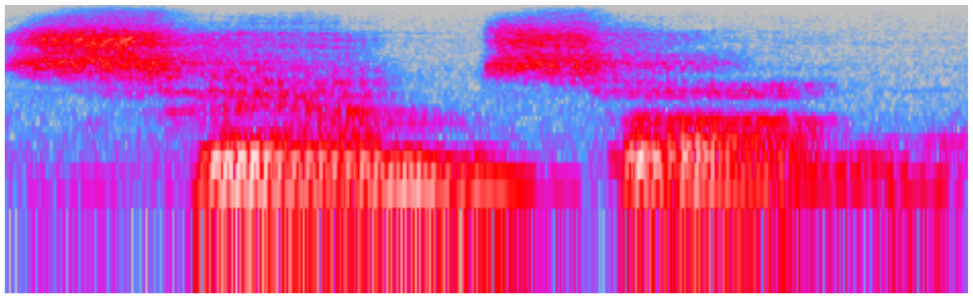
Hidden Markov Model (HMM):



Acoustic Model:



Audio Input:

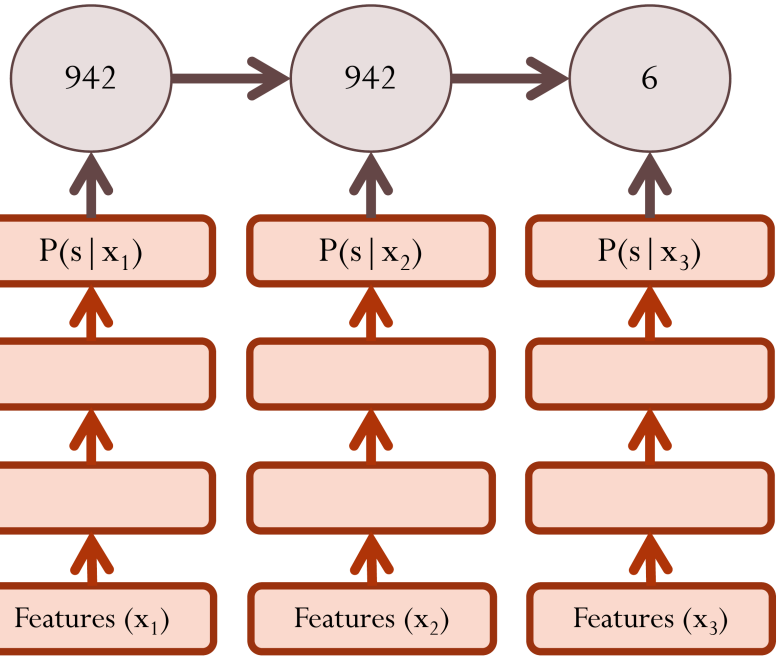


GMM models:
P(x|s)
x: input features
s: HMM state

DNN Hybrid Acoustic Models

Transcription: Samson
Pronunciation: S – AE – M – S –AH – N
Sub-phones : 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):



Acoustic Model:

Audio Input:

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

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

DNN * Constant / State prior

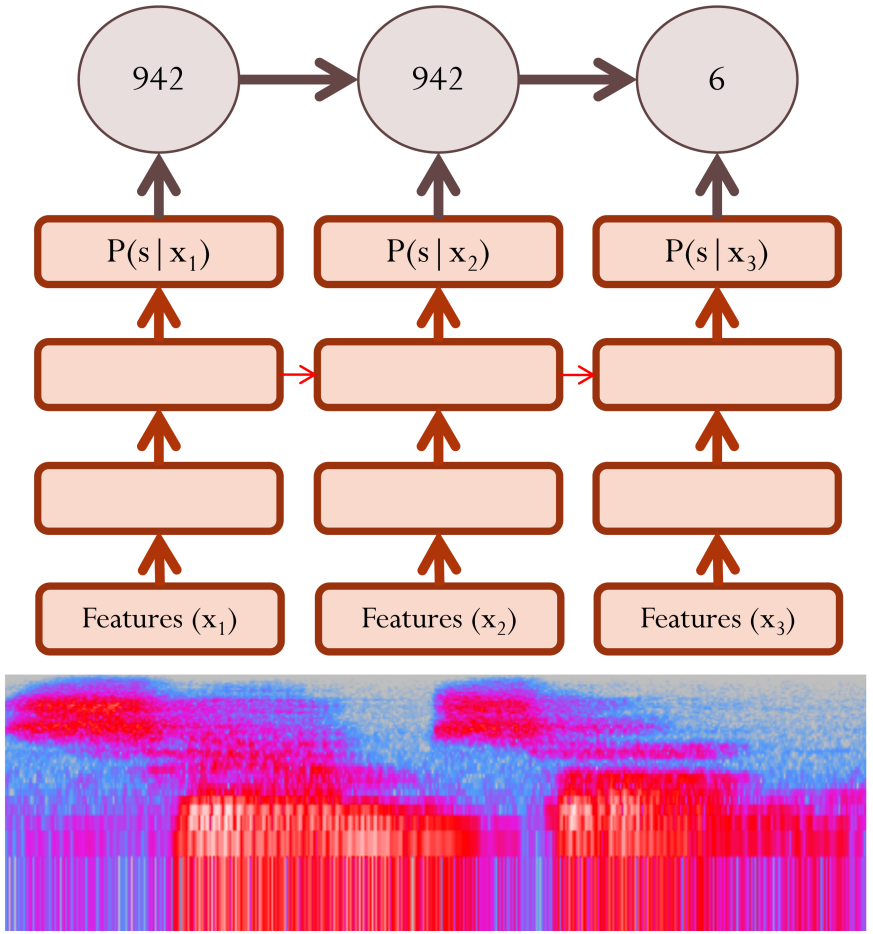
Recurrent DNN Hybrid Acoustic Models

Transcription: Samson
Pronunciation: S – AE – M – S –AH – N
Sub-phones : 942 – 6 – 37 – 8006 – 4422 …

Hidden Markov Model (HMM):

Acoustic Model:

Audio Input:



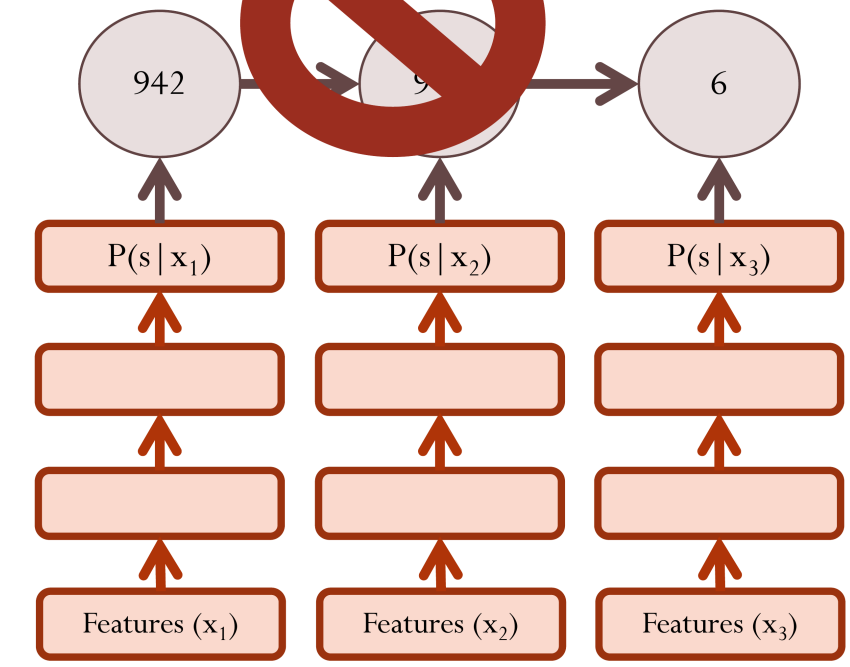
HMM-Free Recognition

Transcription: Samson

Pronunciation: S – AE – M … N

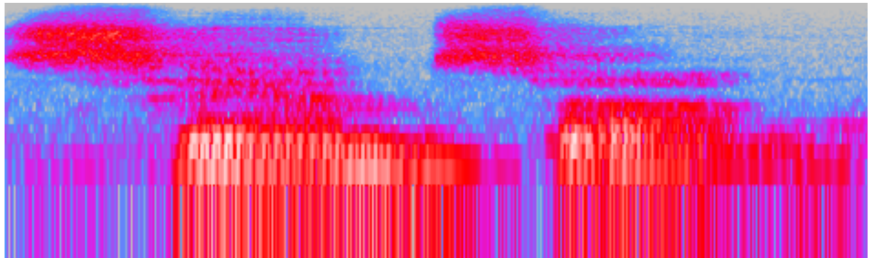
Sub-phones : 942 – 6 … 8006 … 22 …

Hidden Markov Model (HMM):

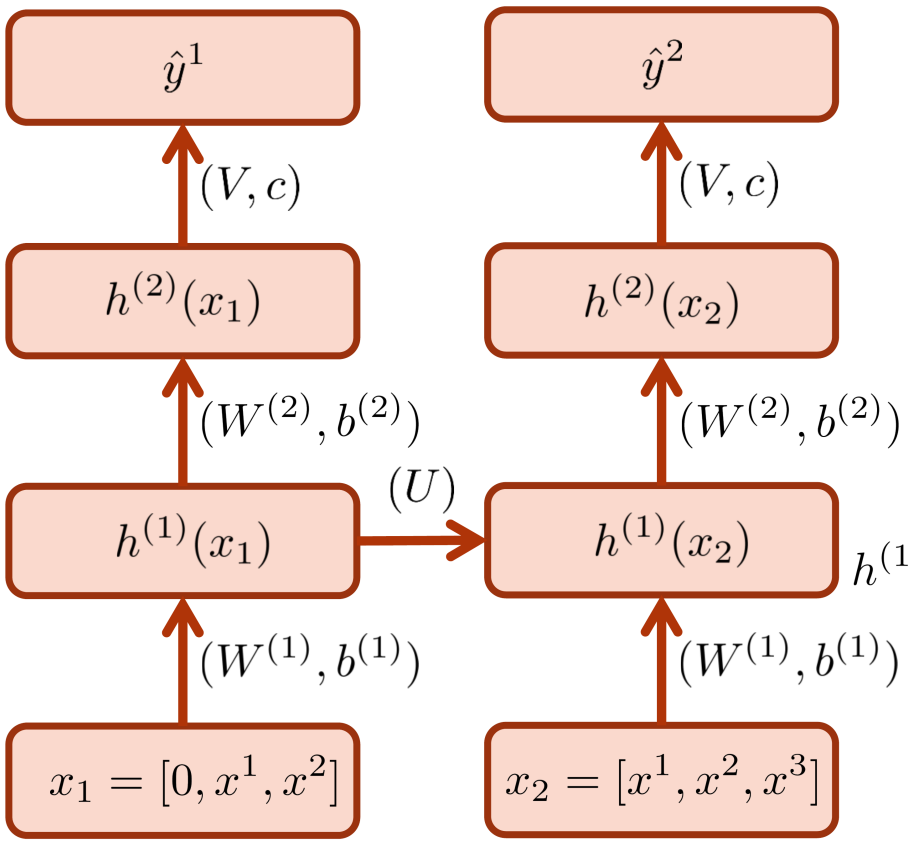


Acoustic Model:

Audio Input:



Deep Recurrent Network



Output Layer

$$\hat{y}^2 = Vh^{(1)}(x_2) + c$$

Hidden Layer

$$h^{(2)}(x_2) = \sigma(W^{(2)}h^{(1)}(x_2) + b^{(2)})$$

Hidden Layer

$$h^{(1)}(x_2) = \sigma(W^{(1)}x_2 + b^{(1)} + Uh^{(1)}(x_1))$$

Input

$$\mathbf{X}$$

HMM-Free Recognition

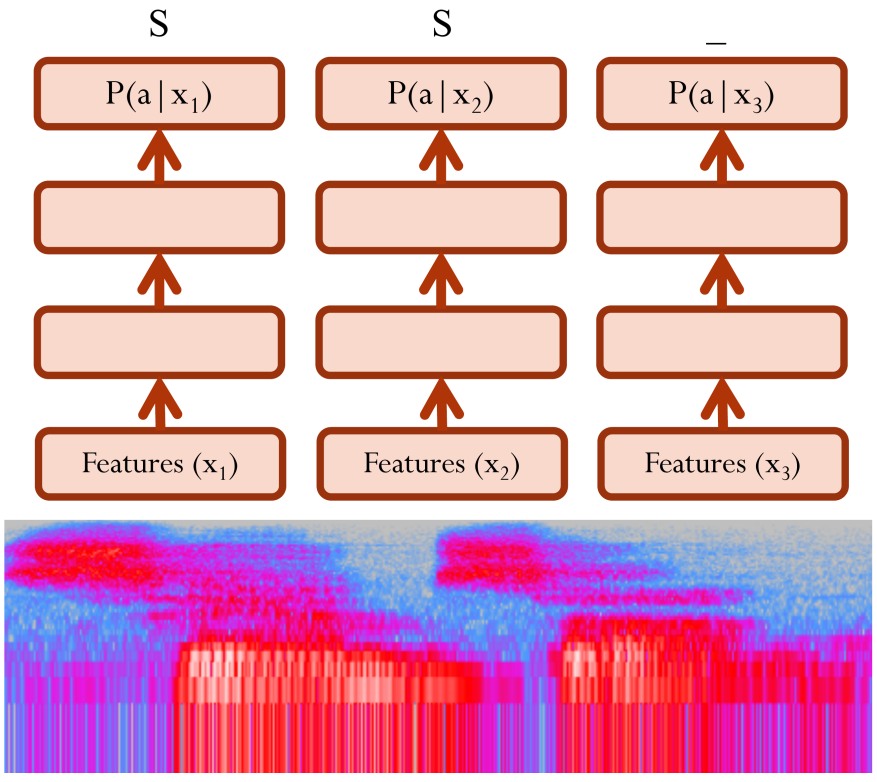
Transcription: Samson

Characters: SAMSON

Collapsing function: SS___AA_M_S___O___NNNN

Acoustic Model:

Audio Input:



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

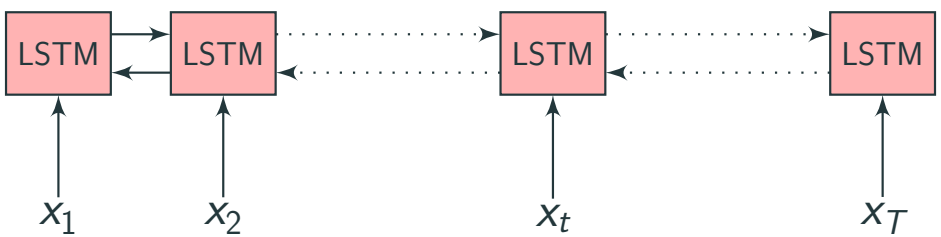
The distribution over *characters*

Challenges in traditional ASR and Motivation for end-to-end

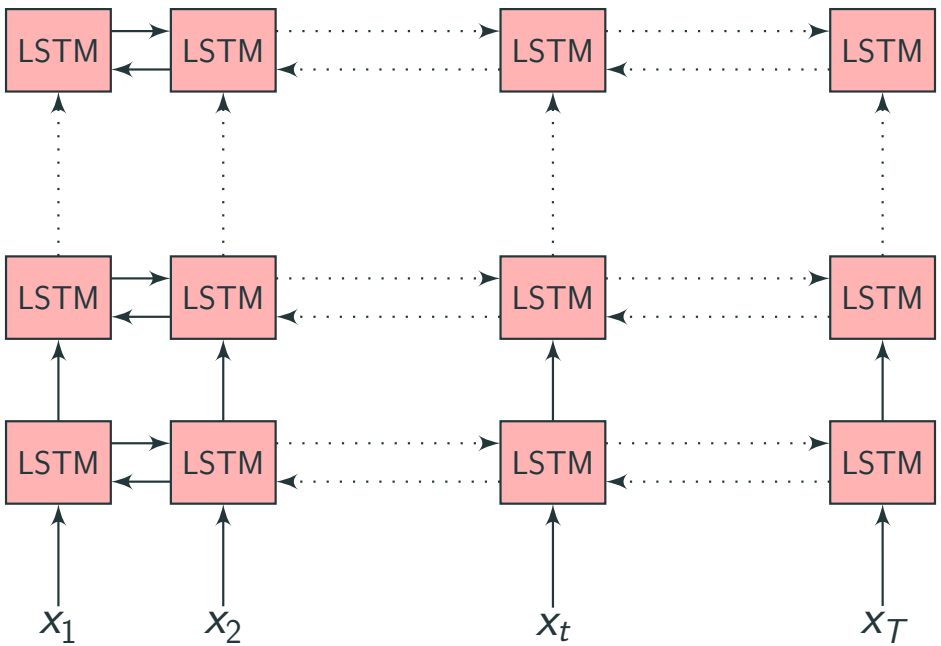
- Step-wise refinement (GMM-HMM -> DNN-HMM)
- Linguistic information (pronunciation dictionary)
- Conditional independence assumptions (Markov assumptions)
- Complex decoding (HCLG graph)
- Incoherence in optimization
- Traditional pipeline is highly tweakable, but also hard to get working well
- Historically, each part of system has own set of challenges (eg. choosing feature representation)
- Popular loss functions (MSE, XE) assume a one-to-one correspondence between the network's output and target labels
- But what if there is no one-to-one correspondence? (Speech Recognition, Machine Translation, handwriting recognition)
- Need a way to find alignment between network outputs and target labels (CTC/Attention/seq2seq)
- Traditionally, try to bootstrap alignment - difficult!

