

ASR

Automatic Speech Recognition Part 2

ASR

- Encoder-Decoder
- RNN-T

Recap

- ASR metric - Word Error Rate
- New sequence loss - CTC loss
- Produce conditionally independent outputs
- Use Phonemes, Letters, BPE
- Fastest approach to ASR: parallel forward pass
- Use greedy decoding to get fastest results
- Beam search yields better quality
- Use NGram language model to introduce new information
- Use smaller models in beam search to get N-best
- Rescore N-best using big models

Alignment

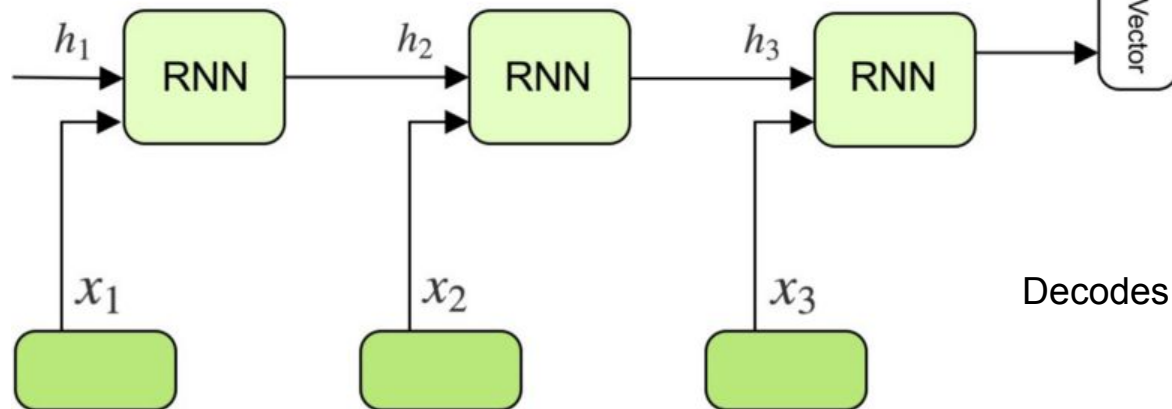
- Speech recognition alignment is much harder problem due to continuous input
- Translation problem is word-to-word task. Both source (SRC) and destination (DST) are discrete input values
- Speech recognition input is continuous
- CTC learns sequence alignment via CTC loss with the help of blanks and repeatable characters