End-to-end

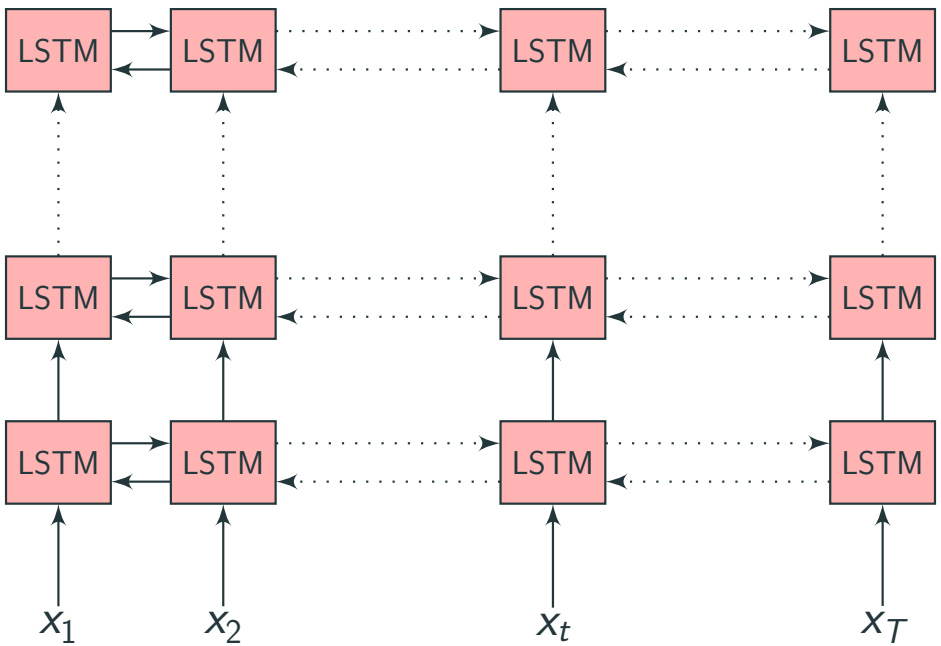
BLSTM Encoder



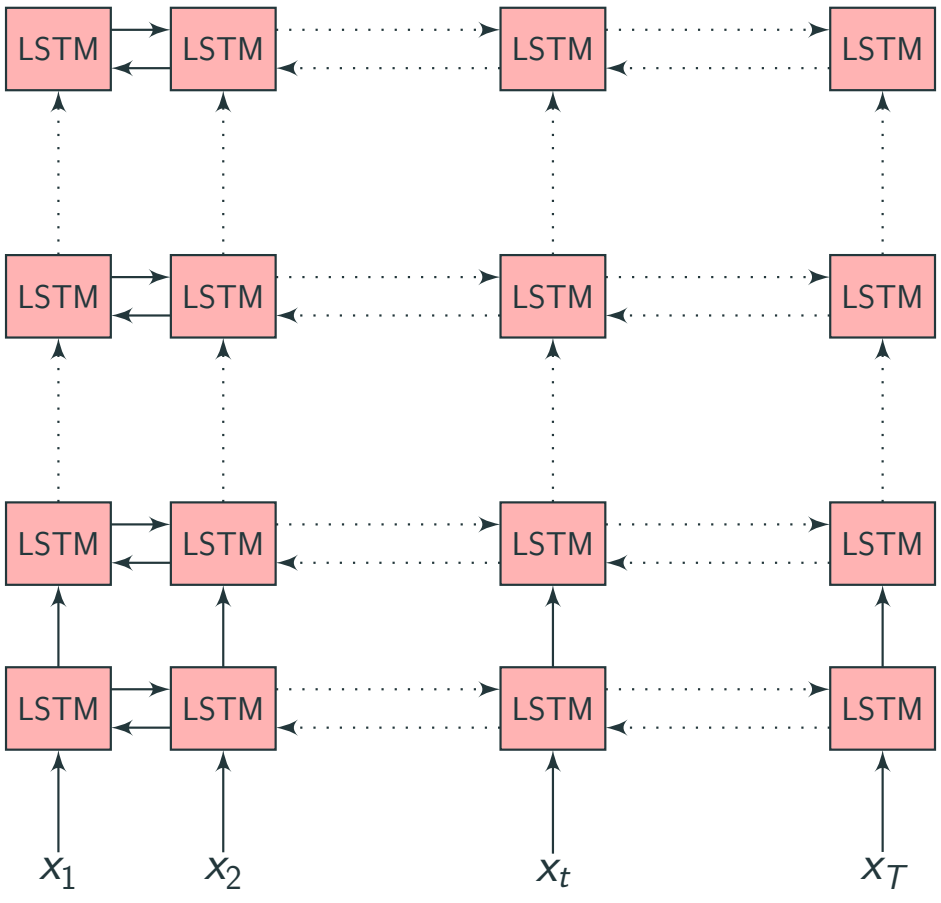
BLSTM Encoder



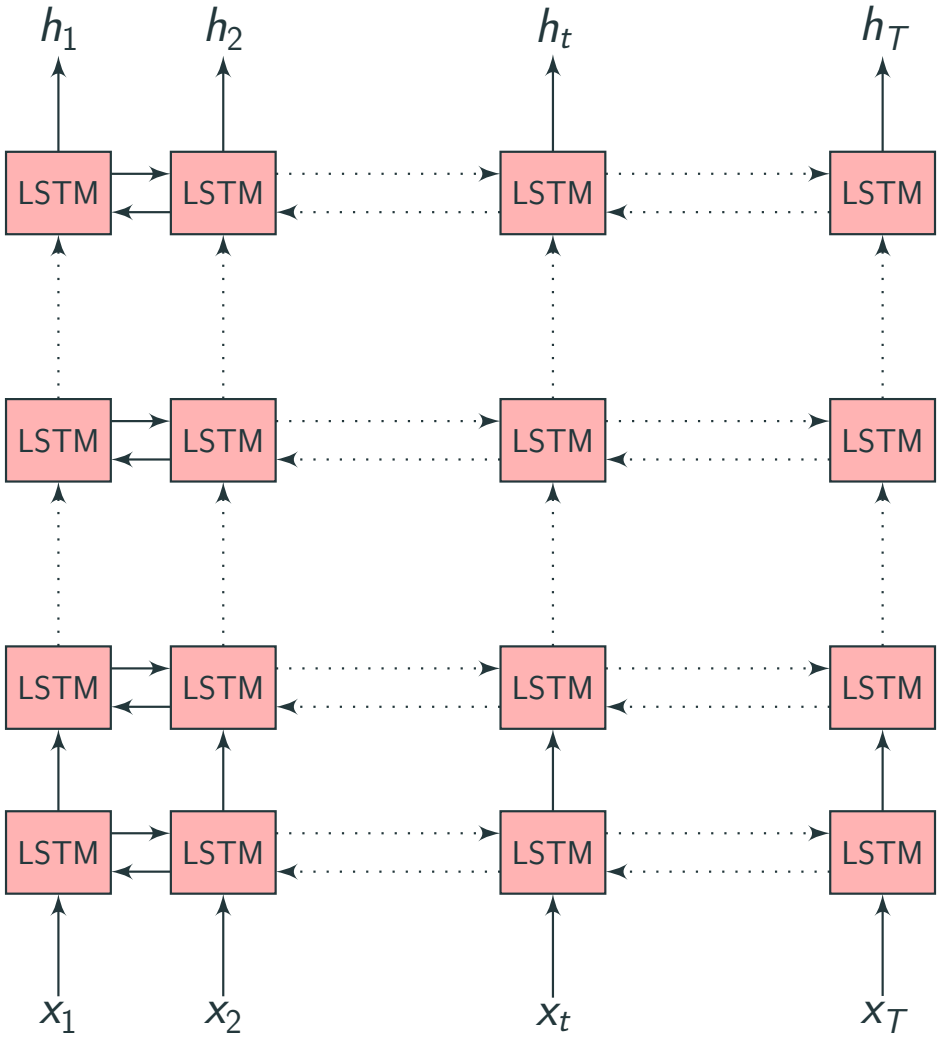
BLSTM Encoder



BLSTM Encoder



BLSTM Encoder



CTC - Connectionist Temporal Classification

- Maximum likelihood training of transcript
- Intuition: Alignments are unknown, so integrate over all possible time-character alignments

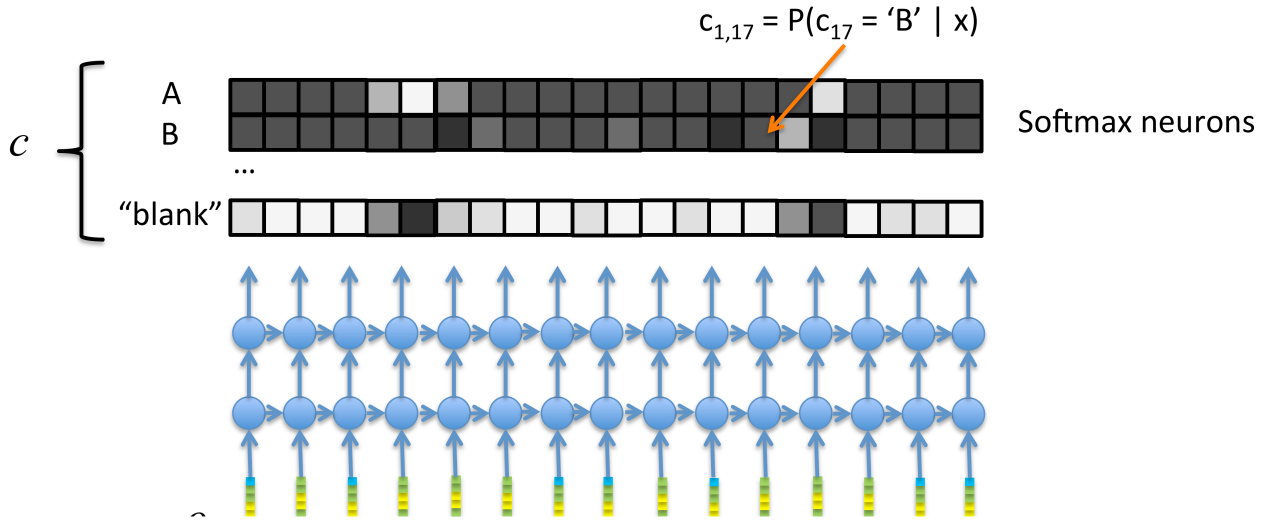
$$L_{ctc}(X, W) = \sum_{C:k(C)=W} p(C|X)$$

$$= \sum_{C:k(C)=W} \prod_{t=1}^{T} p(c_t|X)$$

- Example: W = "hi", T=3
  possible C such that k(C)=W:
  hhi,hii,_hi,h_i,hi_

CTC - Connectionist Temporal Classification

- RNN output neurons c encode distribution over symbols.
  Note length(c) == length(x)
- For grapheme based model: $c \in \{A, B, C, D, ..., Z, blank, space\}$
- Define a mapping $\beta(c) = y$
- Maximize likelihood of y^* under this model



$$P(c = HHH\_E\_LL\_LO\_\_\_|x) = P(c_1 = H|x)P(c_2 = H|x)...P(c_{15} = blank|x)$$

CTC - Connectionist Temporal Classification

- Define a mapping $\beta(c) -> y$
- Given a specific character sequence c, squeeze out duplicates + blanks to yield a transcription

$$y = \beta(c) = \beta(HHH\_E\_LL\_LO\_\_\_) = "HELLO"$$

- Mapping implies a distribution over possible transcripts y:

$P(c|x) =$

0.1	HHH_E__LL_LO___	"HELLO" v.1
0.02	HH__E__LL_LO___	"HELLO" v.2
0.01	HHH_E__L_L_OH__	"HELL OH"
0.01	HHH_EE_LL_L_O__	"HELLO" v.3
…	YY__E__LL_LO_W_	"YELLOW"

$$P(y|x) = \sum_{c:\beta(c)=y} P(c|x)$$

$P($"HELLO"$) =$ 0.1 + 0.02 + 0.01 + …

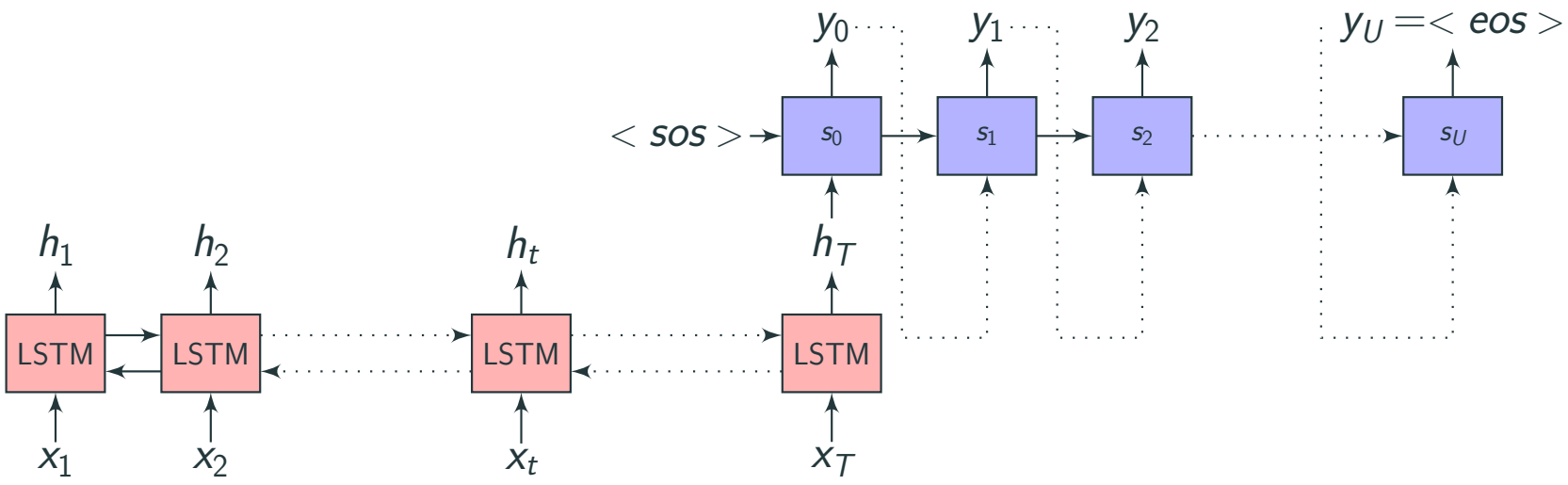
CTC - Connectionist Temporal Classification

- Update network parameters θ to maximize the likelihood of correct label y^*

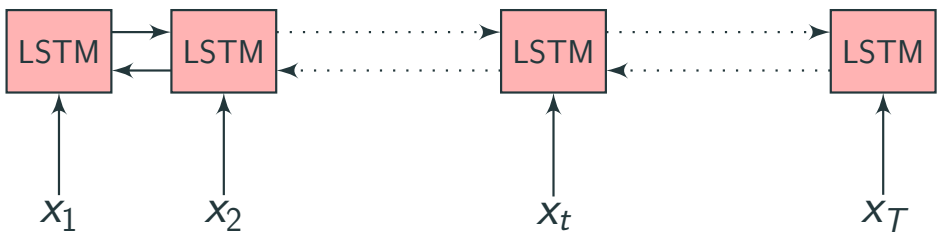
$$\theta^* = \underset{\theta}{\text{argmax}} \sum_i \log P(y^{*(i)}|x^i) = \underset{\theta}{\text{argmax}} \sum_i \log \sum_{c:\beta(c)=y^*(i)} P(c|x^i)$$

- Graves et al., 2006 provides an efficient dynamic programming algorithm to compute the inner summation and it's gradients.
- Off-the-shelf packages to compute CTC loss from c, y^* and gradients w.r.t. c
  - Baidu's warp-ctc: "https://github.com/baidu-research/warp-ctc"
  - PyTorch: "https://pytorch.org/docs/stable/nn.html#ctcloss"
  - TensorFlow: "https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss"
- Getting RNN-CTC to train well is tricky
  - Sortagrad - order utterances by length during first epoch
  - Batch normalization

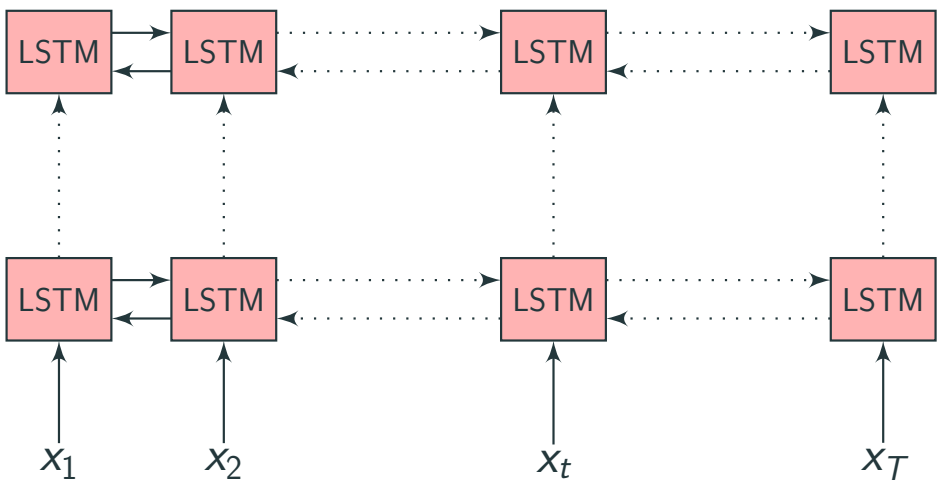
Basic Encoder-Decoder



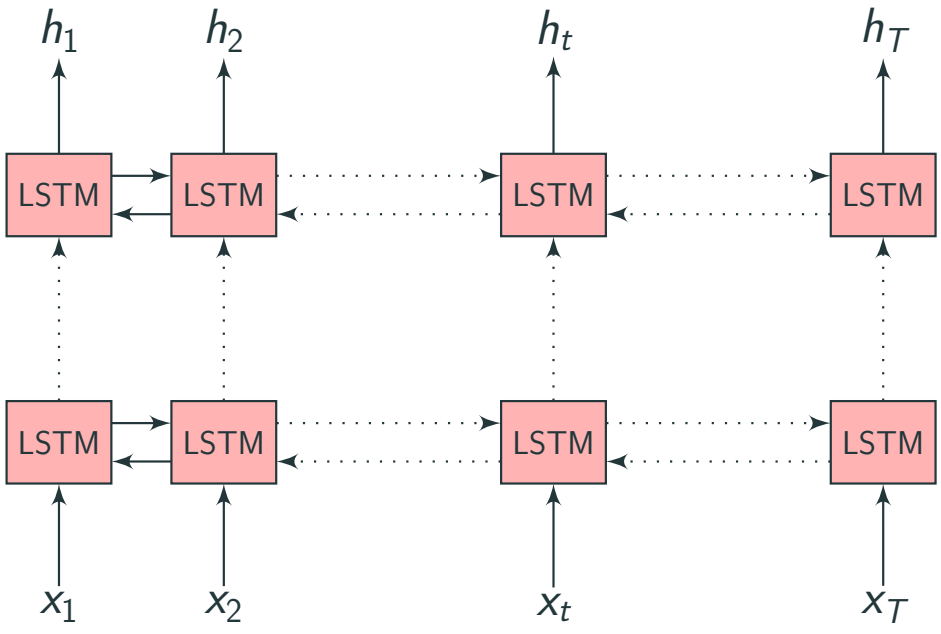
Encoder-Decoder



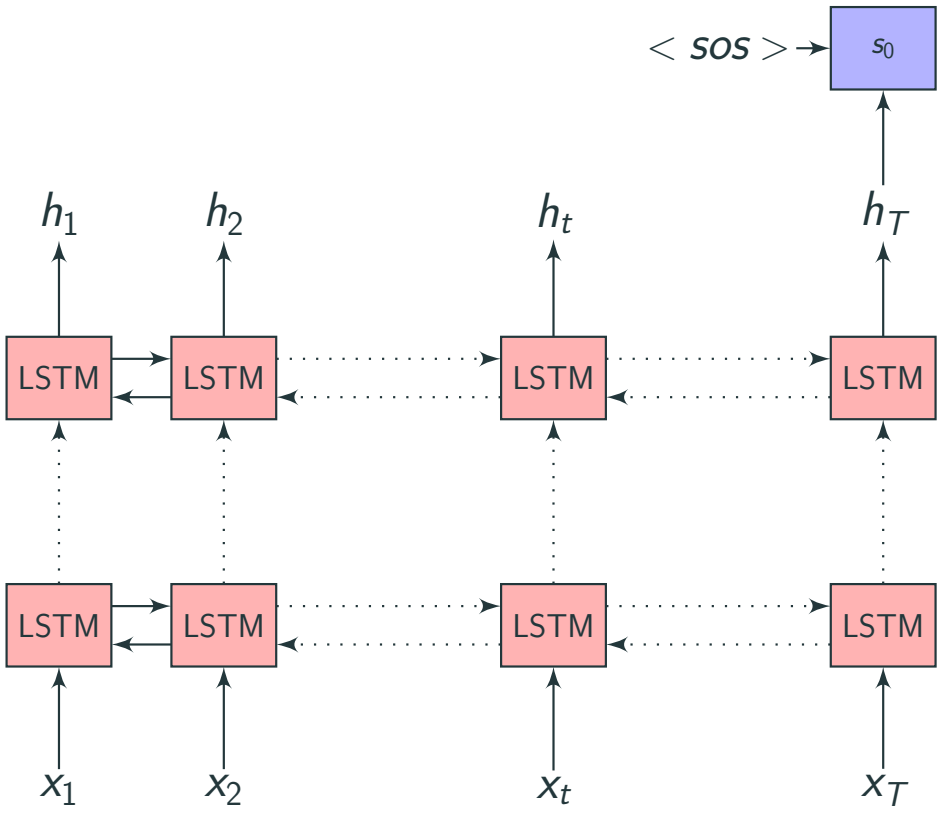
Encoder-Decoder



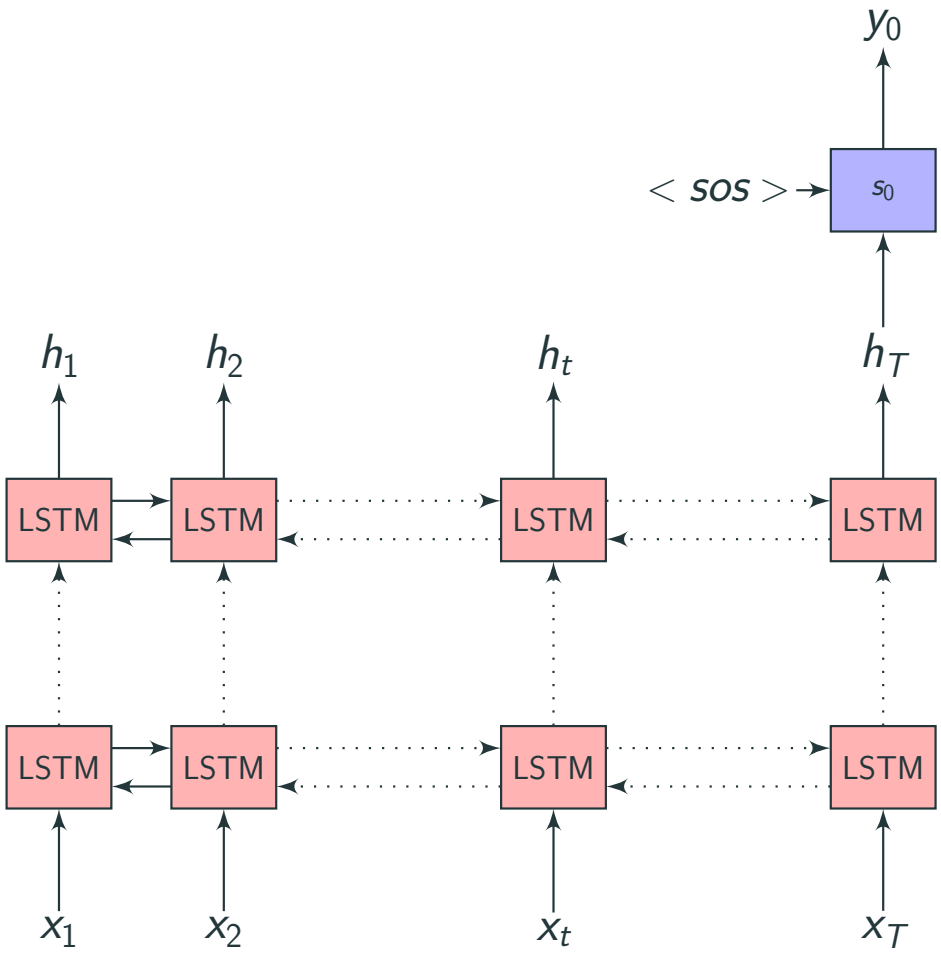
Encoder-Decoder



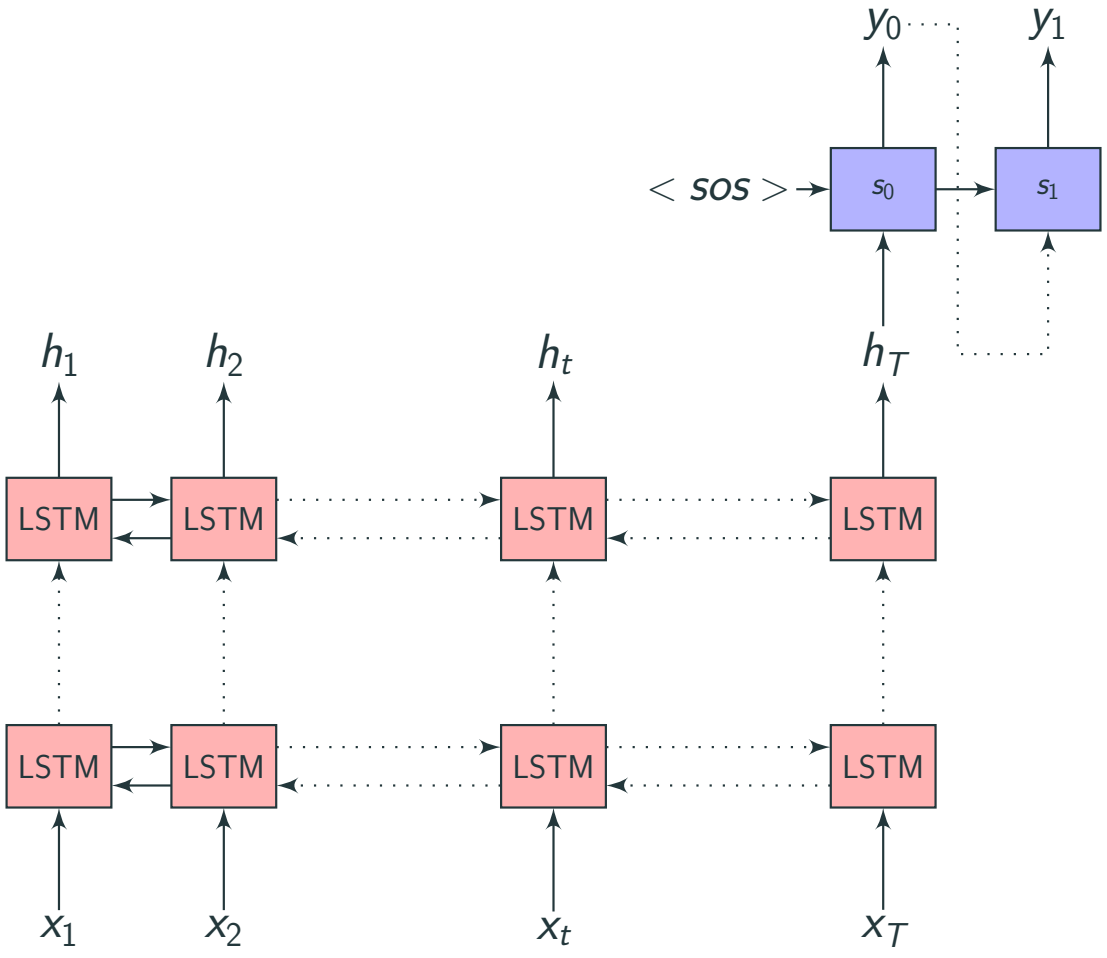
Encoder-Decoder



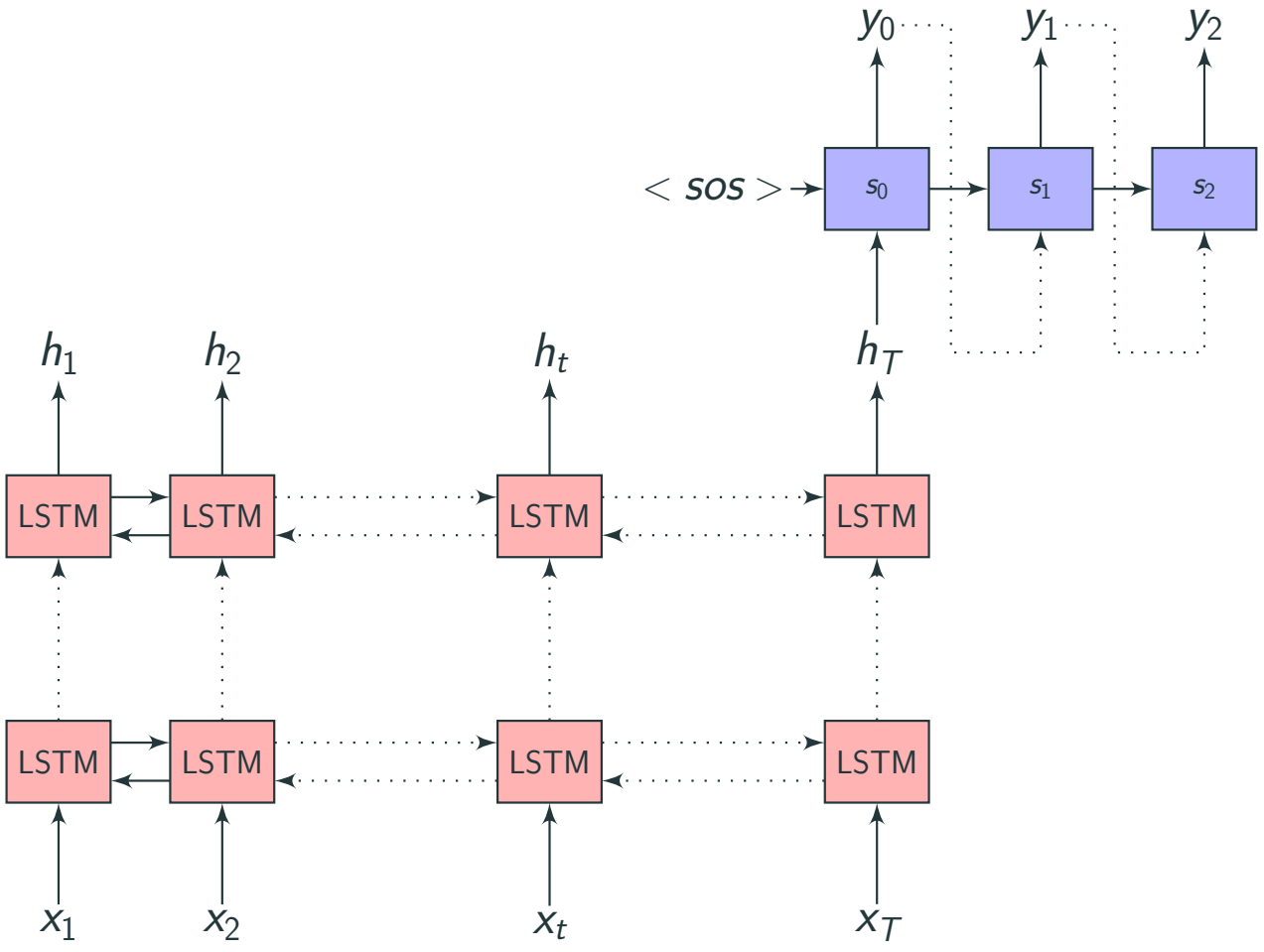
Encoder-Decoder



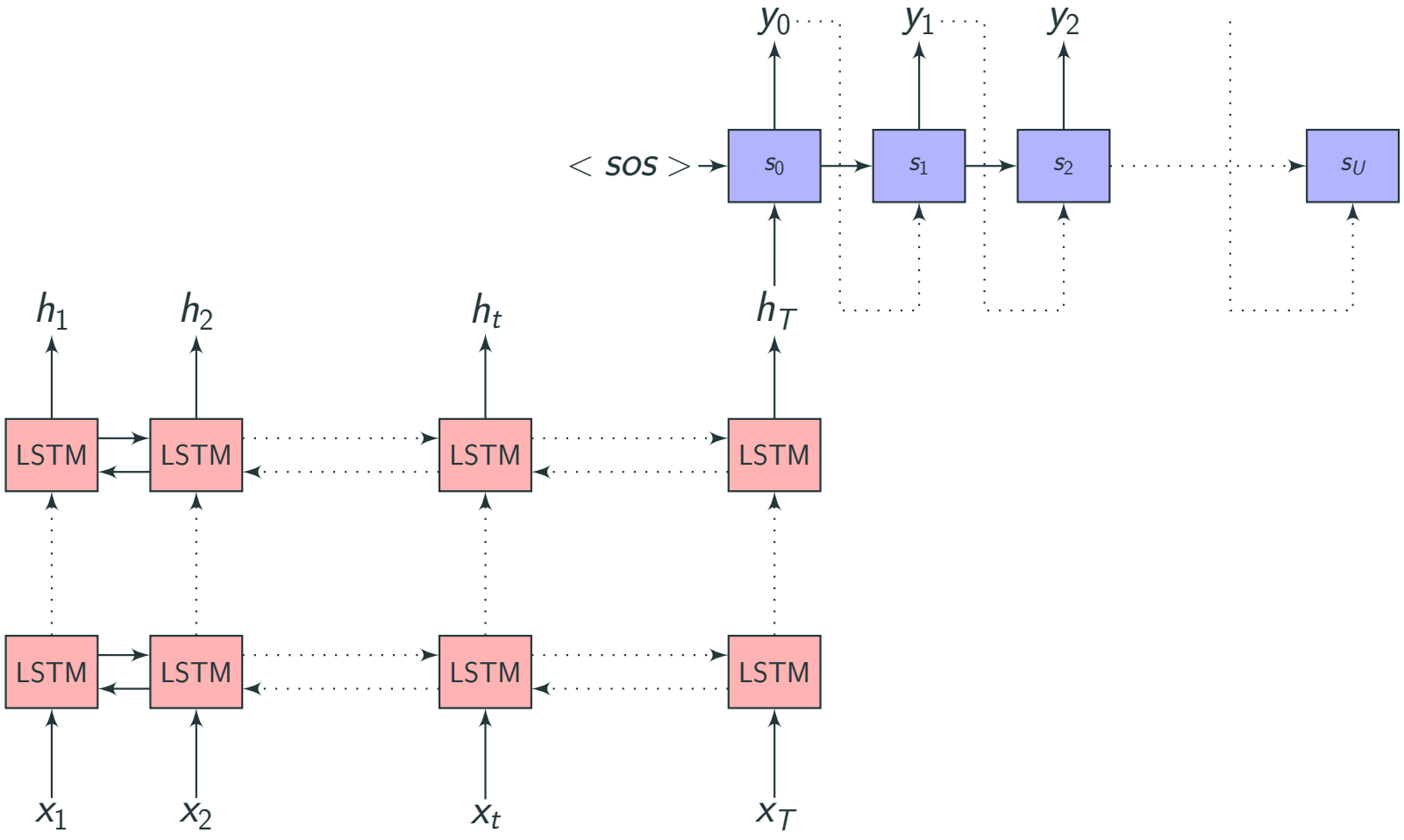
Encoder-Decoder



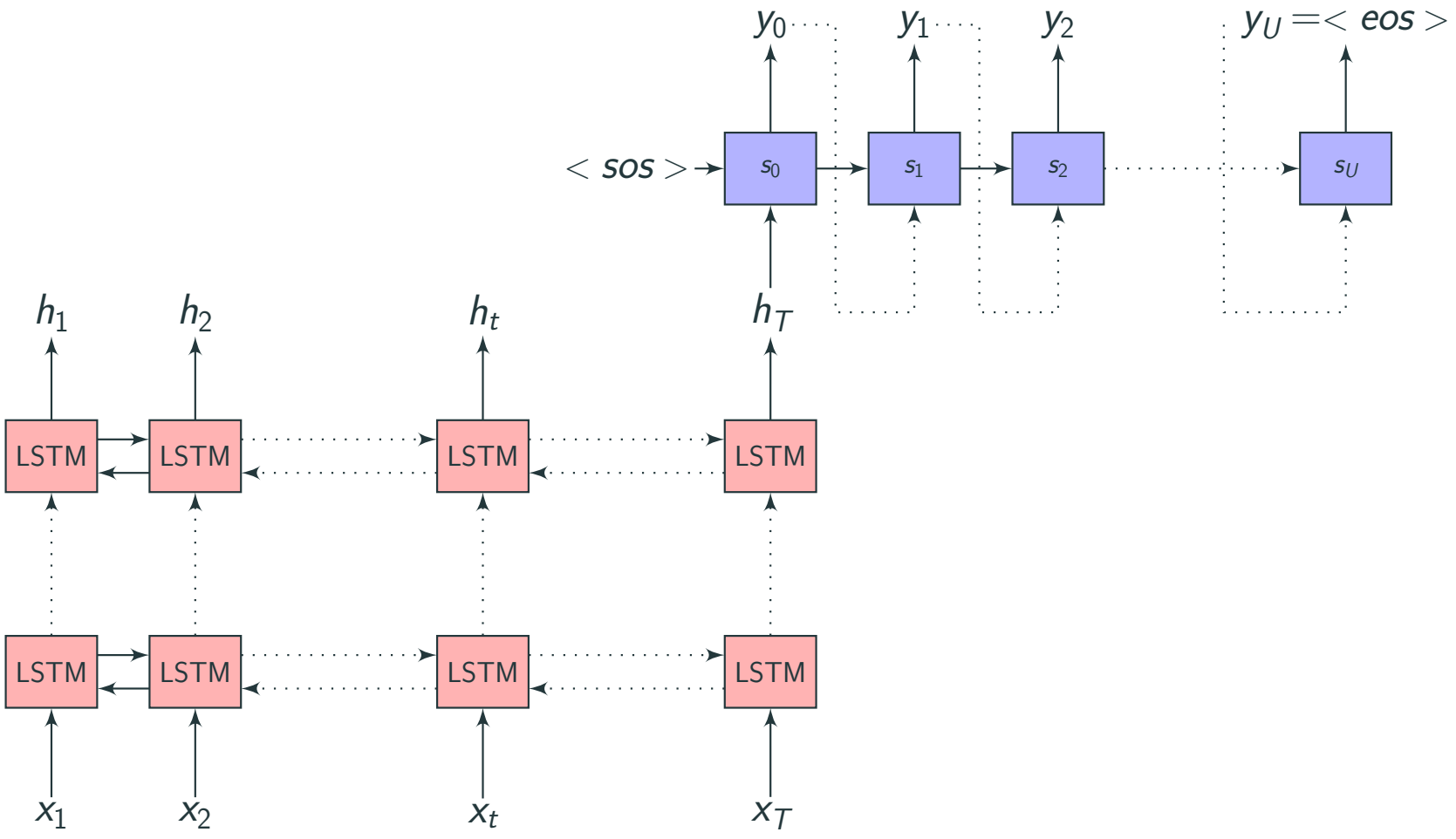