Encoder-Decoder architecture

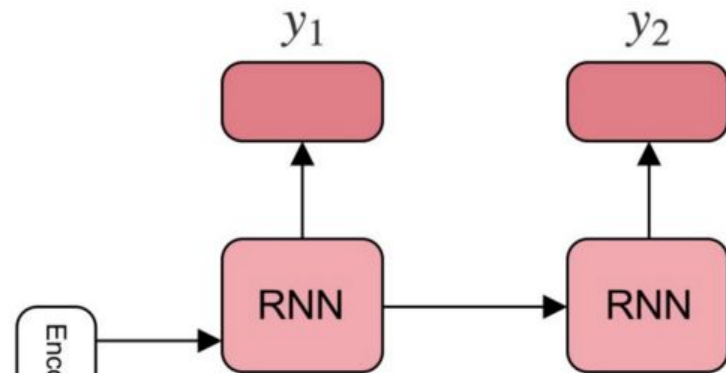
Encoder

Scans input and creates representation

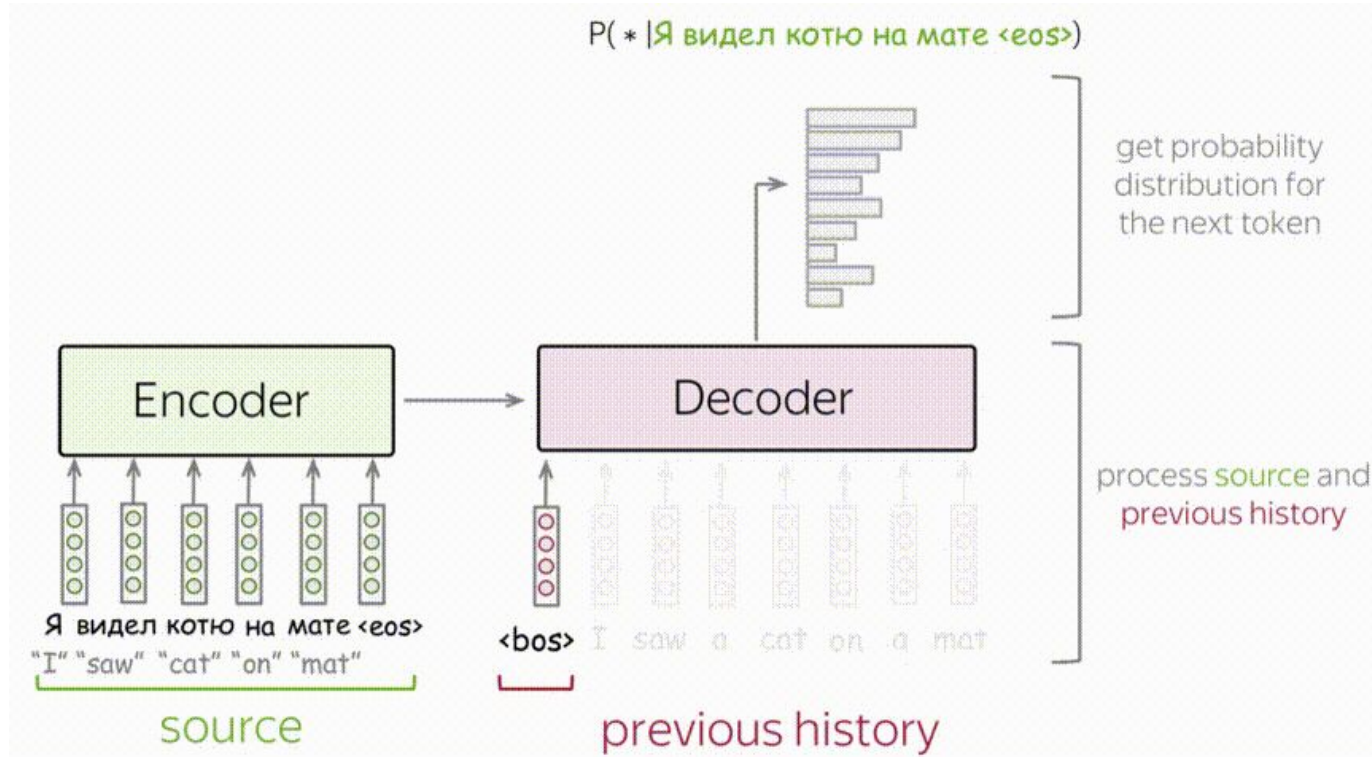


Decoder

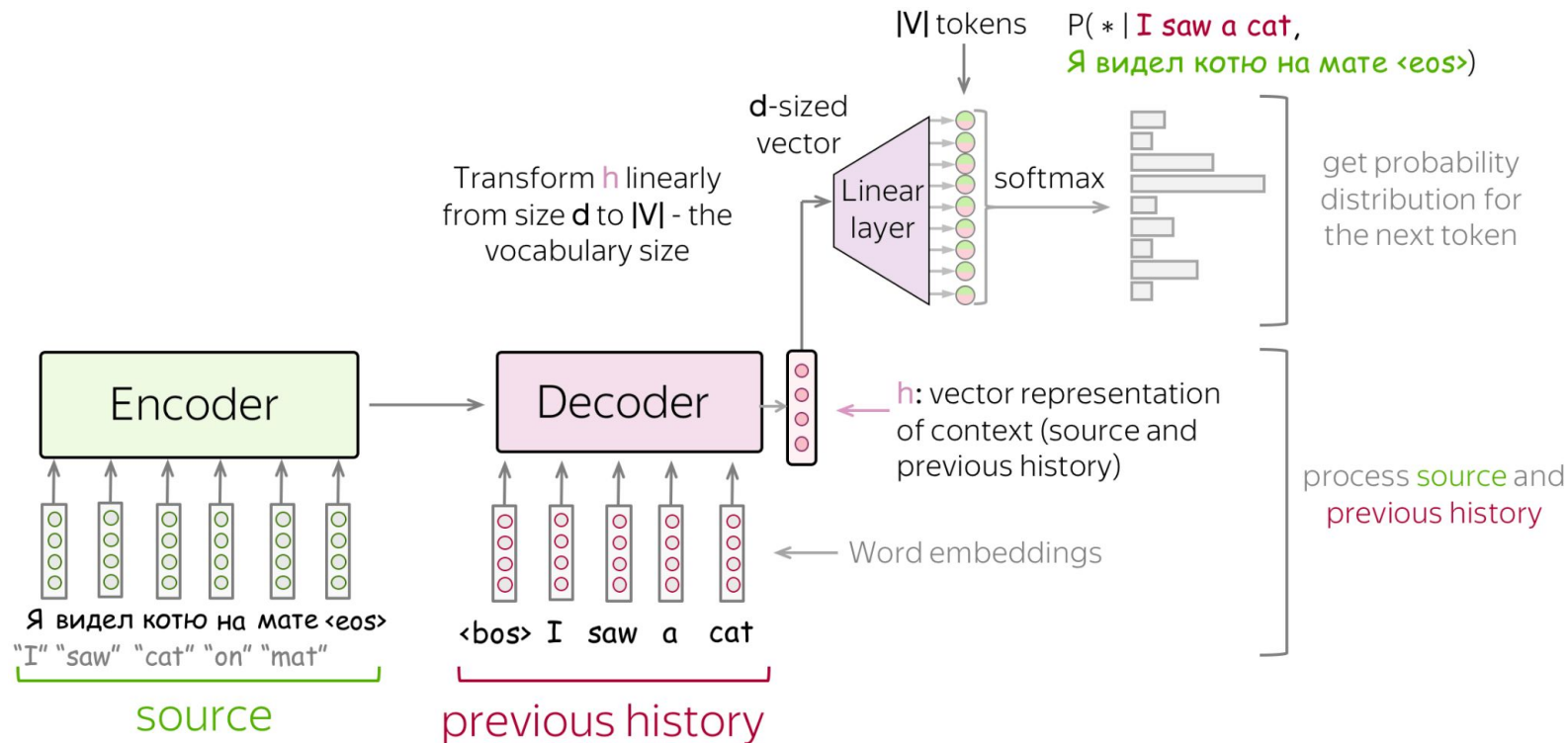
Decodes representation, generate target sequence



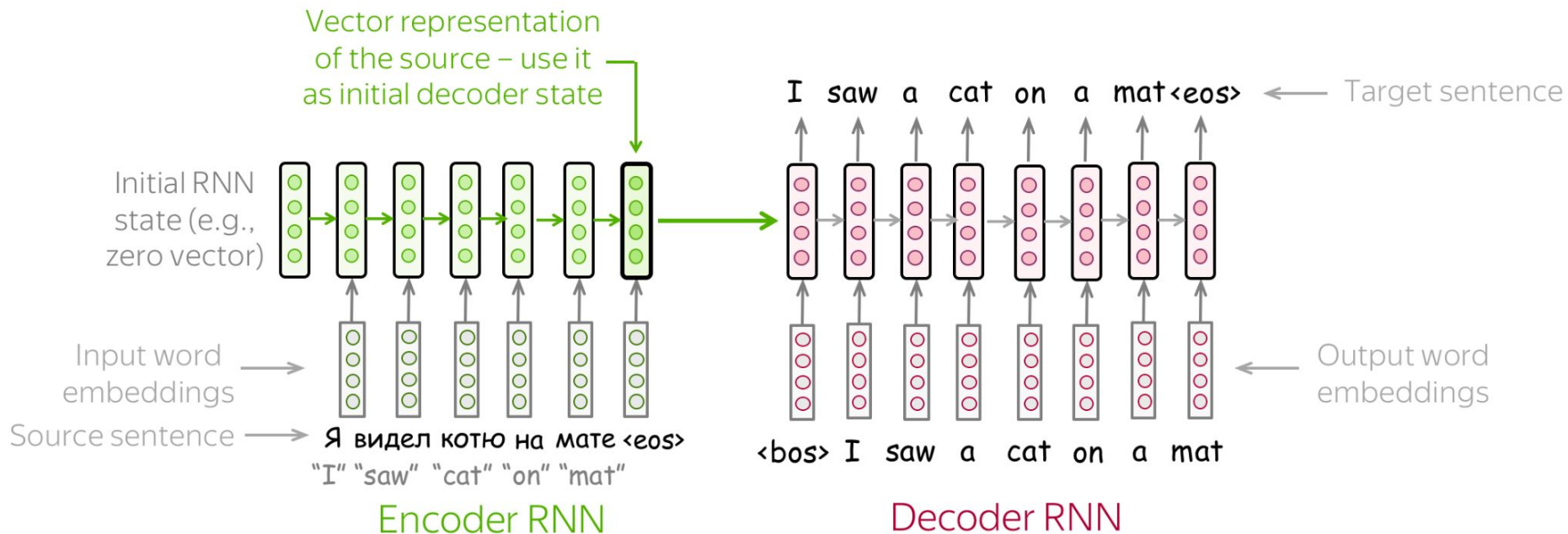
Encoder-Decoder architecture



Encoder-Decoder architecture



Encoder-Decoder architecture



Encoder-Decoder architecture

Source sequence:

Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"

Target sequence:

I saw a cat on a mat <eos>
previous tokens we want the model to predict this

← one training example

← one step for this example

Model prediction: $p(* | \text{I saw a, Я ... <eos>})$

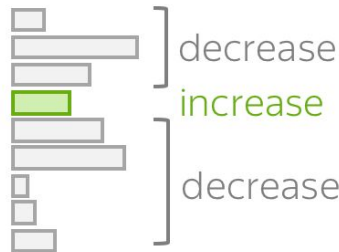


← cat →

Target



Loss = $-\log(p(\text{cat})) \rightarrow \min$



Encoder-Decoder architecture

Source sequence:

Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"

Target sequence:

I saw a cat on a mat <eos>
previous tokens we want the model to predict this

← one training example

← one step for this example

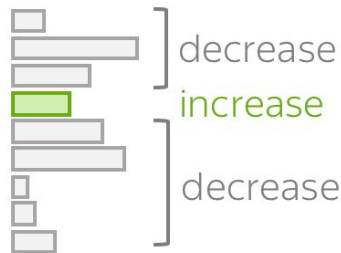
Model prediction: $p(* | \text{I saw a, Я ... <eos>})$



Target



Loss = $-\log(p(\text{cat})) \rightarrow \min$



Encoder-Decoder architecture

Encoder: read source



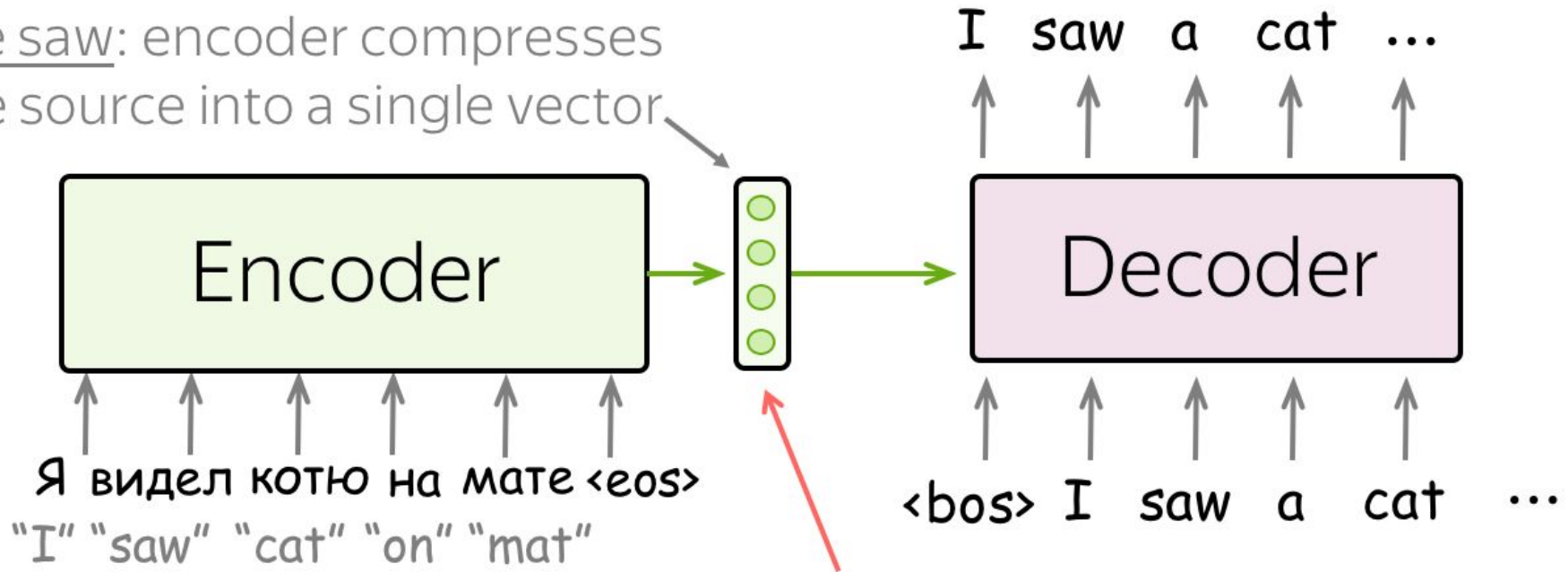
we are here

Source: Я видел котю на мате <eos>
"I" "saw" "cat" "on" "mat"