Encoder-Decoder



Encoder-Decoder

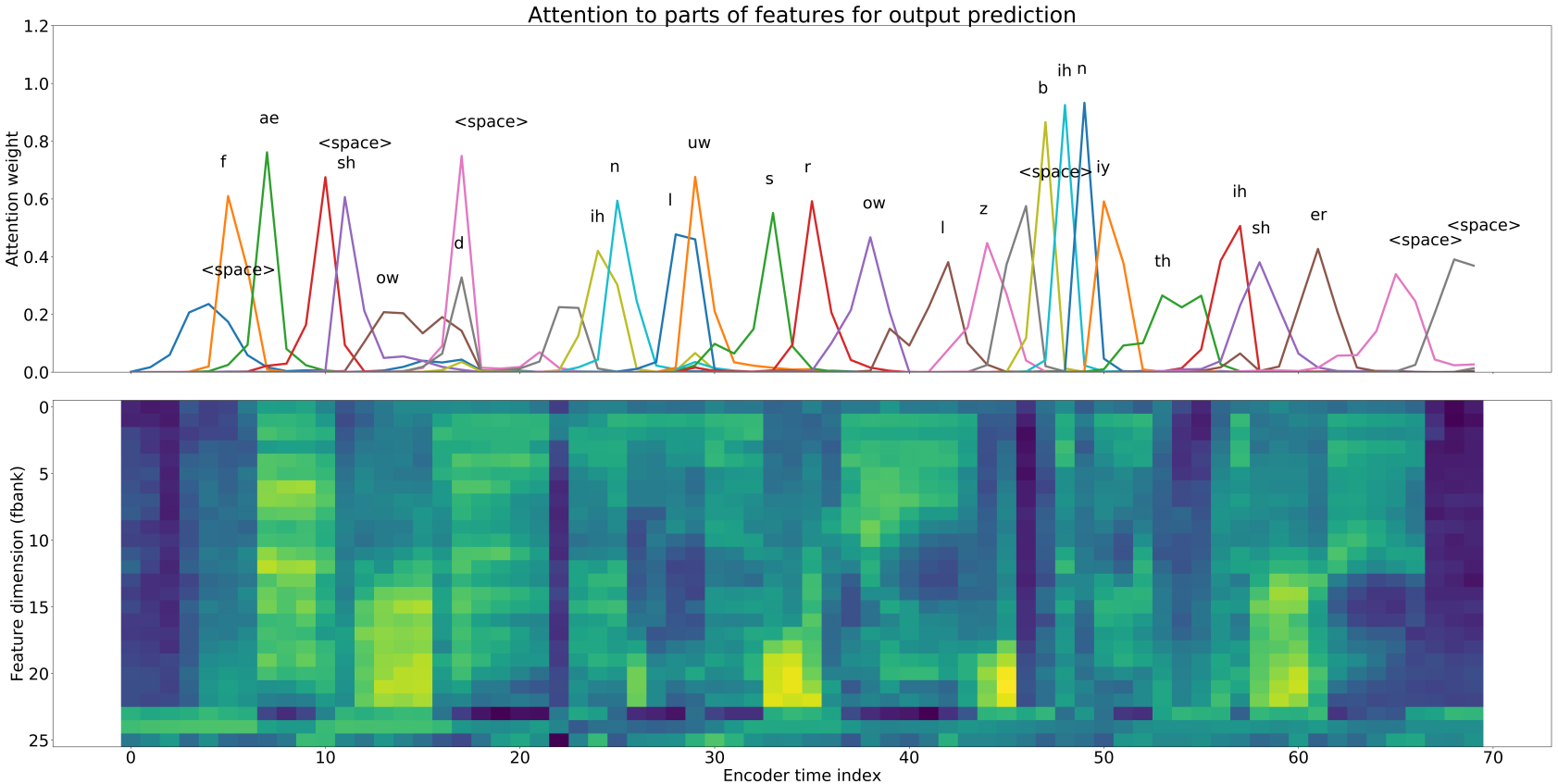


Encoder-Decoder

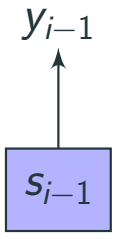


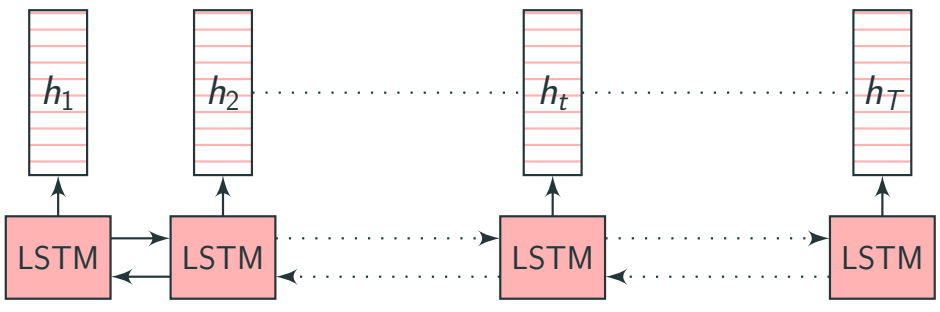
Attention

Attention

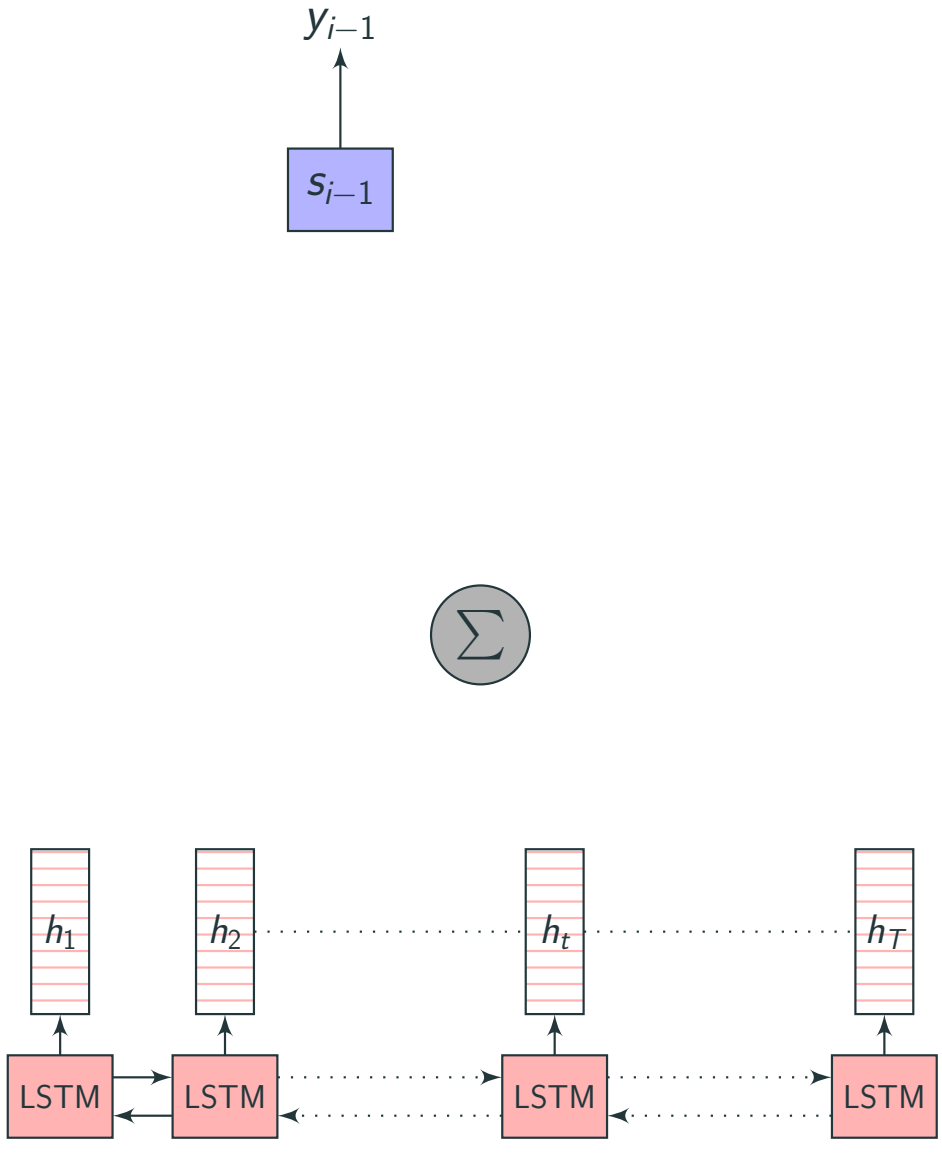

Attention to parts of features for output prediction
Attention weight
Feature dimension (fbank)
Encoder time index
<space>
f
ae
<space>
sh
ow
<space>
d
ih
n
l
uw
s
r
ow
l
z
<space>
b
ih
n
iy
th
ih
sh
er
<space>
<space>