Target: I saw a cat on a mat <eos>

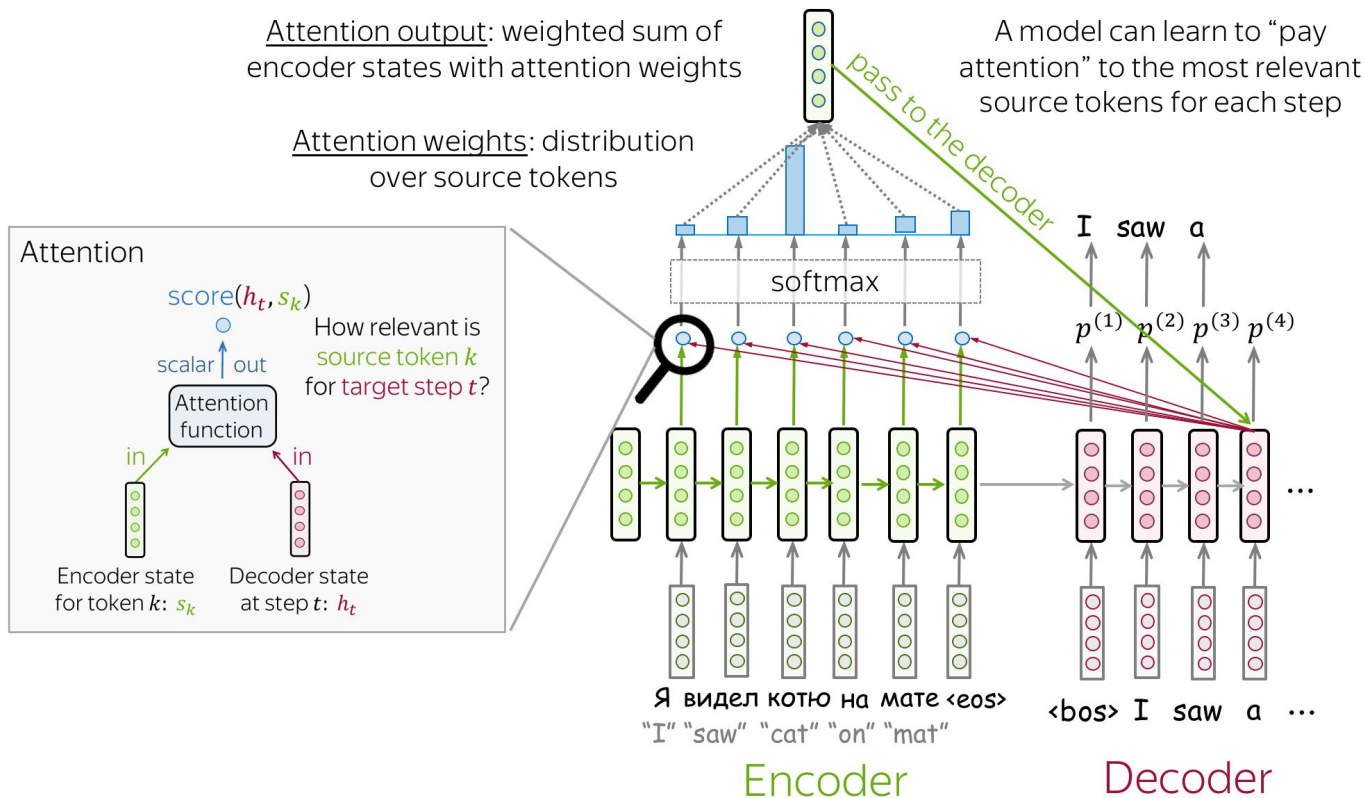
Encoder-Decoder architecture

We saw: encoder compresses the source into a single vector

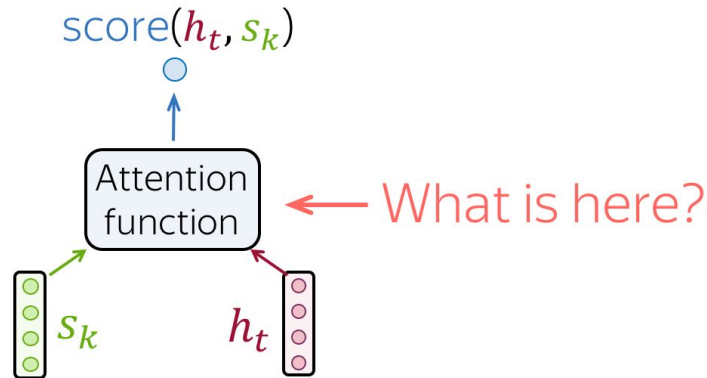


Problem: this is a bottleneck!

Encoder-Decoder architecture



Encoder-Decoder architecture



Dot-product

$$h_t^T \times s_k$$

$$\text{score}(h_t, s_k) = h_t^T s_k$$

Bilinear

$$h_t^T \times [W] \times s_k$$

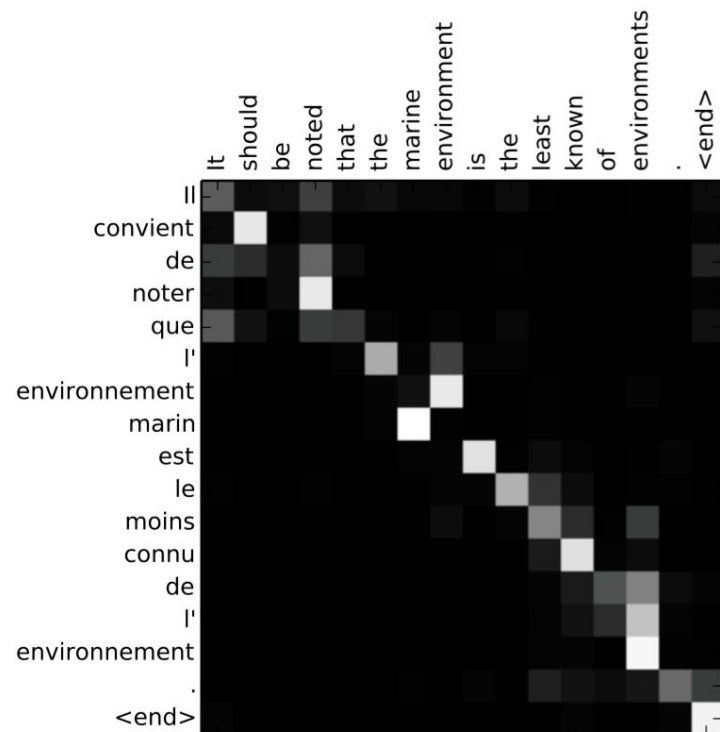
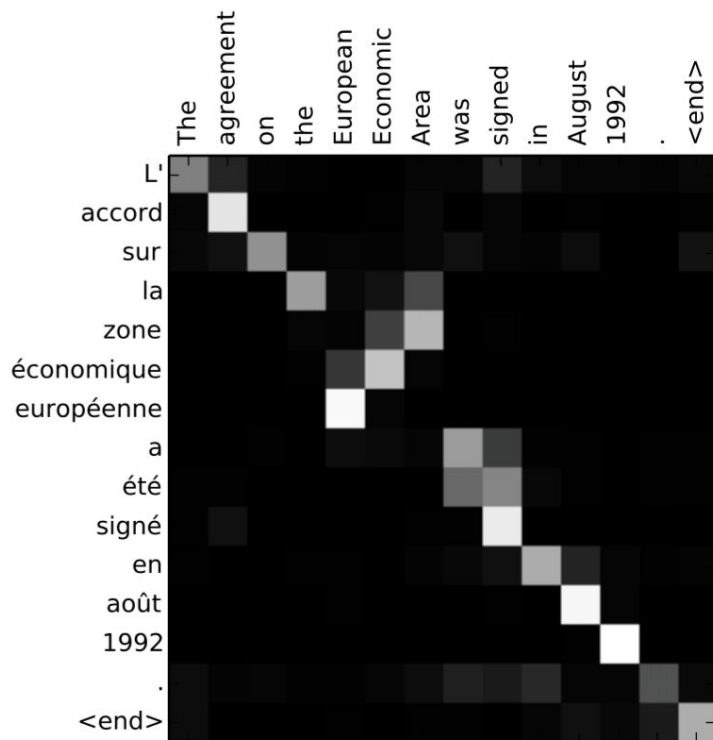
$$\text{score}(h_t, s_k) = h_t^T W s_k$$

Multi-Layer Perceptron

$$w_2^T \times \tanh \left[W_1 \times \begin{bmatrix} h_t \\ s_k \end{bmatrix} \right]$$

$$\text{score}(h_t, s_k) = w_2^T \cdot \tanh(W_1 [h_t, s_k])$$

Alignment

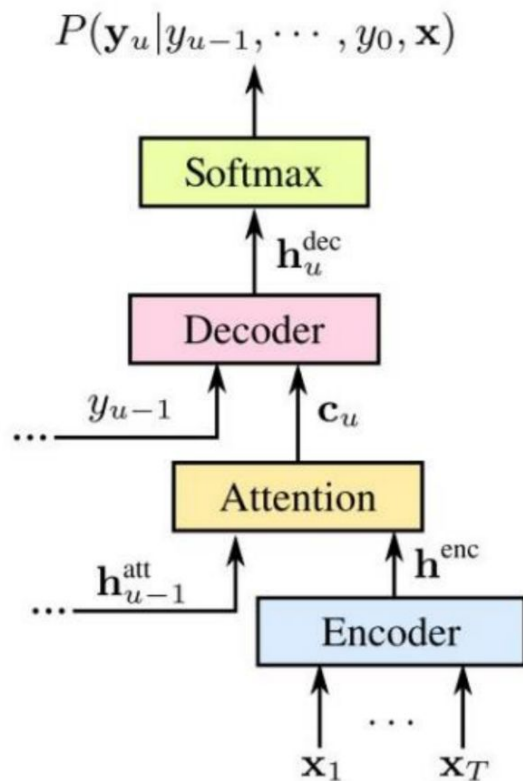


Beam Search

<bos>

Start with the begin of sentence token or with an empty sequence

Listen, Attend and Spell











- **Encoder (analogous to AM):**
 - Transforms input speech into higher-level representation
- **Attention (alignment model):**
 - Identifies encoded frames that are relevant to producing current output
- **Decoder (analogous to PM, LM):**
 - Operates autoregressively by predicting each output token as a function of the previous predictions

Listen, Attend and Spell

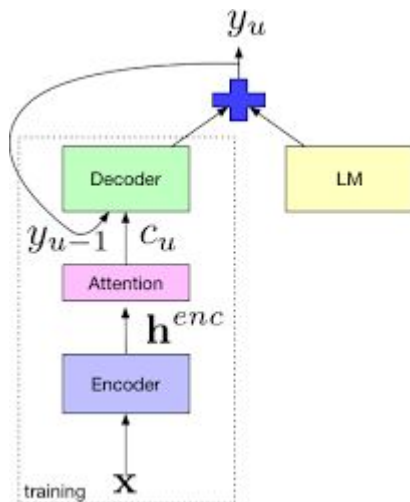
- [Link to LAS Paper](#)
- Encoder-Decoder architecture beats CTC Quartznet with LM
- Heavy inference

QuartzNet results

LM	LibriSpeech		WSJ	
	test-clean	test-other	93-test	92-eval
-	4.19	10.98	8.97	6.37
4-gram	3.21	8.04	5.57	3.51
T-XL	2.96	7.53	4.82	2.99

27	LAS + SpecAugment	2.5	×	SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition			2019	
28	Transformer	2.6	×	A Comparative Study on Transformer vs RNN in Speech Applications			2019	Transformer
29	QuartzNet15x5	2.69	×	QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions			2019	
30	LAS (no LM)	2.7	×	SpecAugment: A Simple Data Augmentation Method for Automatic Speech Recognition			2019	

Shallow Fusion

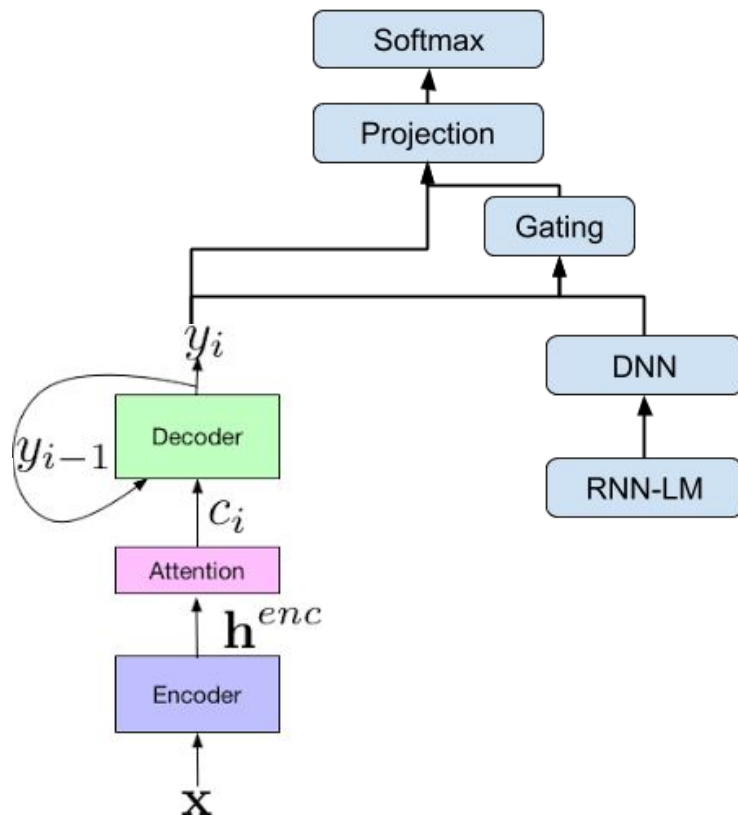


- Add language model to prediction
- Mix $y_{\{t\}}$ scores using linear rule:

$$\hat{y}_t = (1 - \alpha) \cdot y_t + \alpha \cdot l_t$$

System	Dev	Test
LAS-G	13.0	10.3
LAS-G + 20-GRAM-G	10.3	7.7
LAS-G + 3-GRAM-W	10.0	7.6
LAS-G + RNN-G	9.3	6.9

Cold Fusion



- Add language model early in model
- Combine information through gating and projection (trainable weighting)
- **Train together**

Table 3. Speech recognition results for the various models discussed in the paper.

Model	Train Domain	Test on Source		Test on Target		
		CER	WER	CER	WER	Domain Gap
Baseline Attention Model	Source	7.54%	14.68%	23.02%	43.52%	100%
Baseline Attention Model	Target			8.84%	17.61%	0%
Baseline + Deep Fusion	Source	7.64%	13.92%	22.14%	37.45%	76.57%
+ s^{AM} in gate	Source	7.61%	13.92%	21.07%	37.9%	78.31%
+ Fine-Grained Gating	Source	7.47%	13.61%	20.29%	36.69%	73.64%
+ ReLU layer	Source	7.50%	13.54%	21.18%	38.00%	78.70%
Baseline + Cold Fusion						
+ s^{AM} in gate	Source	7.25%	13.88%	15.63%	30.71%	50.56%
+ Fine-Grained Gating	Source	6.14%	12.08%	14.79%	30.00%	47.82%
+ ReLU layer	Source	5.82%	11.52%	14.89%	30.15%	48.40%
+ Probability Projection	Source	5.94%	11.87%	13.72%	27.50%	38.17%

Encoder Decoder Recap

- Encoder-Decoder does not need actual alignment, alignment is created internally
- Encoder-Decoder models are **autoregressive**, in order to make prediction the output $X_{\{n\}}$ model needs output $X_{\{n-1\}}$.
- Outputs are not conditionally independent
- Needs full input sequence in order to make prediction

Encoder Decoder Recap

- Encoder-Decoder does not need actual alignment, alignment is created internally
- Encoder-Decoder models are **autoregressive**, in order to make prediction the output $X_{\{n\}}$ model needs output $X_{\{n-1\}}$.
- Outputs are not conditionally independent
- Needs full input sequence in order to make prediction

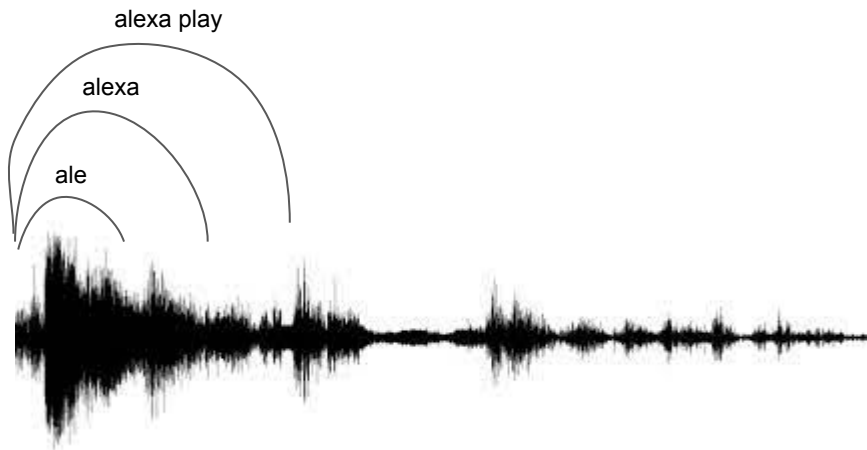
What if we want to recognize in streaming mode?