Introduction to Attention

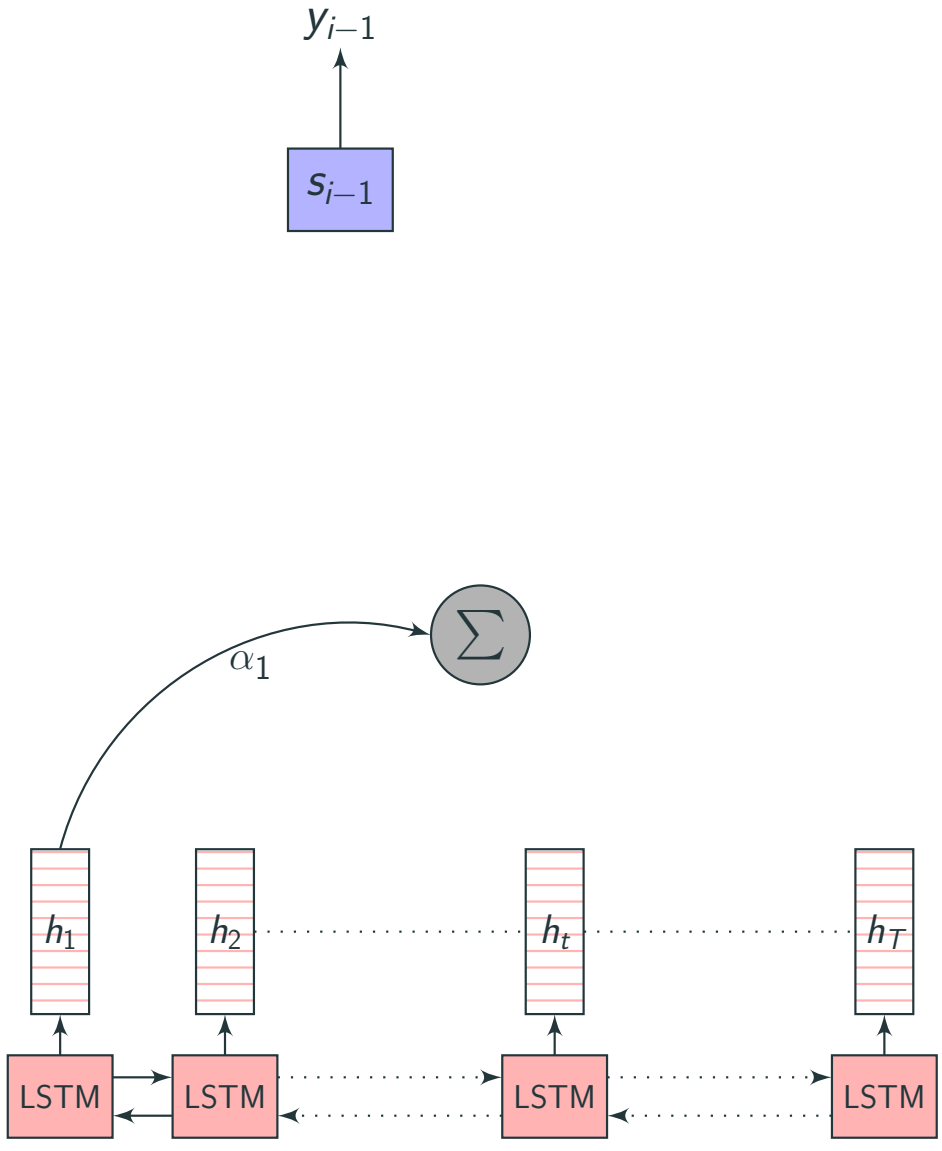




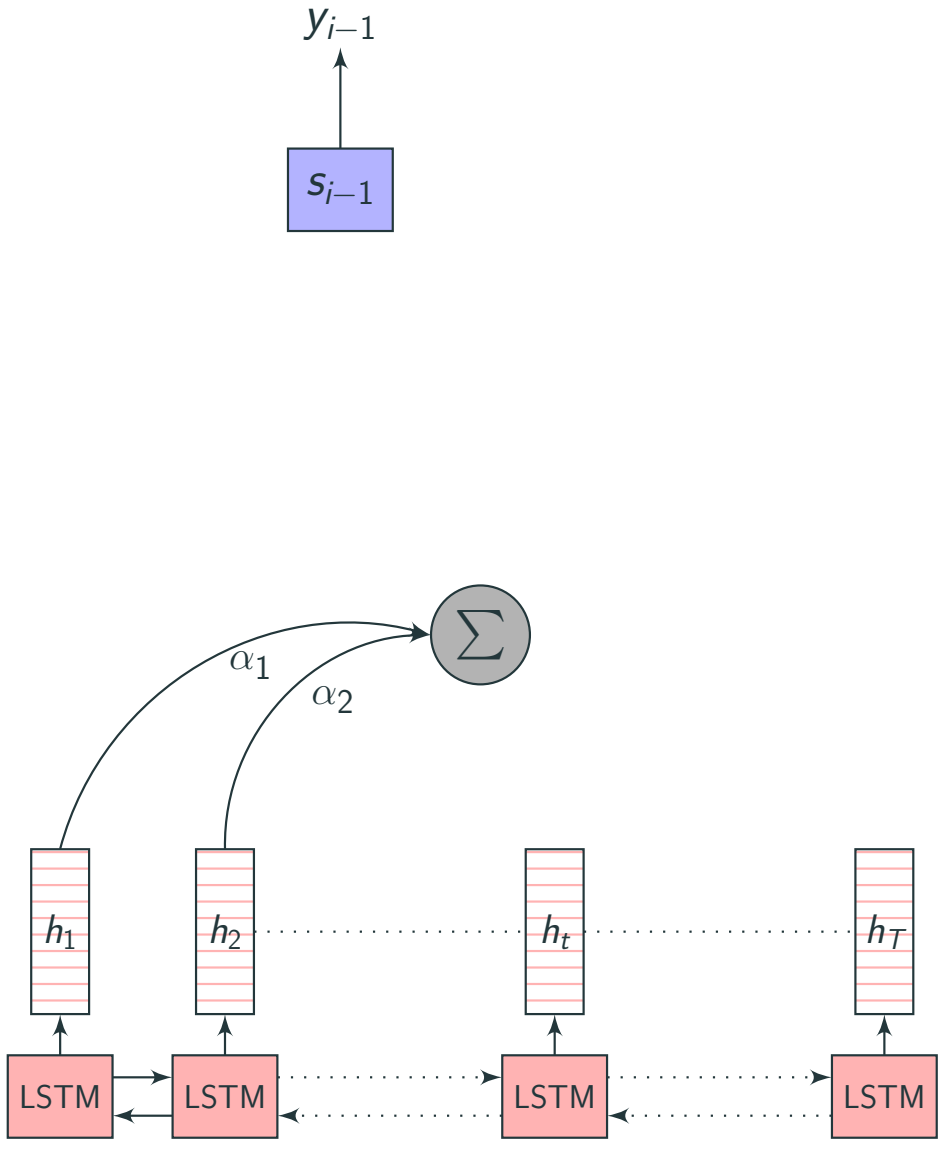
Introduction to Attention



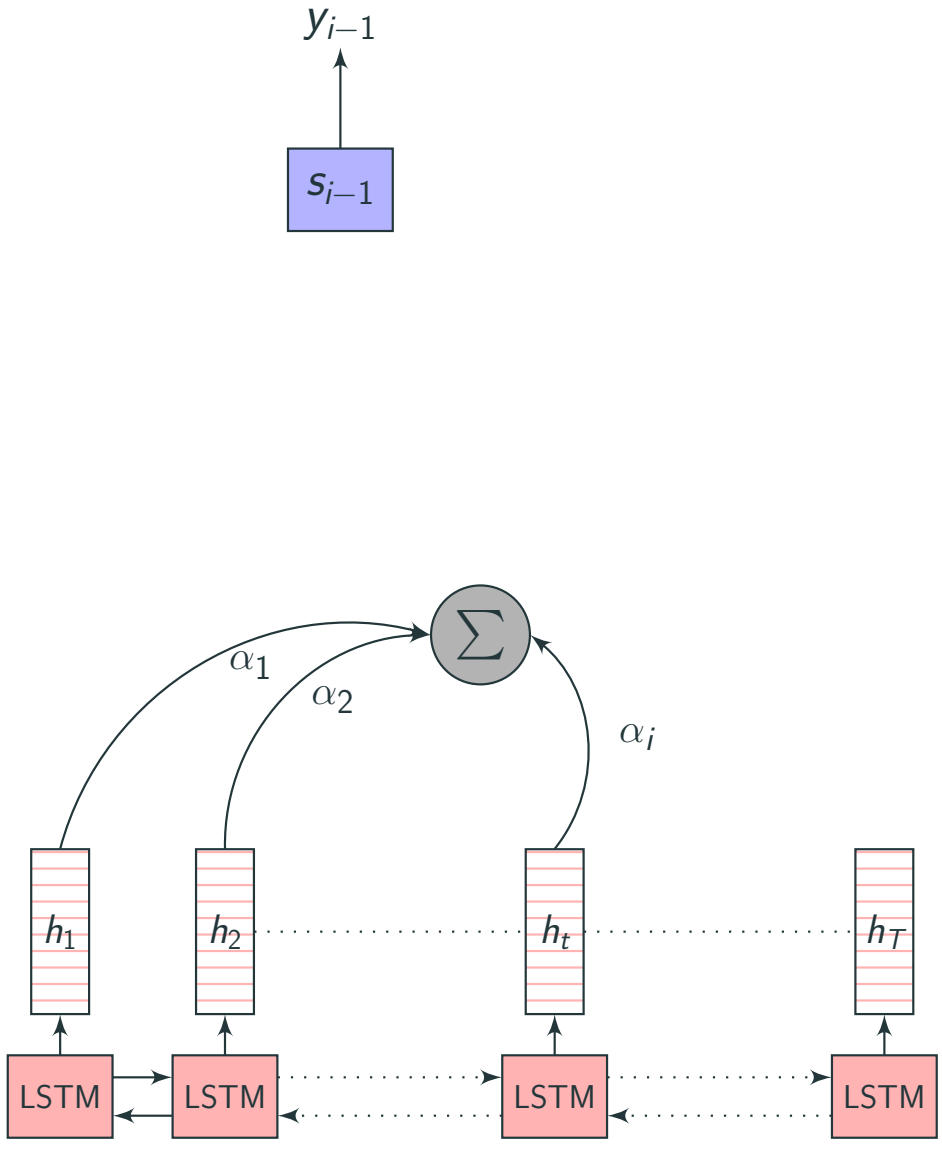
Introduction to Attention



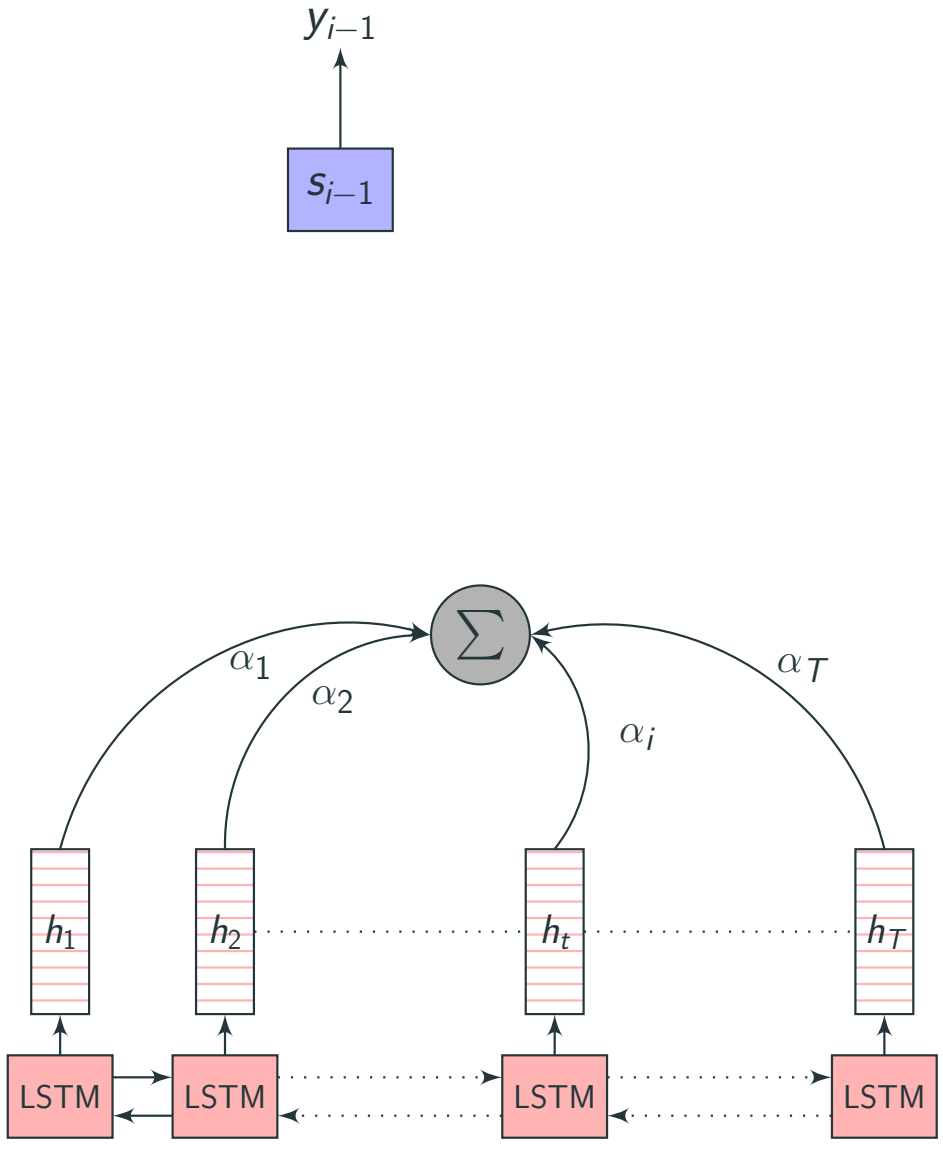
Introduction to Attention



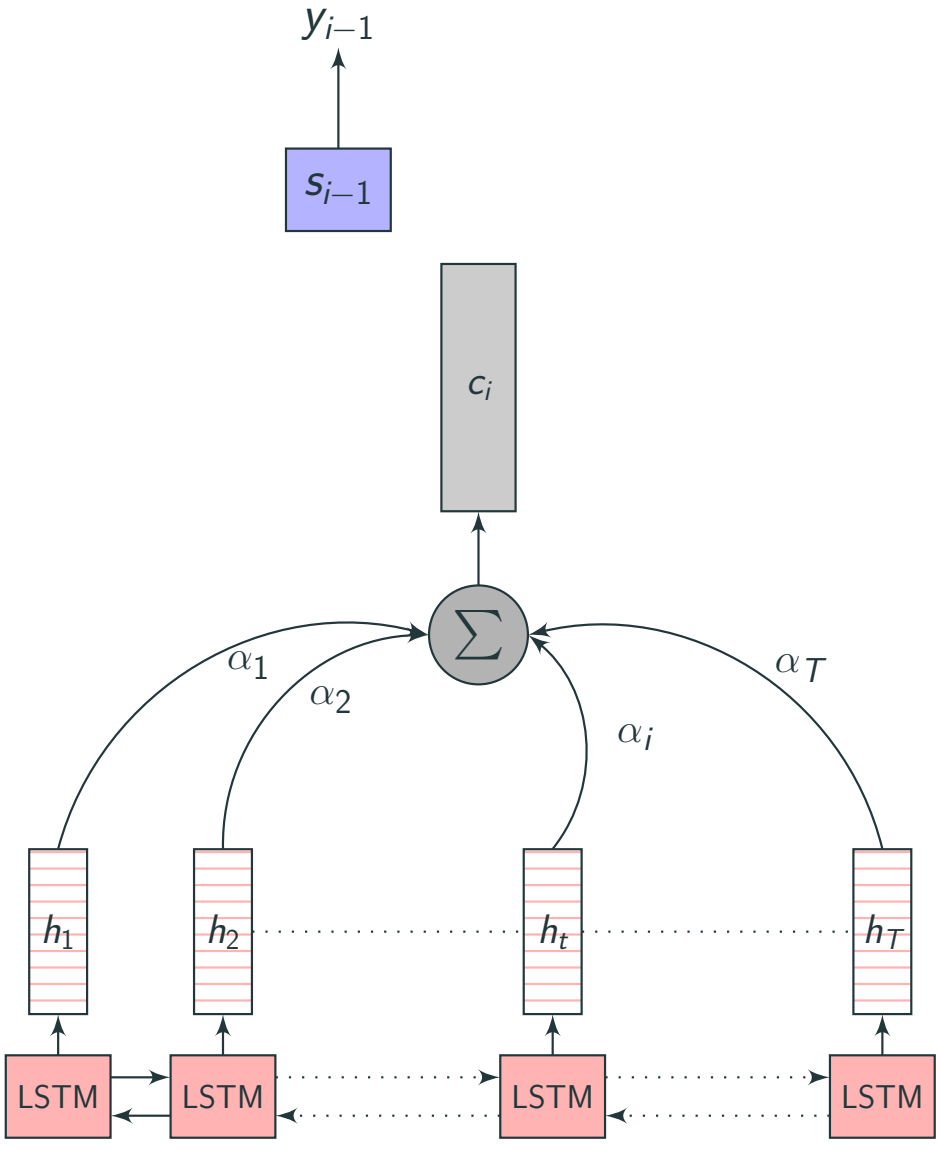
Introduction to Attention



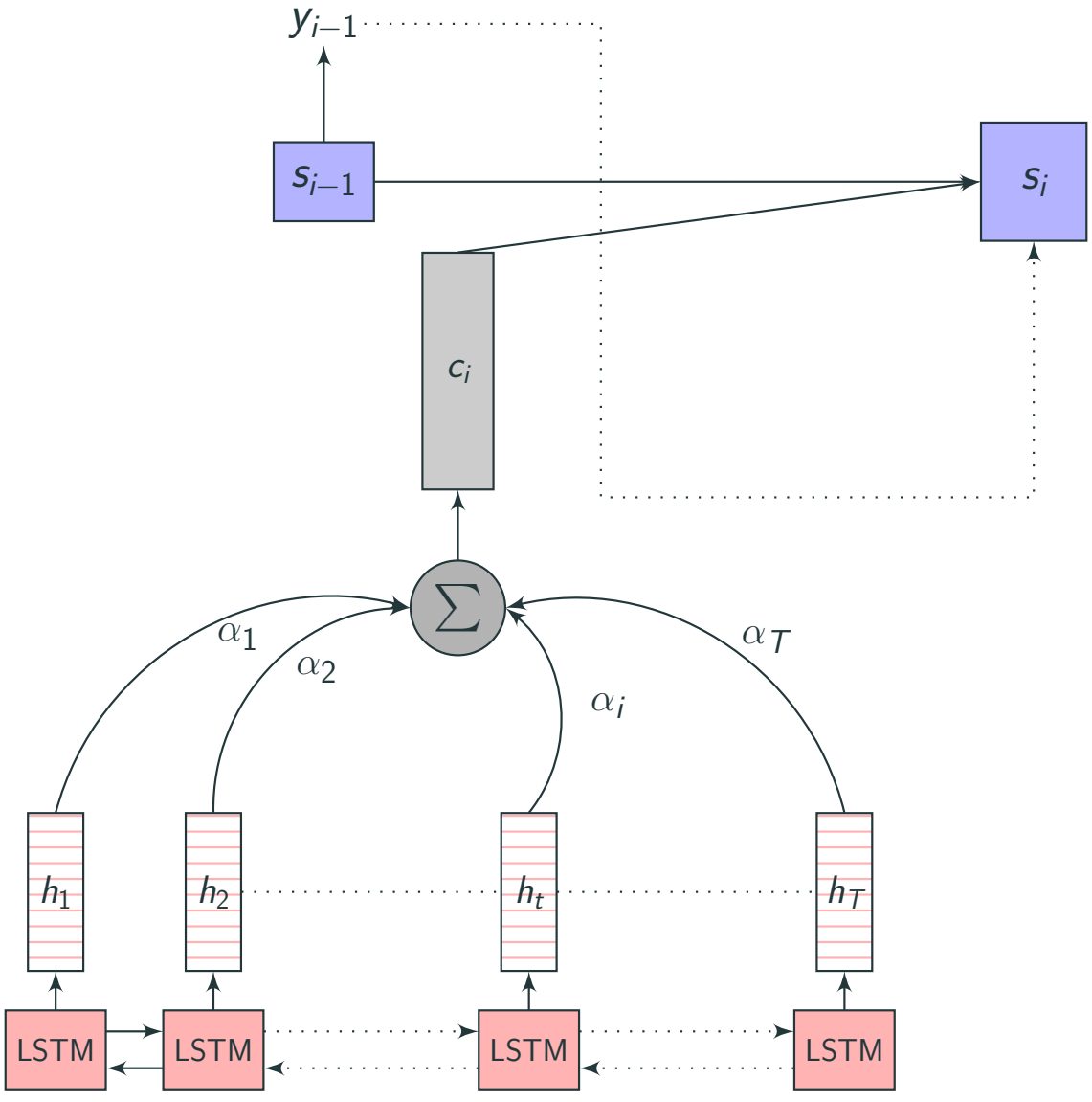
Introduction to Attention



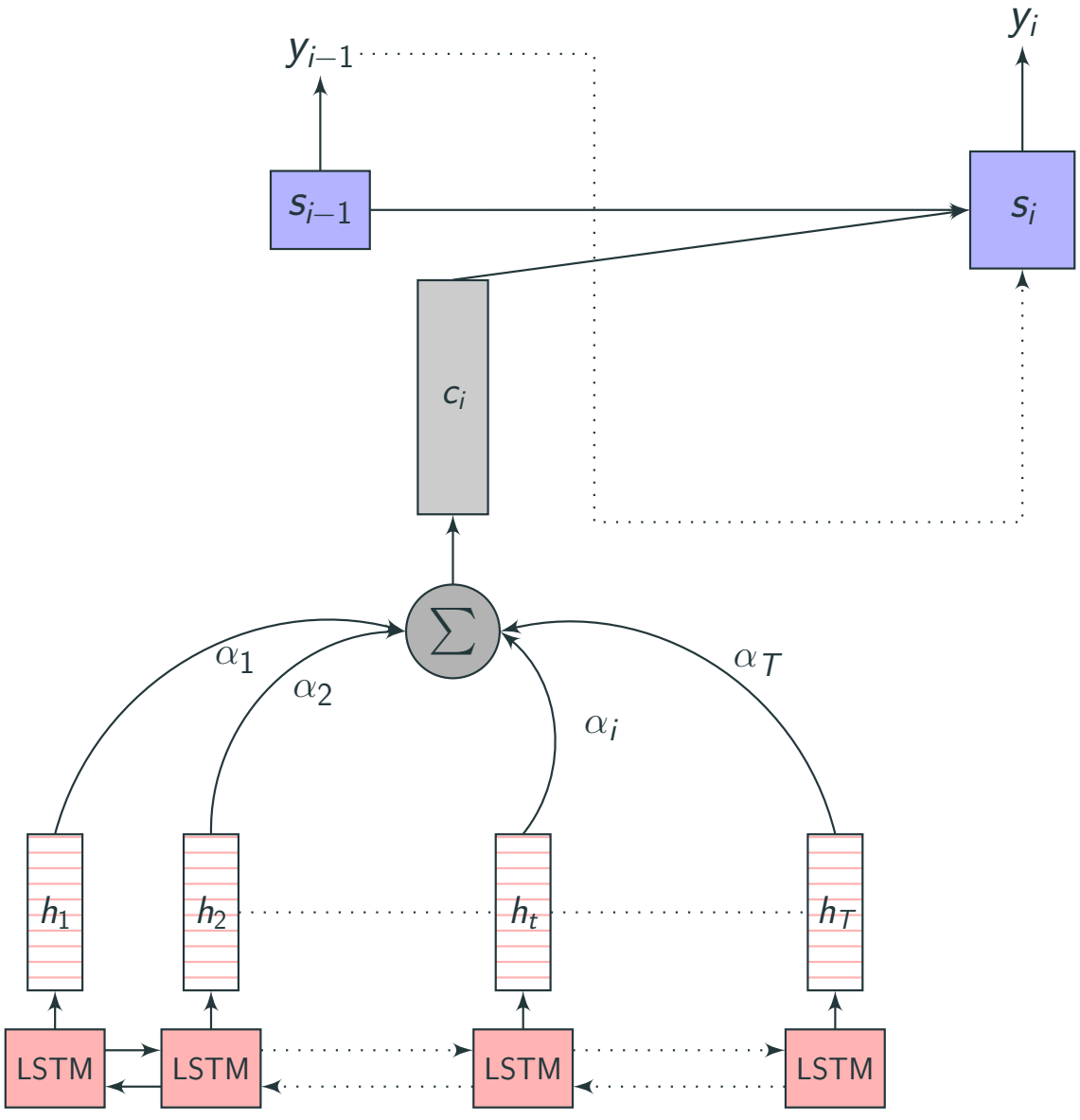
Introduction to Attention



Introduction to Attention



Introduction to Attention



Attention

Attention

An attention function can be described as mapping a query and a set of key-value pairs to an output. Each time the model needs to generate an output symbol, it (soft-) searches for a set of positions in the input feature vector sequence where the most relevant information is concentrated.

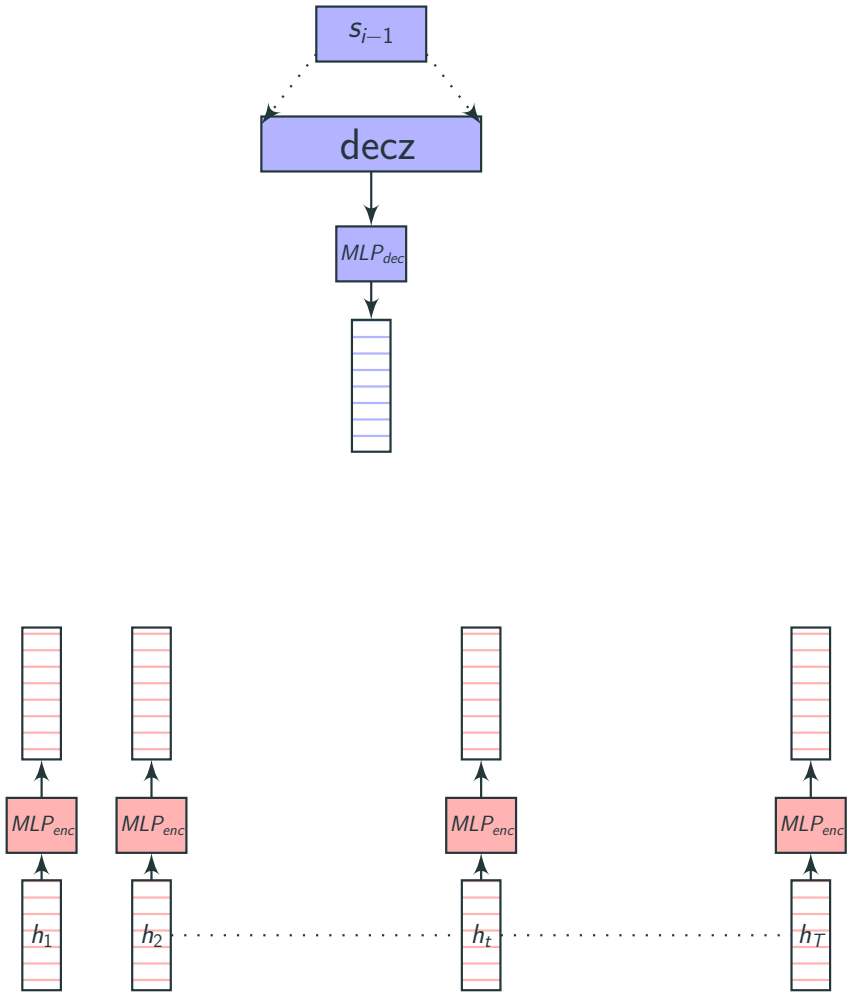
- The query, keys, values, and output are all vectors
- The output is computed as a weighted sum of the values, where the weight assigned to each value is computed by a compatibility function of the query with the corresponding key.
- Query : $dec_z \in \mathcal{R}^{d_q}$
- Key : $f(h) \in \mathcal{R}^{d_k}$
- Value : $g(h) \in \mathcal{R}^{d_v}$
- $Attention(Q, K, V) = Softmax(\frac{QK^T}{\sqrt{d_k}})V$

Attention plots.