Streaming Speech Recognition

- Output result as fast as possible (ideally after letter is pronounced)
- Guarantee that already predicted words will stay the same
- Realtime speech recognition
- On Device speech recognition

Streaming Speech Recognition

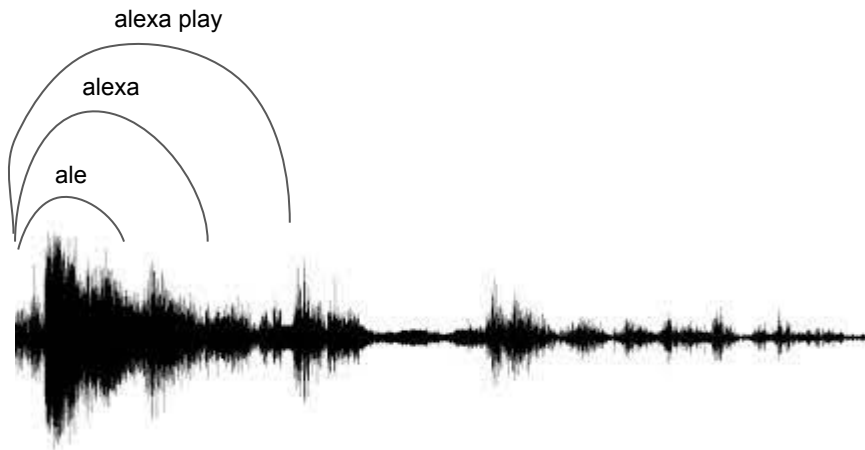
- Use LAS / CTC-LM model on GPU
- Recalculate every N milliseconds
- (Possibly) force decode previous hypotheses to guarantee already recognized outputs stay the same



Streaming Speech Recognition

A lot of processing

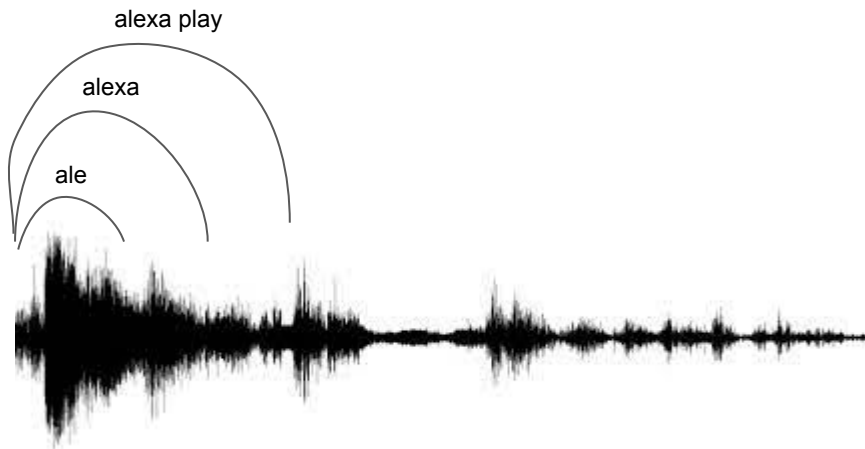
- Use LAS / CTC-LM model on GPU
- Recalculate every N milliseconds
- (Possibly) force decode previous hypotheses to guarantee already recognized outputs stay the same



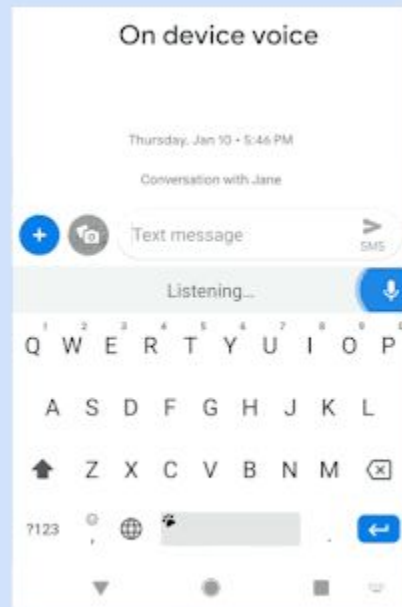
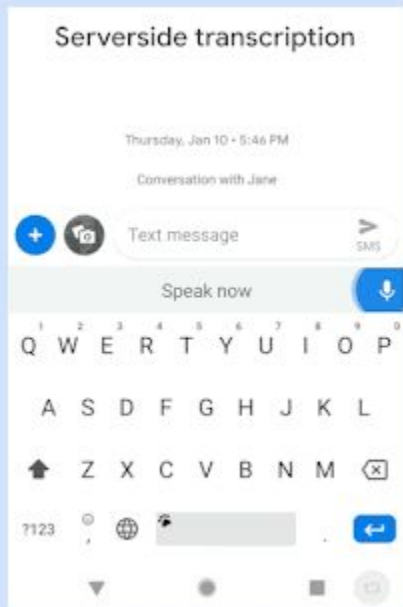
Streaming Speech Recognition

A lot of processing

- Use LAS / CTC-LM model on GPU
- Recalculate every N milliseconds
- (Possibly) force decode previous hypotheses to guarantee already recognized outputs stay the same



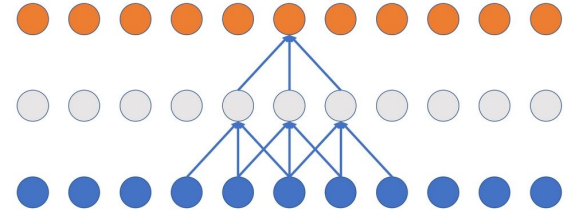
Streaming Speech Recognition



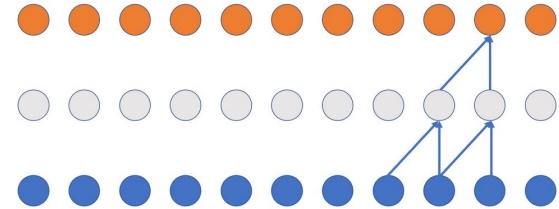
Streaming CTC model

- CTC does not need whole input
- Create streaming encoder e.g. with causal convolutions
- Fetch output

Standard Convolution



Causal Convolution



Streaming CTC model

- CTC Loss accepts any alignment
- Often latency of produced words is high
- Models tends to “save up” information
- WER degradation is quite high
- Use big language model in decoding time, device computation power does not allow that
- What happens, if we left with N frames to output but have $M > N$ letters to output?

chunk size (right)	WER(%)	
	test clean	test other
Full	3.53	8.52
200 (2.00s)	3.78	9.38
64 (0.64s)	4.06	9.98
32 (0.32s)	4.30	10.55
16 (0.16s)	5.88	12.01

Streaming CTC model

- CTC Loss accepts any alignment
- Often latency of produced words is high
- Models tends to “save up” information
- WER degradation is quite high
- Use big language model in decoding time, device computation power does not allow that
- What happens, if we left with N frames to output but have $M > N$ letters to output?

Output will never be correct

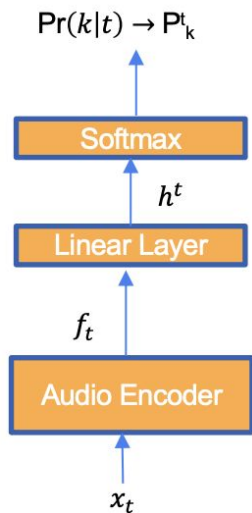


chunk size (right)	WER(%)	
	test clean	test other
Full	3.53	8.52
200 (2.00s)	3.78	9.38
64 (0.64s)	4.06	9.98
32 (0.32s)	4.30	10.55
16 (0.16s)	5.88	12.01

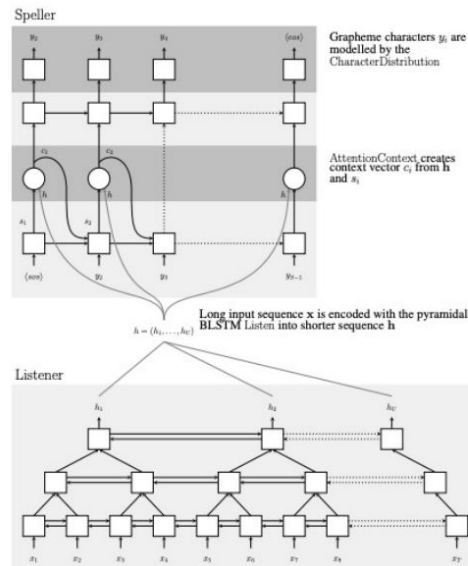
Streaming Speech Recognition

- Need more powerful approach to streaming ASR
 - **Conditional independence** is a problem for streaming
 - One character per frame is a problem for streaming
-
- Merge CTC and LAS approaches into new RNN-Transducer approach
 - RNN-T elegantly solves both problems associated with CTC, while retaining some of its advantages over attention models

Conditional Independence Assumption



>



CTC:

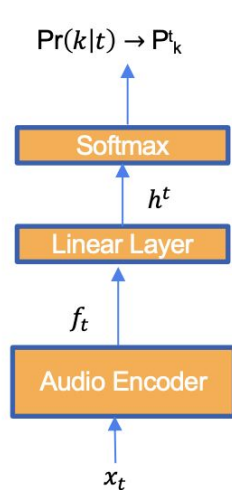
- Only uses audio embedding at time "t" (x_t) to generate probability distribution over output units. CTC assumes that each output unit is conditionally independent of others.
- Streaming

LAS:

Attends to all audio embeddings and uses text history produced so far to generate probability distribution over output units.

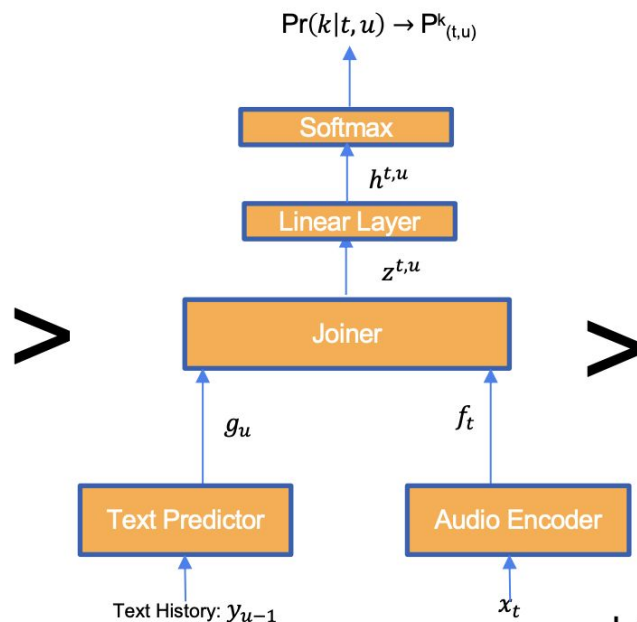
$$f(y_u | y_{u-1}, y_{u-2}, \dots, y_0, h_0, h_2, h_3, \dots, h_v)$$

Conditional Independence Assumption



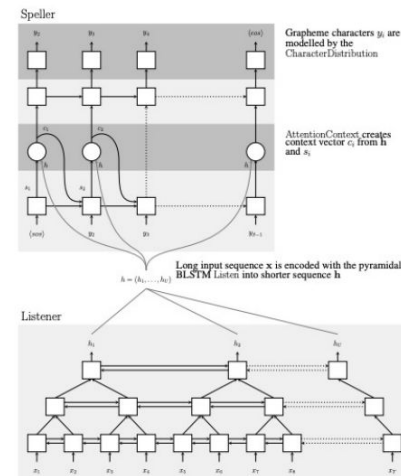
CTC:

- Only uses audio embedding at time "t" (x_t) to generate probability distribution over output units. CTC assumes that each output unit is conditionally independent of others.
- Streaming



RNN-T:

- Uses both audio embedding at time "t" (x_t) and text history produced so far (y_{u-1}) to generate probability distribution over output units.
- Streaming



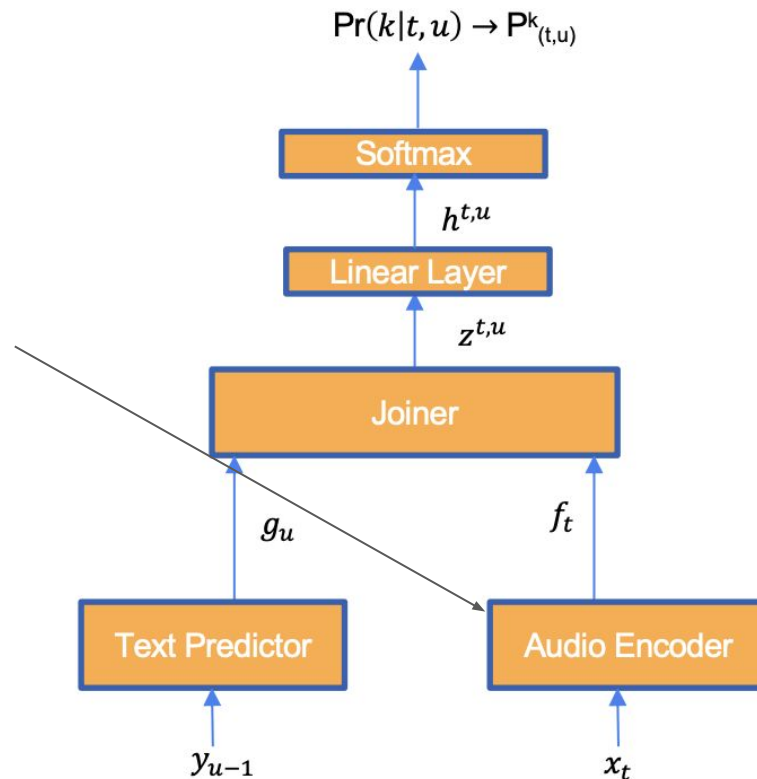
LAS:

Attends to all audio embeddings and uses text history produced so far to generate probability distribution over output units.

$$f(y_u | y_{u-1}, y_{u-2}, \dots, y_0, h_0, h_2, h_3, \dots, h_v)$$

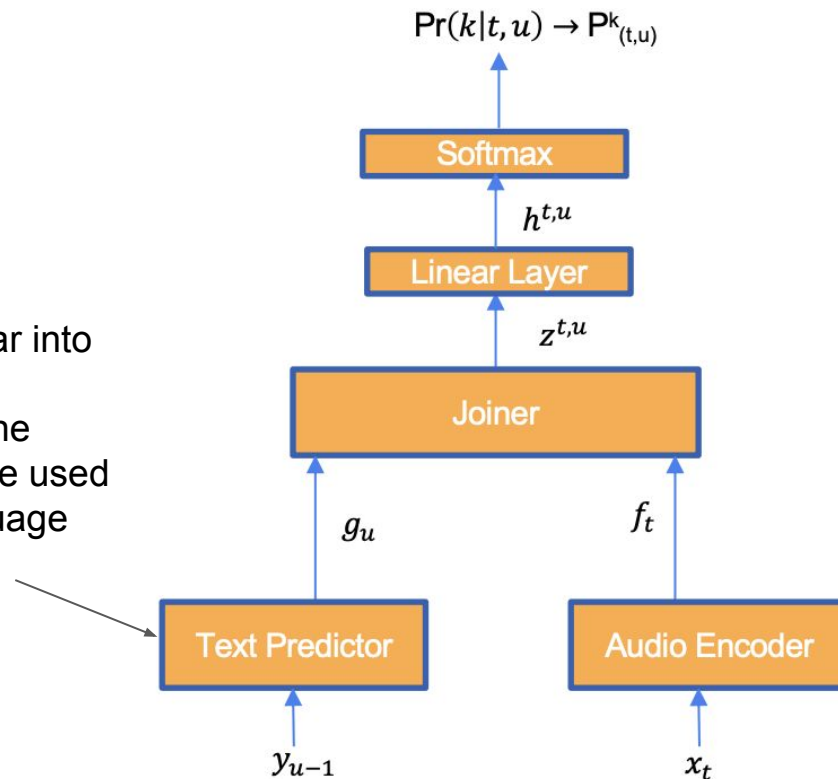
RNN-T

- Audio Encoder: To encode sequence of audio features into audio embeddings. We can use Conformer, Quartznet, or streaming Causal Convolution network