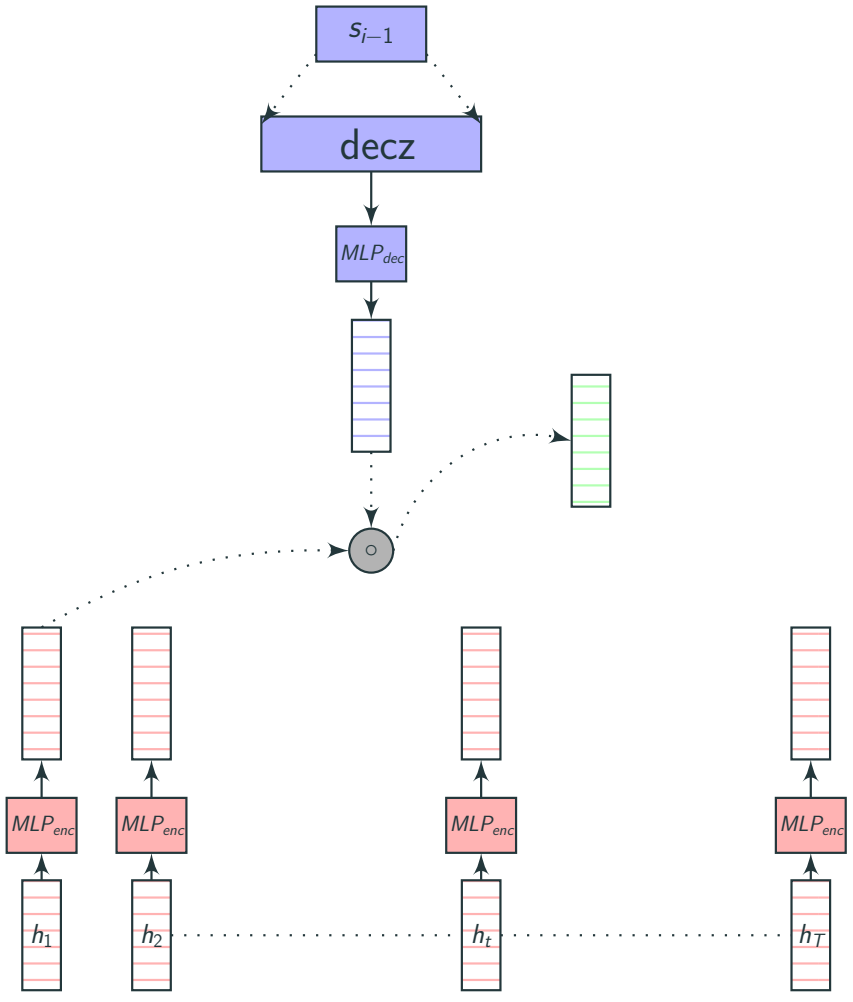
Attention

- $x = (x_1, x_2, ..., x_T)$ is the input sequence
- $y = (y_1, y_2, ..., y_U)$ is the target output sequence
- $h_t = f(x_t, h_{t-1})$ is the Encoder function
- $h = (h_1, h_2, ..., h_T)$ is the output of the Encoder
- $C_i = \sum_{j=1}^{T} \alpha_{i,j} \cdot h_j$ is the Context vector
- $\alpha_{i,j} = Softmax(e_{i,j}) = \frac{e^{e_{i,j}}}{\sum_{k=1}^{T} e^{e_{i,k}}}$ are the Attention weights
- $e_{i,j} = a(s_{i-1}, h_j)$ is the importance parameter for every encoded input
- $\sum_{j=1}^{T} e_{i,j} \neq 1$ the importance parameter need not sum to 1
- $\sum_{j=1}^{T} \alpha_{i,j} = 1$ the attention weights sum to 1
- $p(y_i|\{y_1, y_2, ..., y_{i-1}\}, C_i) = g(y_{i-1}, s_i, C_i)$ is the output symbol at the current time step
- $p(y) = \prod_{i=1}^{U} p(y_i|\{y_1, y_2, ..., y_{i-1}\}, C_i)$ is the probability of the entire output sequence

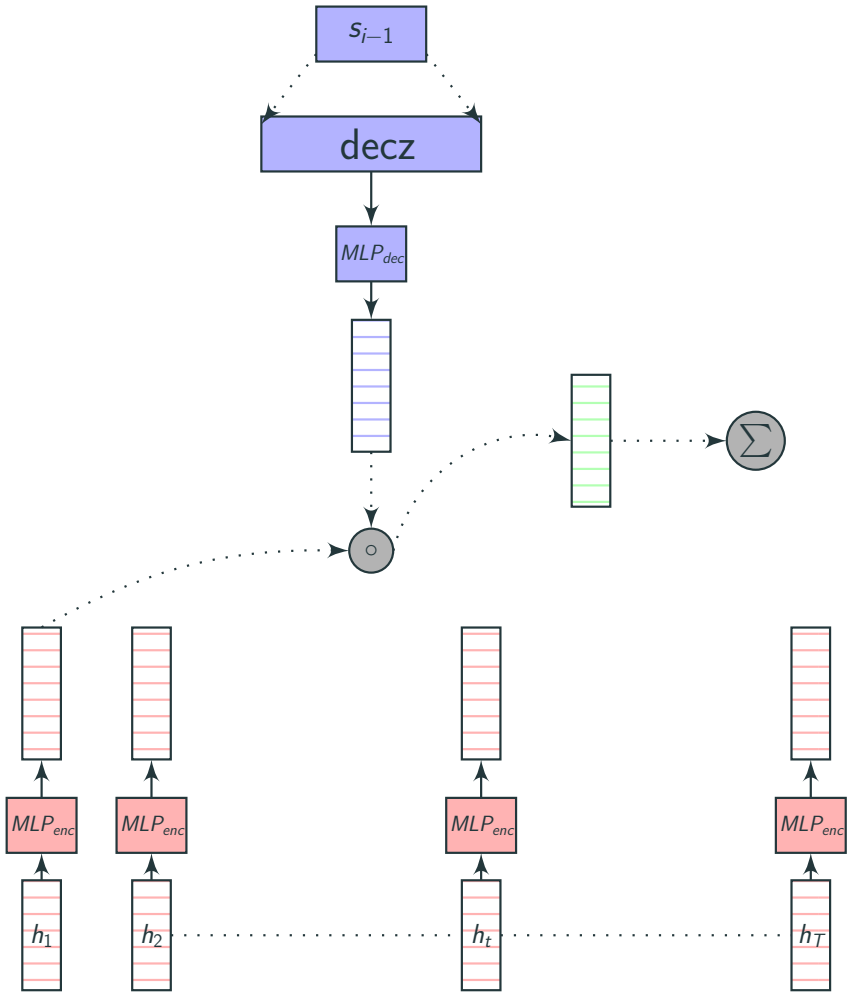
Dot product Attention



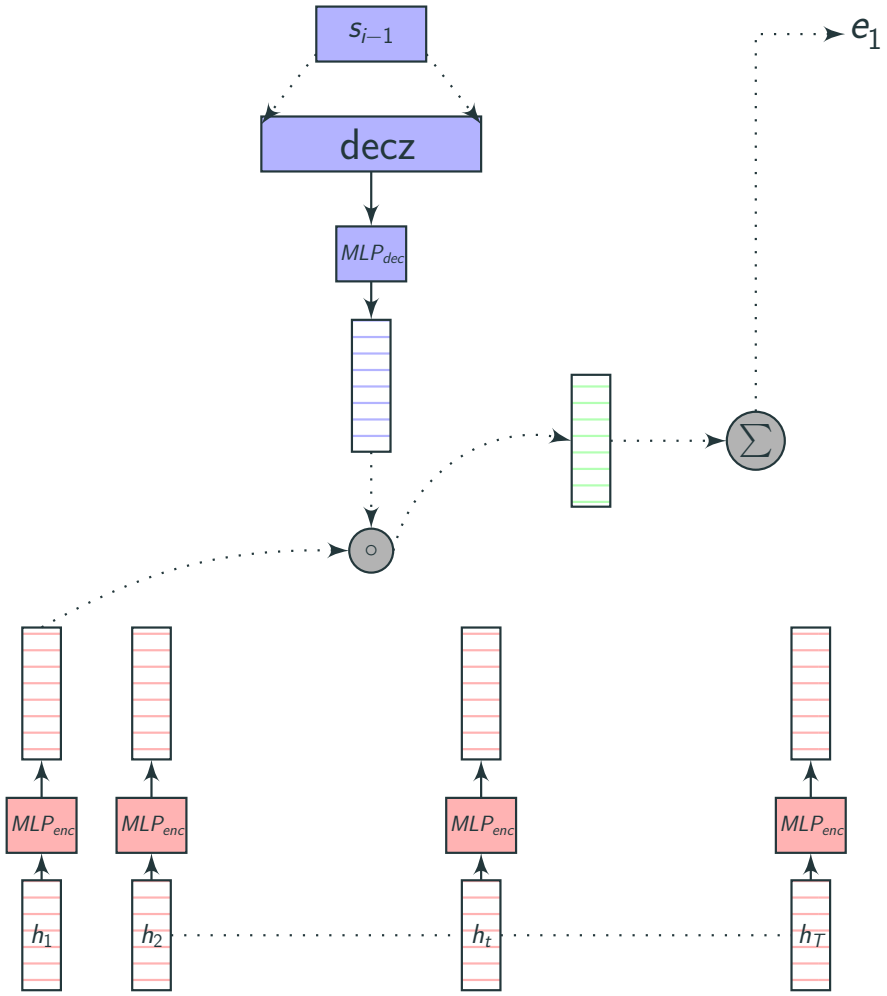
Dot product Attention



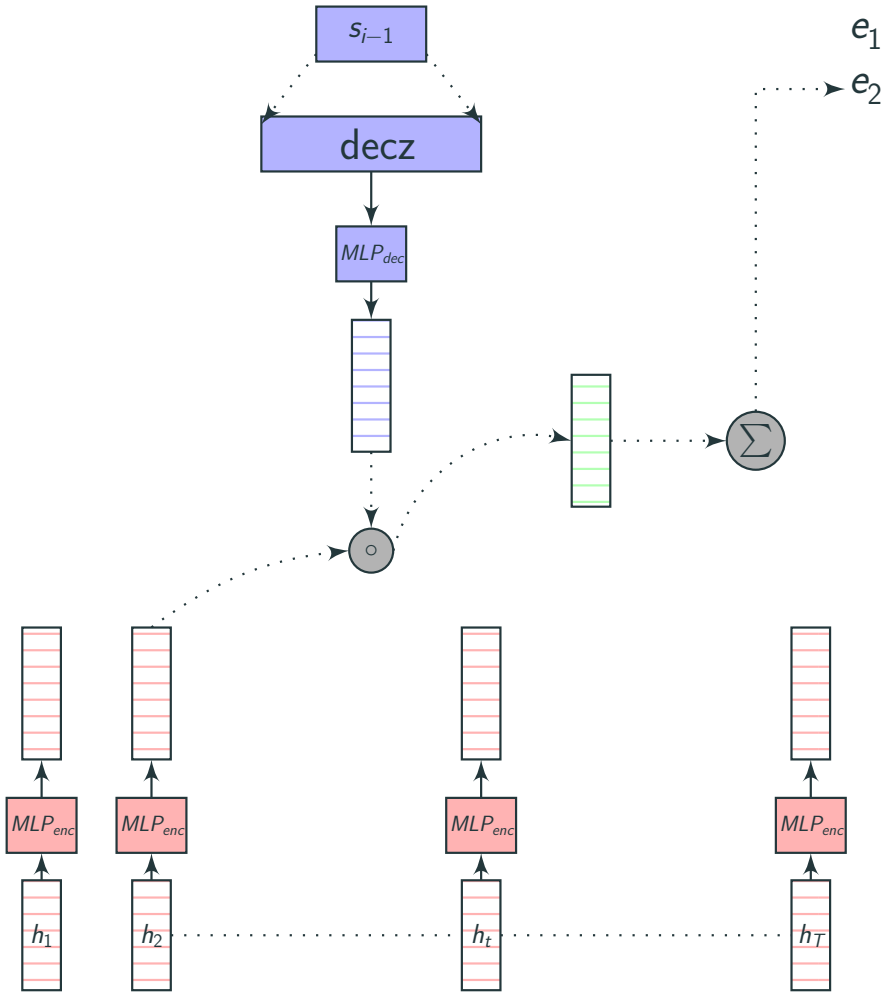
Dot product Attention



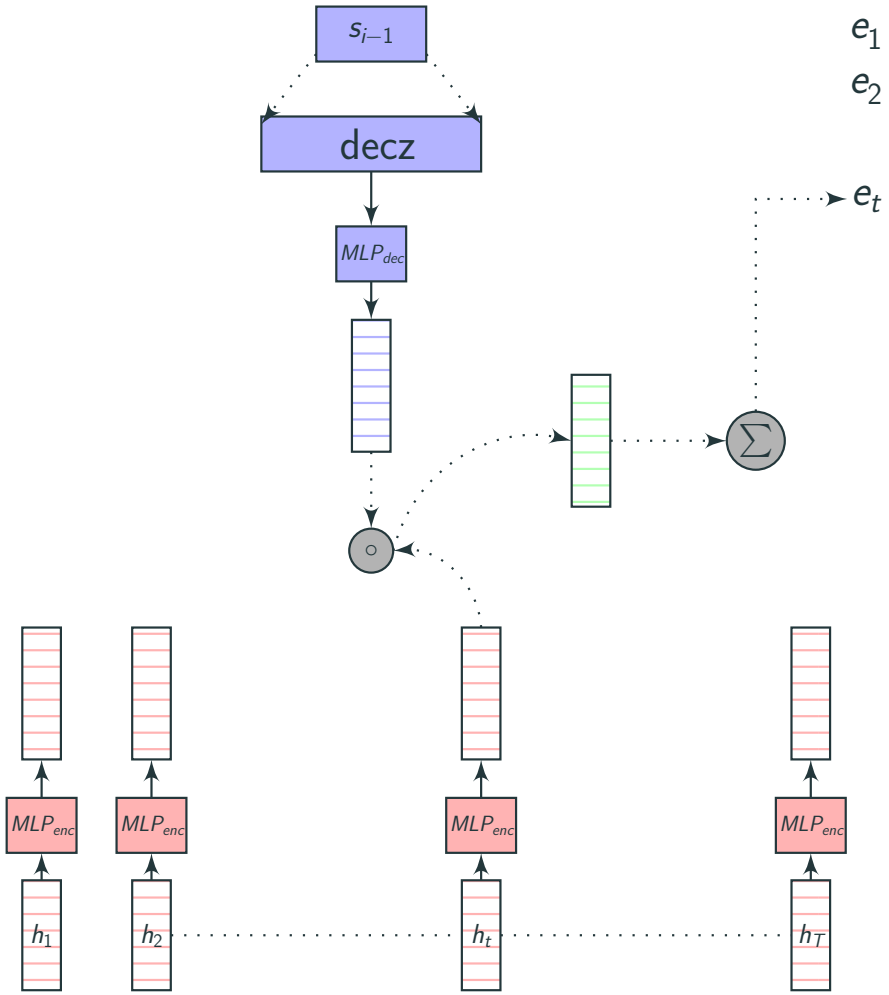
Dot product Attention



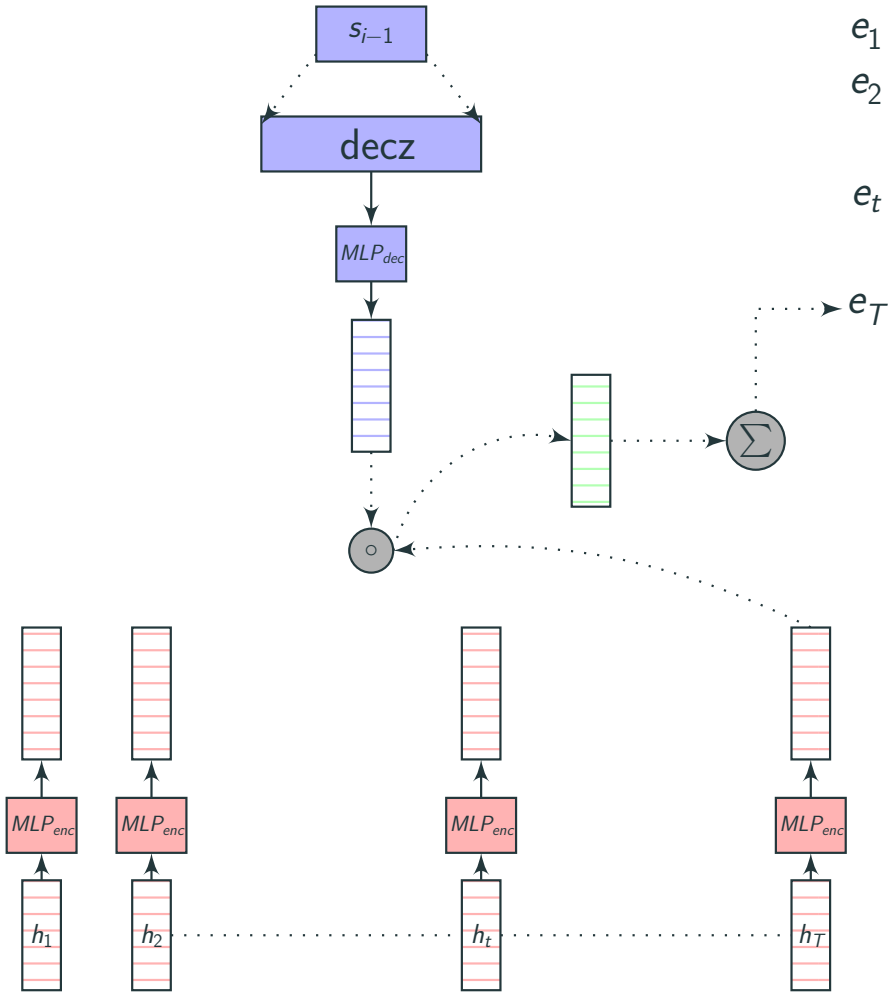
Dot product Attention



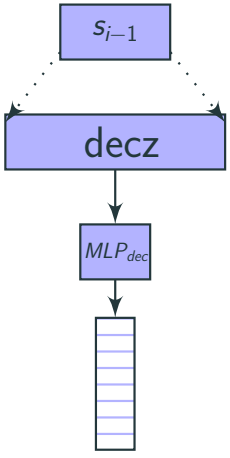
Dot product Attention

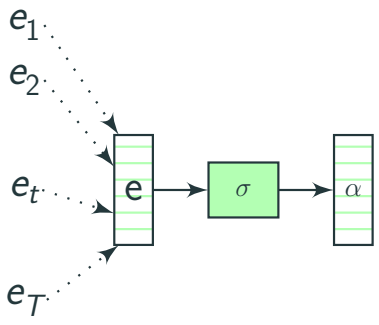


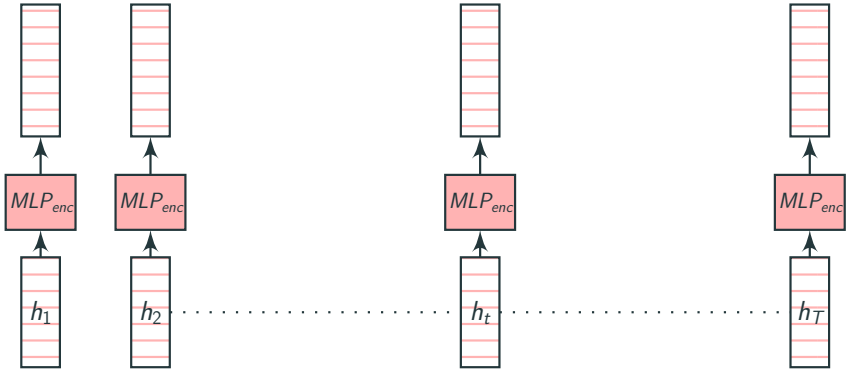
Dot product Attention



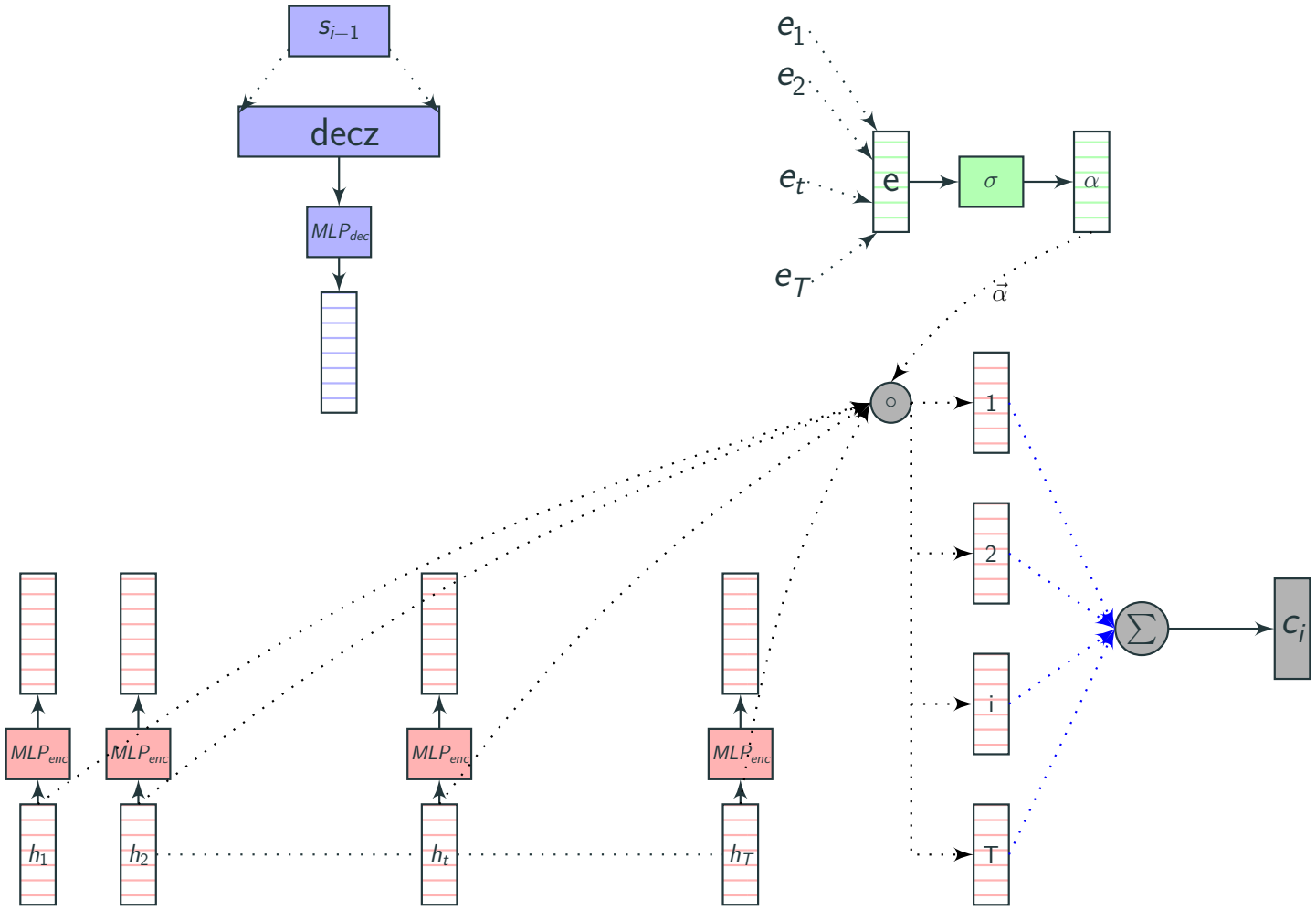
Dot product Attention



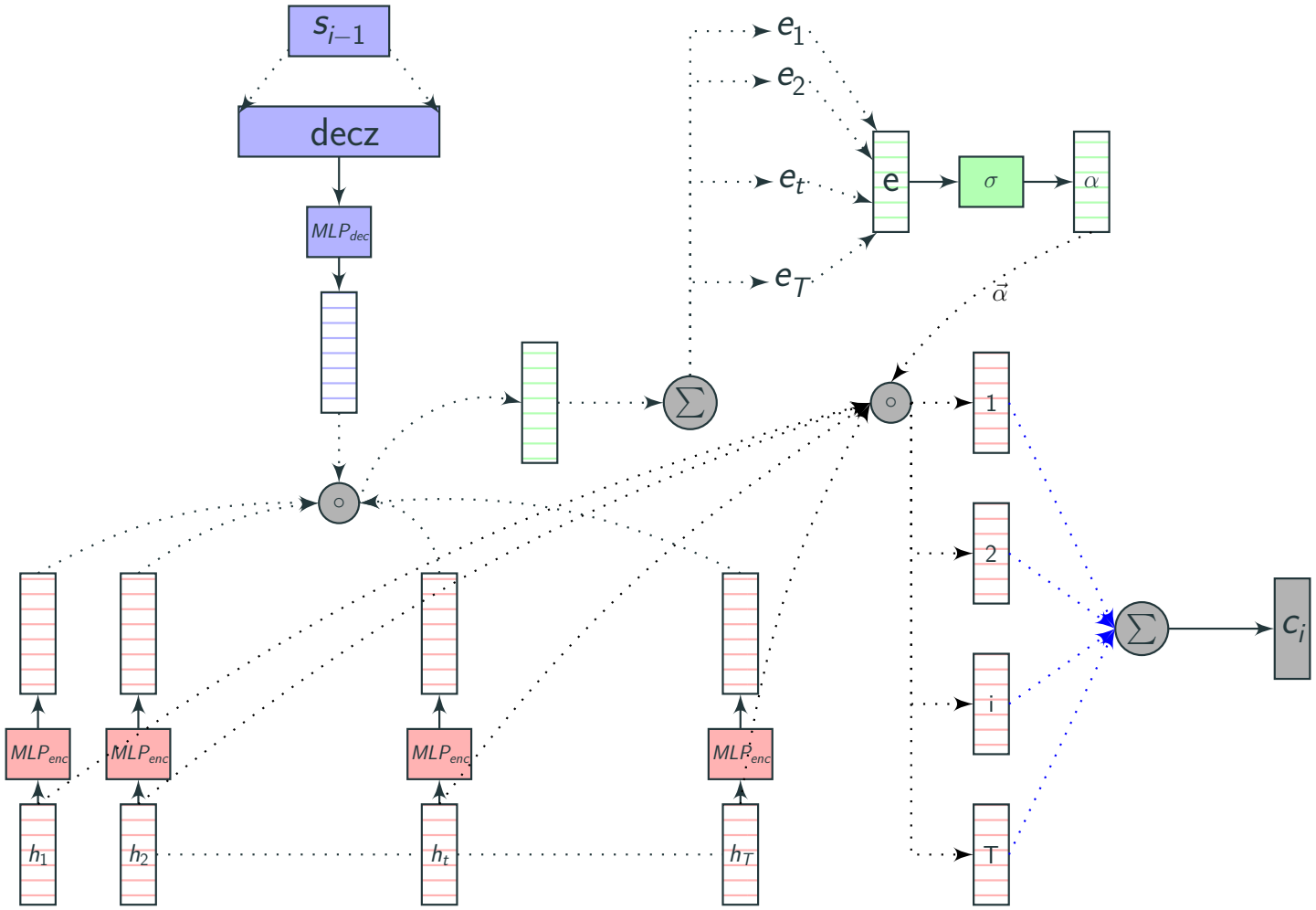




Dot product Attention



Dot product Attention



Types of attention

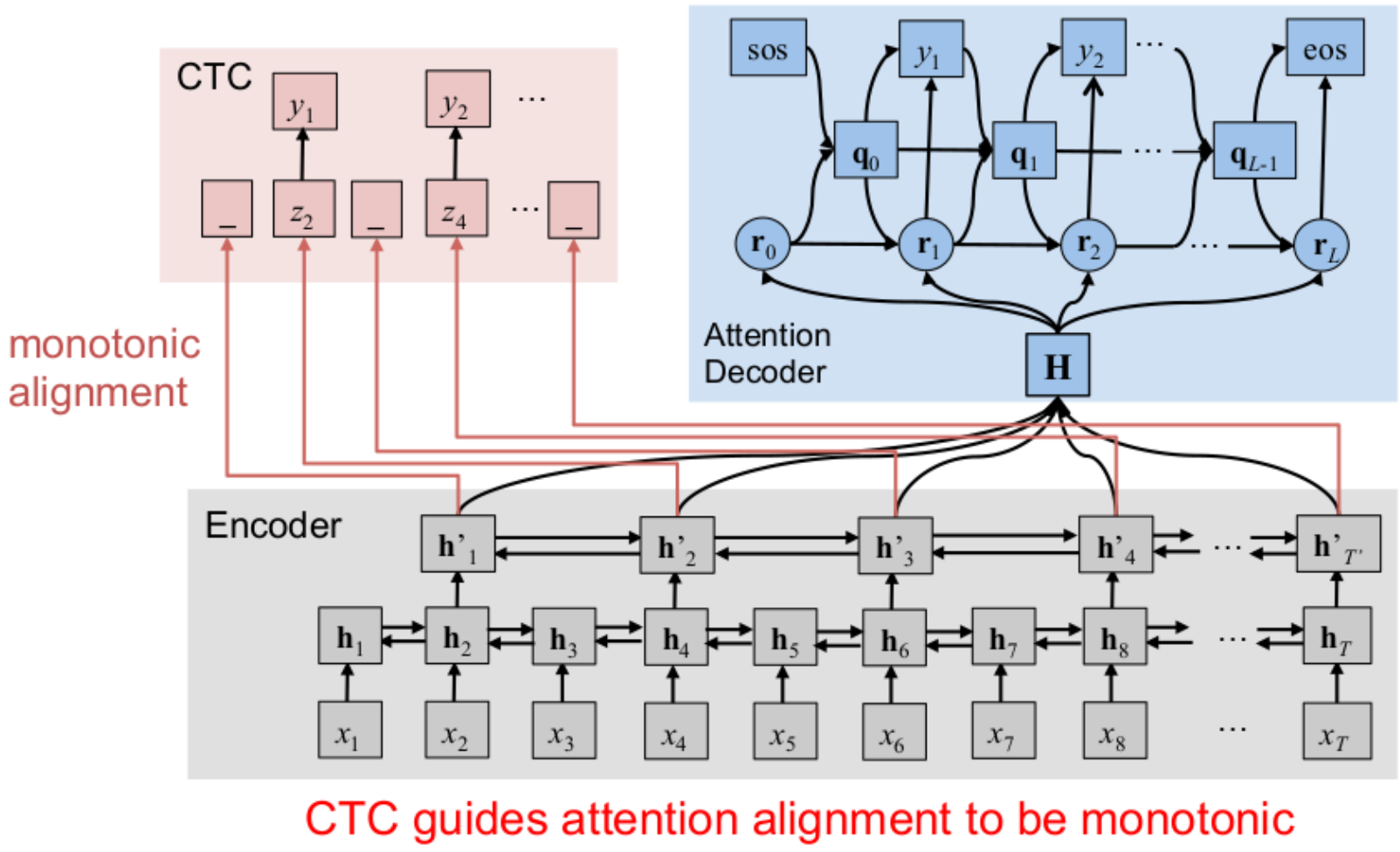
- Classified based on the **relevance/compatibility** function
- Content-based (Depends only on the contents of encoder/decoder states)
- Location-aware (Depends on previous attentions)
- Hybrid attention
- Multi-head attention
- Transformer architectures

Joint CTC/Attention architectures [Watanabe et al., 2017]

Joint CTC/Attention

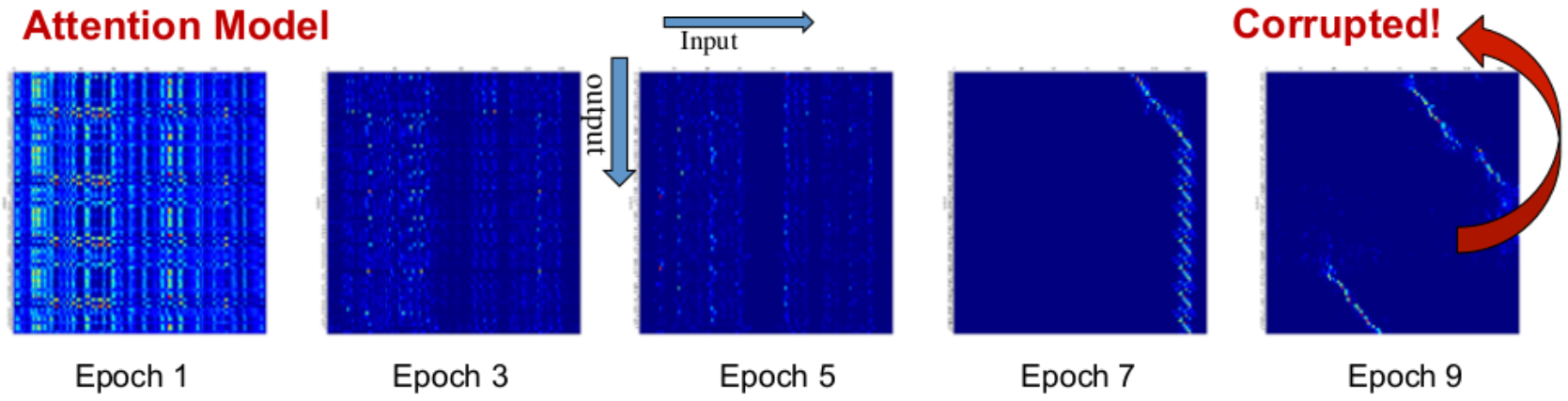
- $\mathcal{L}_{MOL} = \lambda \log\left(p_{ctc}(y|X)\right) + (1-\lambda)\log\left(p_{att}(y|X)\right)$

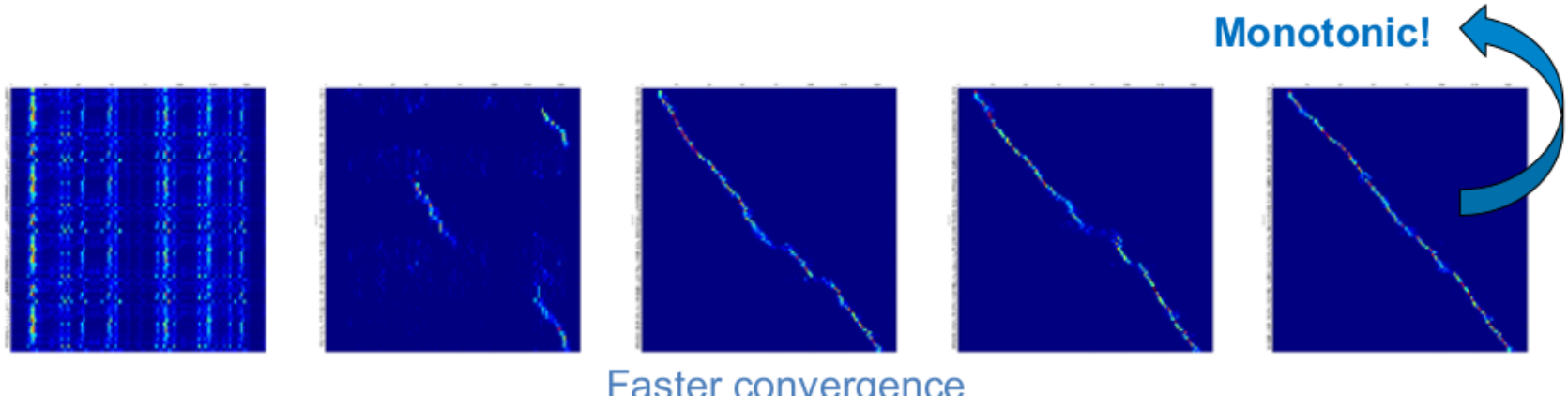
Multitask learning: $\mathcal{L}_{\mathrm{MTL}} = \lambda\mathcal{L}_{\mathrm{CTC}} + (1-\lambda)\mathcal{L}_{\mathrm{Attention}}$



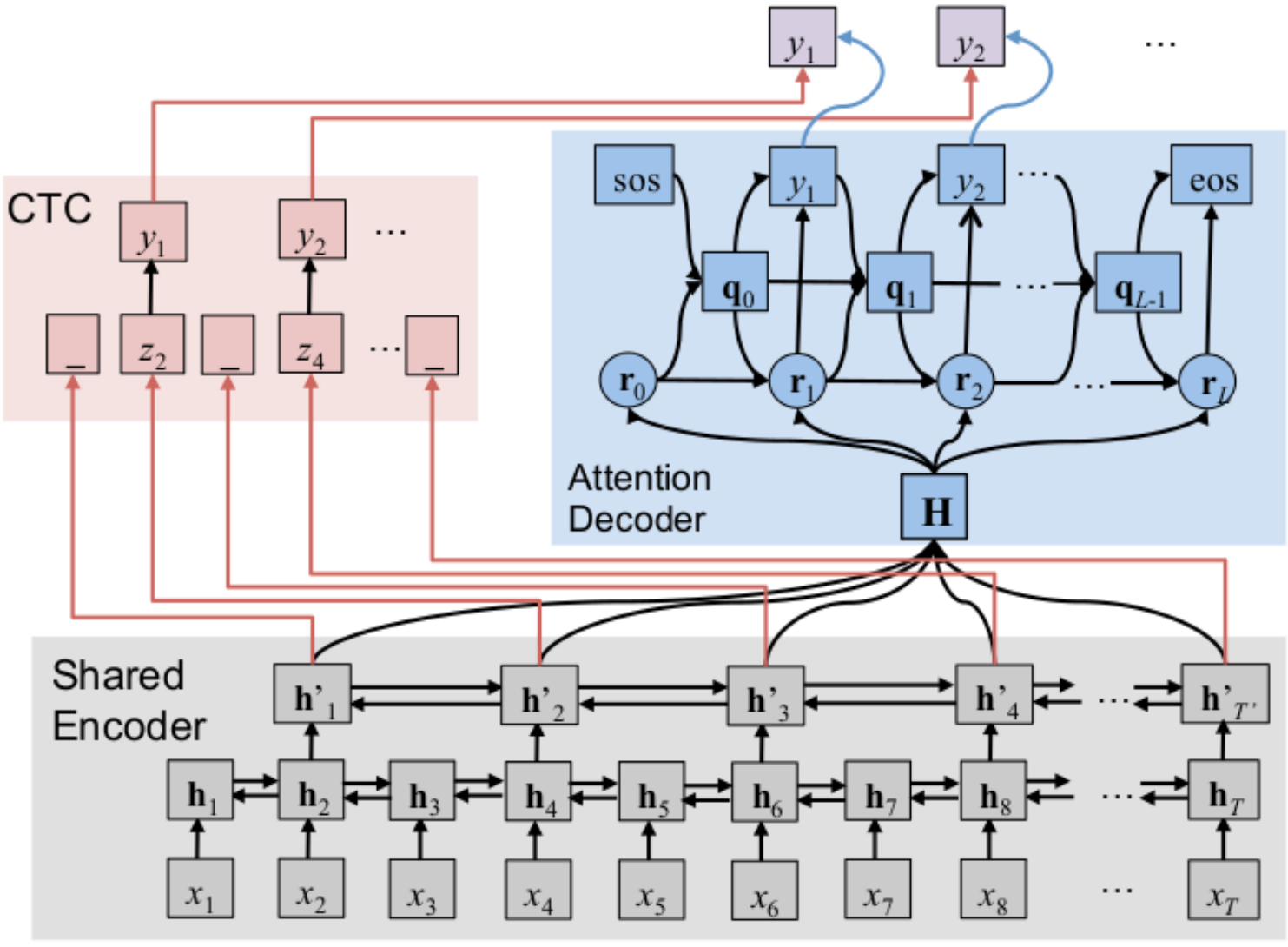
CTC guides attention alignment to be monotonic

Joint CTC/Attention - Monotonic alignments



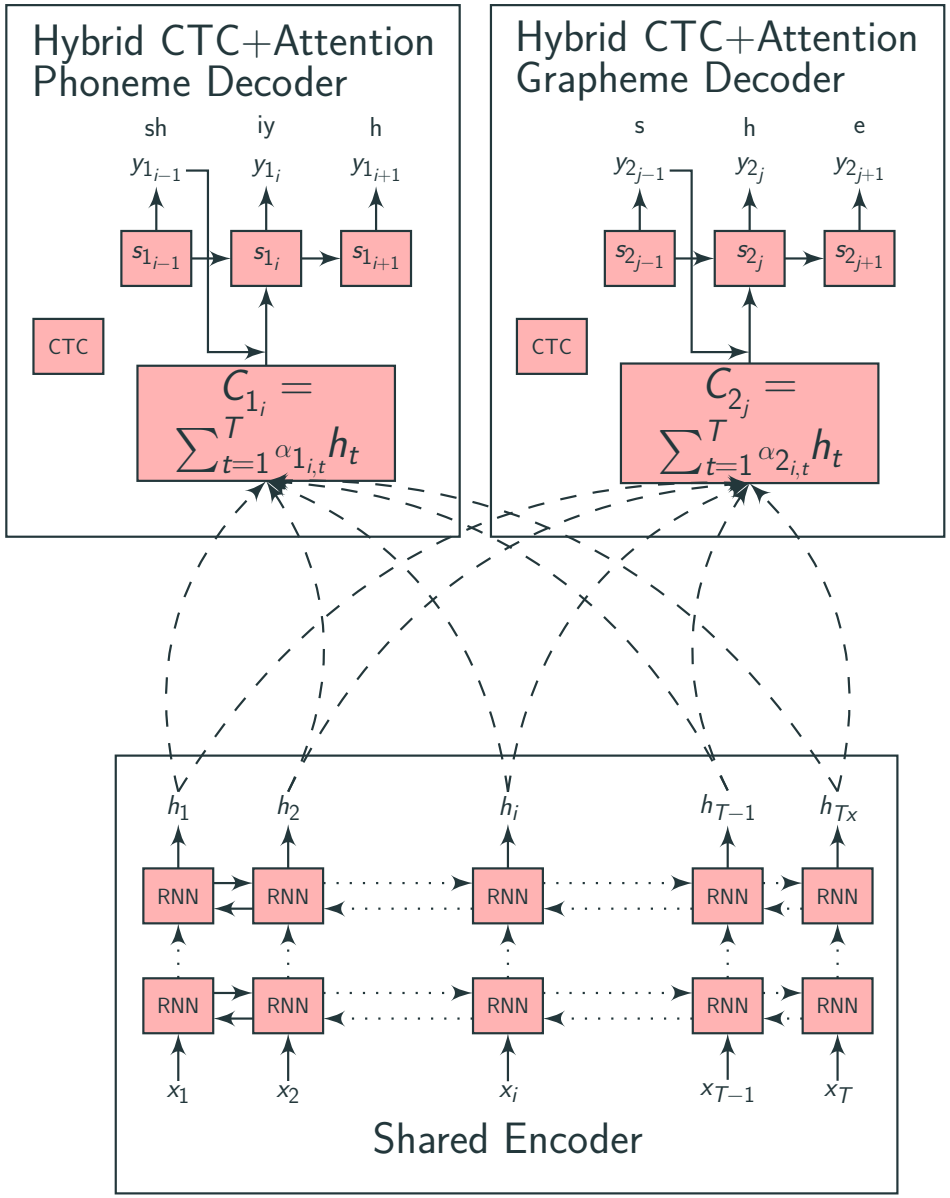


Joint CTC/Attention - Decoding



Our Work - Proposed multi-target architecture [Under review at ASRU 2019]

Architecture



Results

Joint attention on phoneme-grapheme

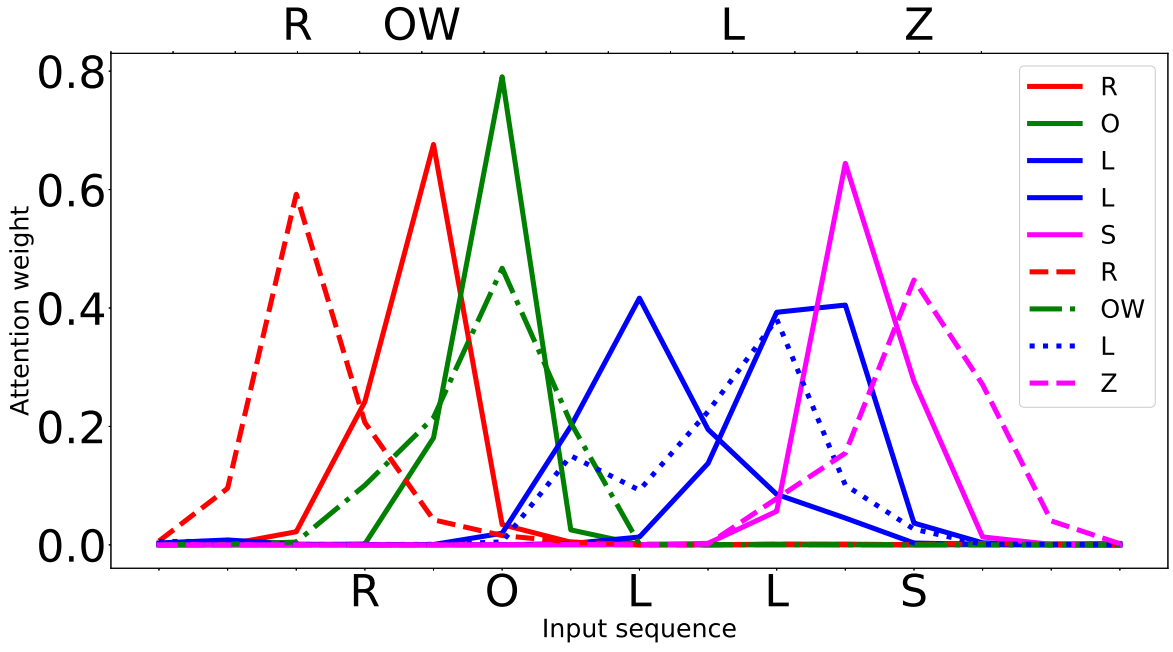


Figure 1: Attention weights for predicting phonemes (*r ow l z*) and graphemes (r o l l s) of the word "rolls"

Attention profile on phoneme-grapheme

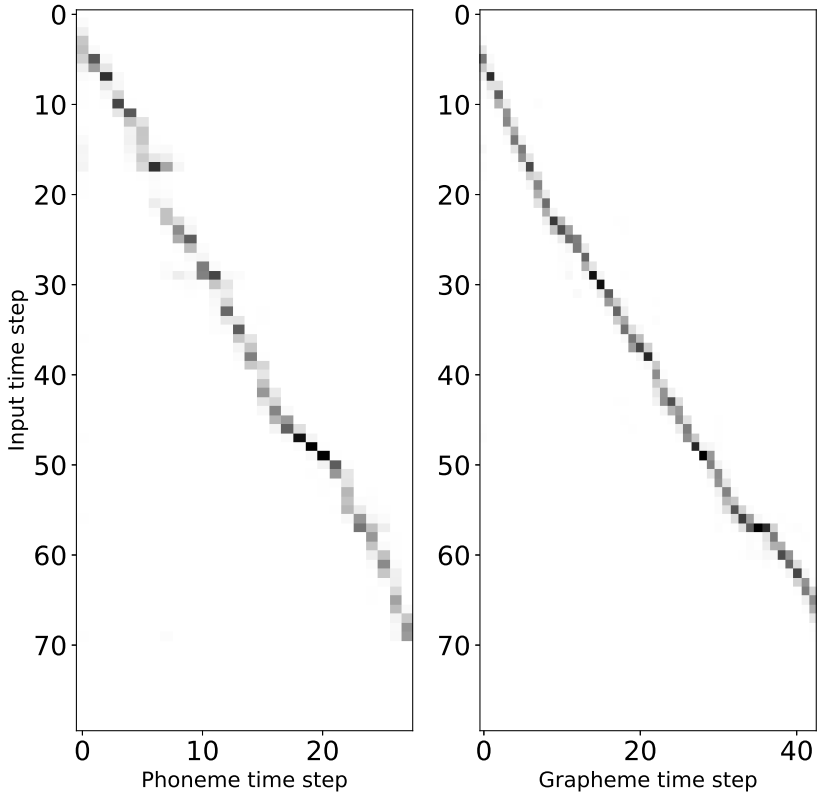


Figure 2: Attention profile for predicting phoneme and grapheme sequences for the example TIMIT utterance **FADG0_SI1909**

Results on Librispeech train_clean_100 and test_clean

Target	PER	CER
phoneme	7	-
grapheme	-	7.7
phoneme @ 5L, grapheme @ 5L	7.4	8.1
phoneme @ 3L, grapheme @ 5L	7.2	7.8
phoneme @ 1L, grapheme @ 5L	**6.8**	**7.1**

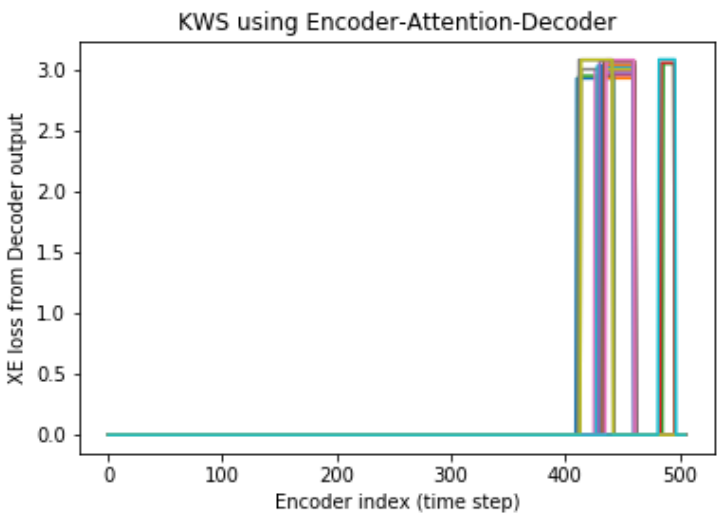
Future Work

Multi-Head decoder with ASWU targets

- Introduce automatically derived acoustic sub-word units
- Performance with multi-target decoder
- Hierarchical target learning.
  ASWU->phonemes->characters->(word-pieces?)

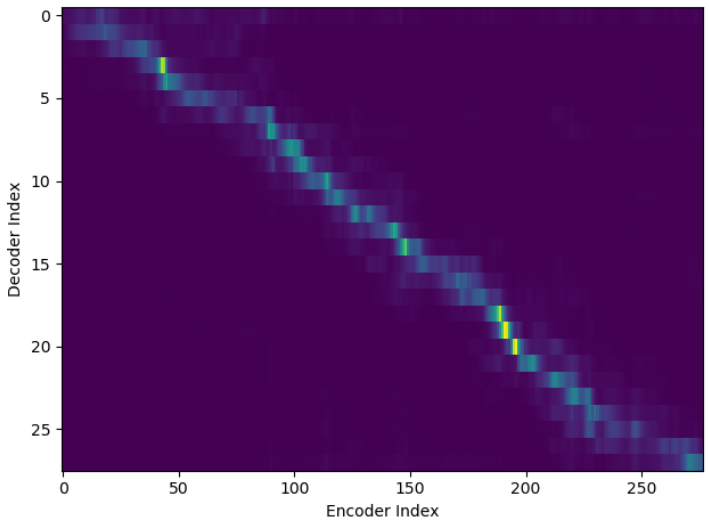
Attention as alignment - Attention-assisted Forced Alignment

- Align features based on attention distribution
- Attention is too sensitive for transcript. (Need some slack?)
- Confidence measure for Forced Alignment
- Keyword spotting with this framework. (Sliding window method)
- Transcript : "<space> w iy <space> d ih <space> n aa dx ah <space> k s eh <space> dh ah <space> d ay <space> n aa s ih s ae w ah n s <space> <space> b ah <space> g r ae sh ah l iy <space> w iy aa <space> **k ah m ih ng** <space> t uw <space>"



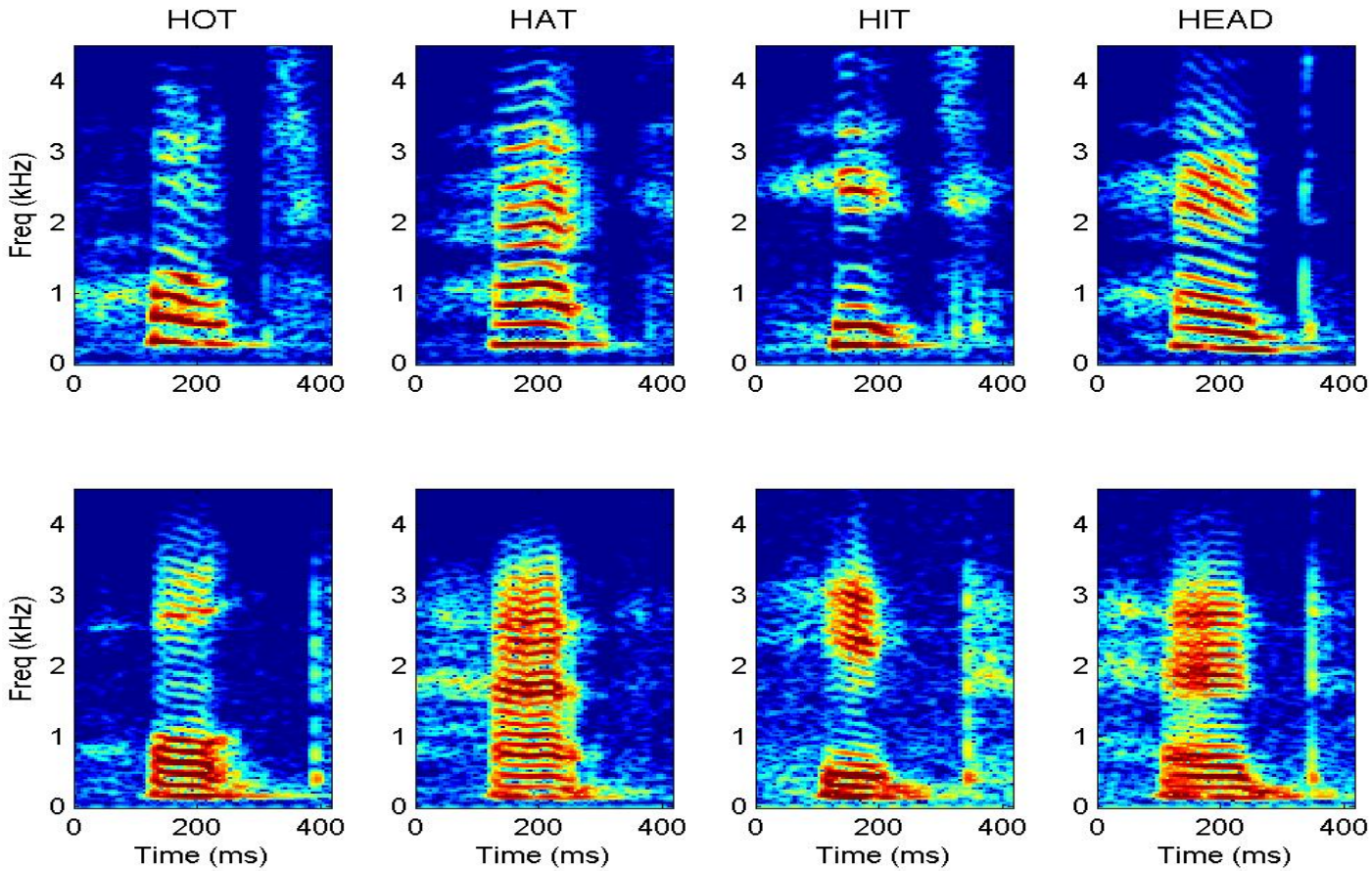
ORACLE Attention - Ekam Sat Vipra Bahudha Vadanti?

- Transcript is always monotonically aligned to speech
- There are 13 variants of Attention mechanisms widely used.
- Is there a "one true" attention (alignment)?
- ORACLE as an initial estimate (from GMM-HMM)
- Effects on convergence
- Early results using Wasserstein metric



Interpretability and Explainability

- Track attention to formant movement?



Analysis

- When the model is making a mistake, it it looking at the wrong location in the signal?
- Could we decouple the location part of the network from the content part? (Helps in transfer learning)
- The Context vector C - how is it distributed after every epoch? Could we see them getting separated?
- Location-aware attention always starts from left (strict). Content-based attention can span over entire signal. Use this for faster kws models.

Suggestions

Suggestions please.