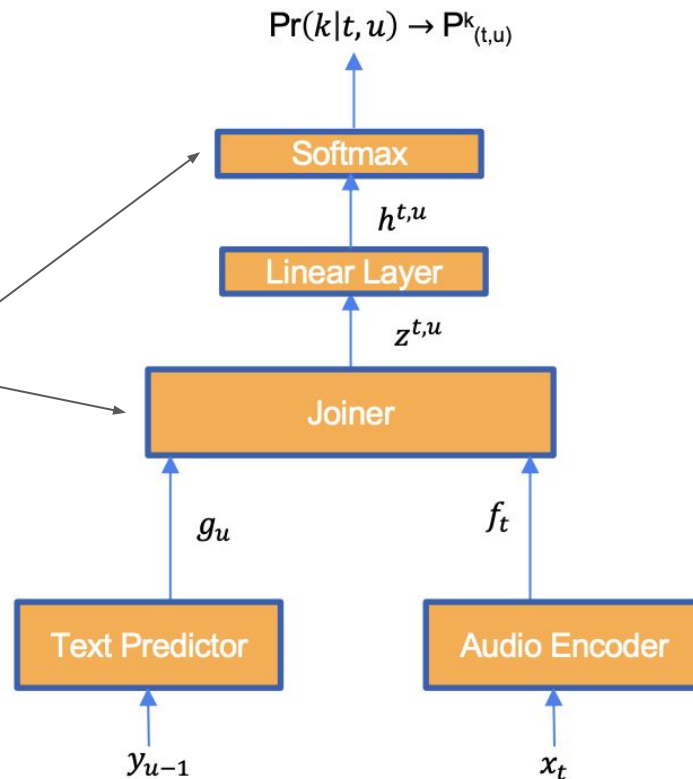
RNN-T

- Text Predictor: To encode transcript produced so far into text embedding. Typically a LSTM.
Text Predictor is autoregressive: it takes as input the previous outputs and produces features that can be used for predicting the next output, like a standard language model.



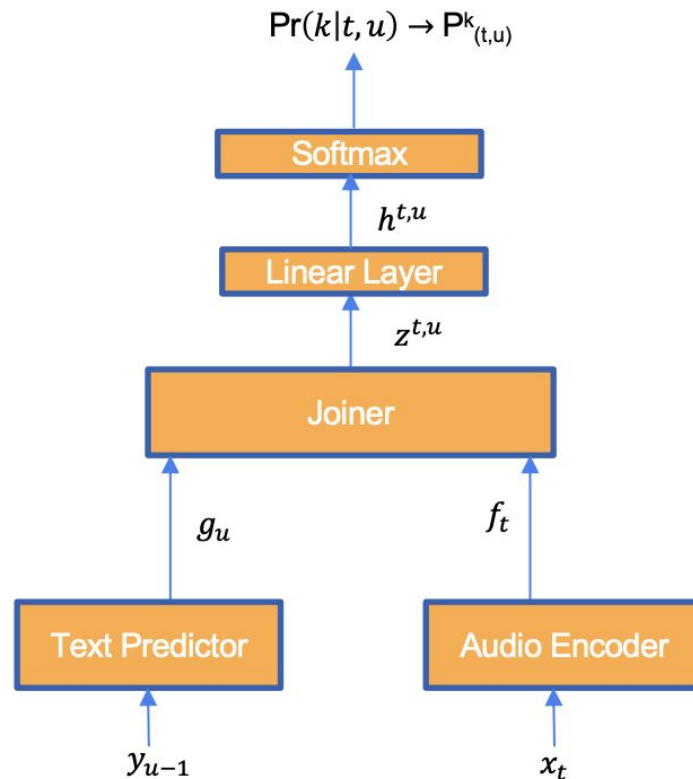
RNN-T

- Joiner combines output of audio encoder and text predictor.
- Softmax probability distribution over output units. $P(k|t, u)$ is probability of emitting k from time t and text predictor vector u



RNN-T

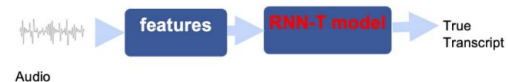
- Note: we don't need repetition of characters: prediction network decides to output a symbol
- We allow produce more than 1 output for each time frame t



RNN-T

Training Objective

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t) : 4
- We don't know alignment i.e. which portion of audio aligns to what output unit (A path taken in lattice)
- Training Objective

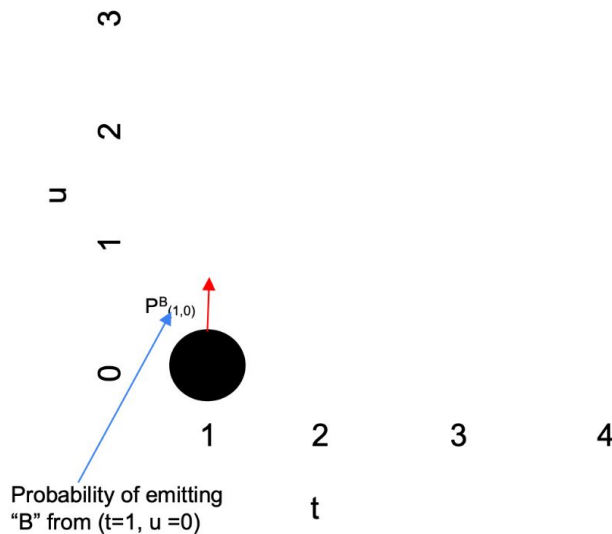
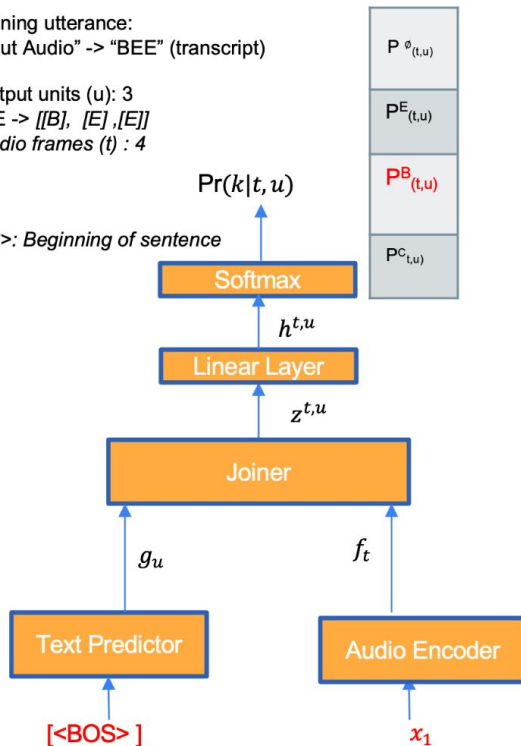


$$P(BEE|X) = \sum_{\text{alignment}} P(\text{alignment}, BEE|X)$$
$$P(BEE|\text{alignment}, X) = 1 \quad \sum_{\text{alignment}} P(BEE|\text{alignment}, X) P(\text{alignment}|X)$$
$$\sum_{\text{alignment}} P(\text{alignment}|X)$$

RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t): 4

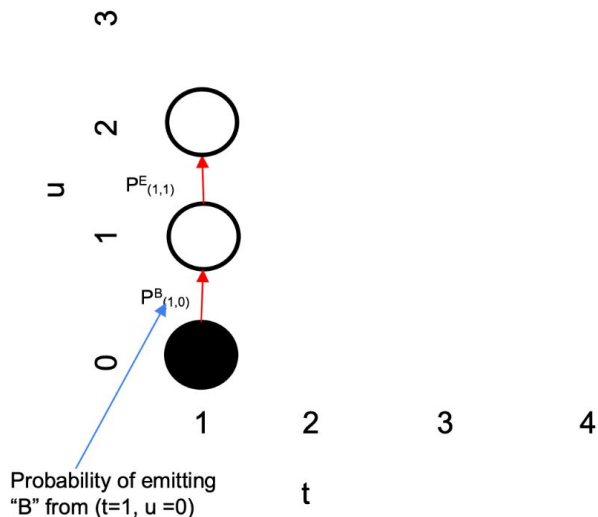
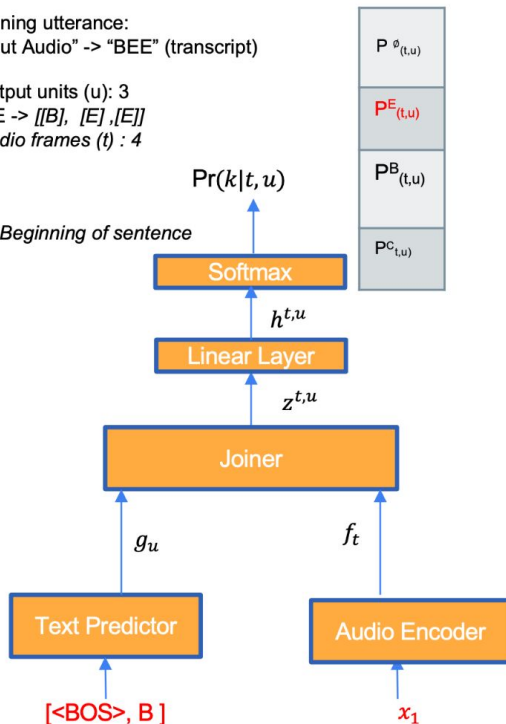


Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t): 4

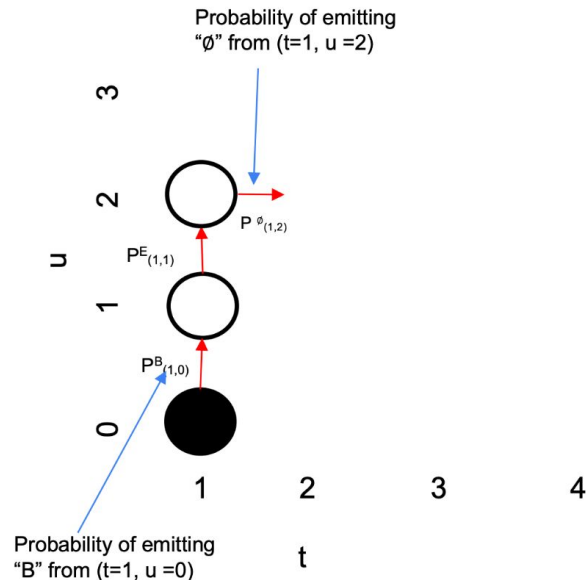
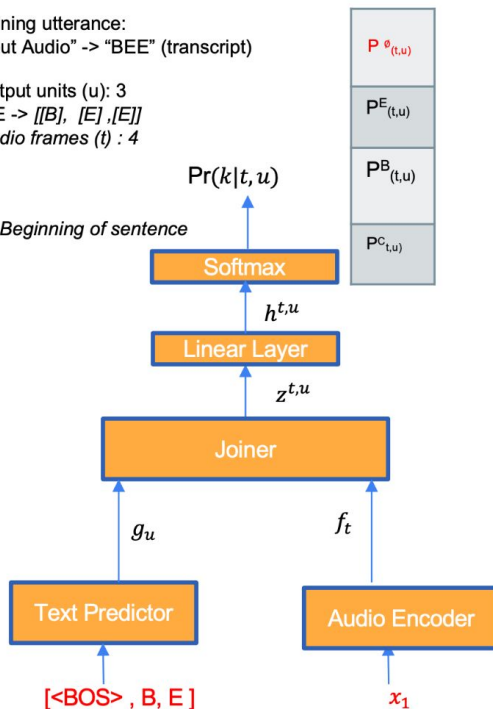


Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t): 4



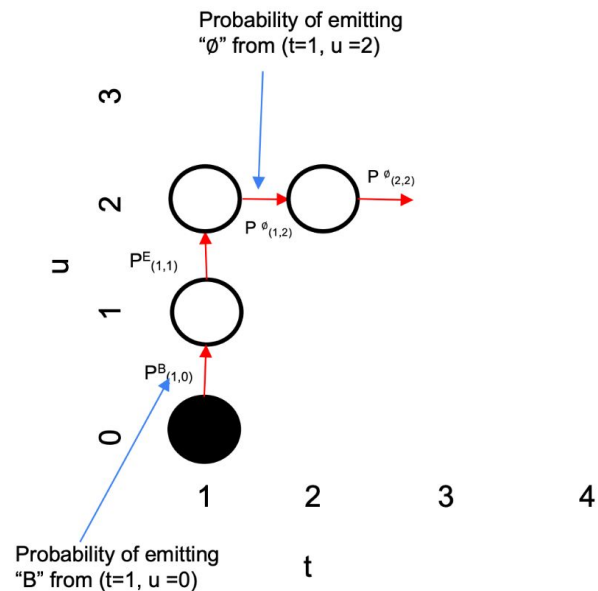
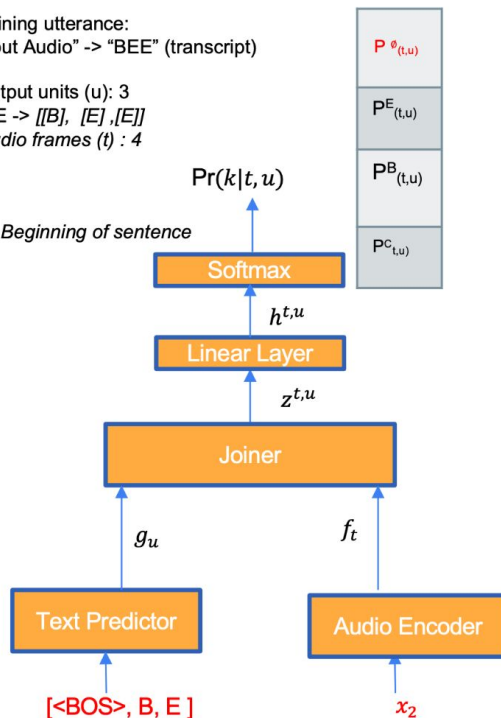
Graves et al., "[Sequence transduction with recurrent neural networks](https://arxiv.org/abs/1609.09262)"

RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t): 4

- BOS: Beginning of sentence

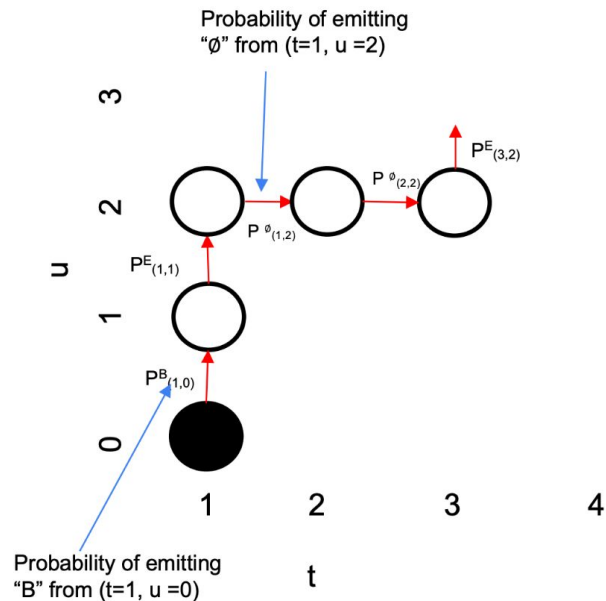
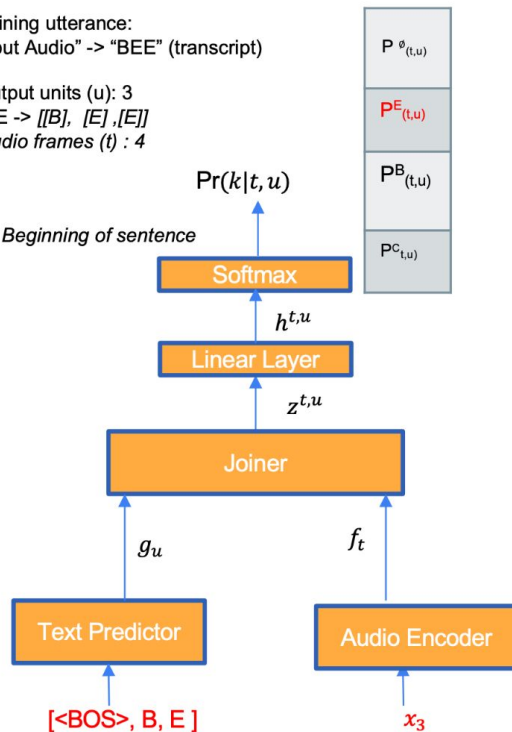


Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T

An Alignment During Training

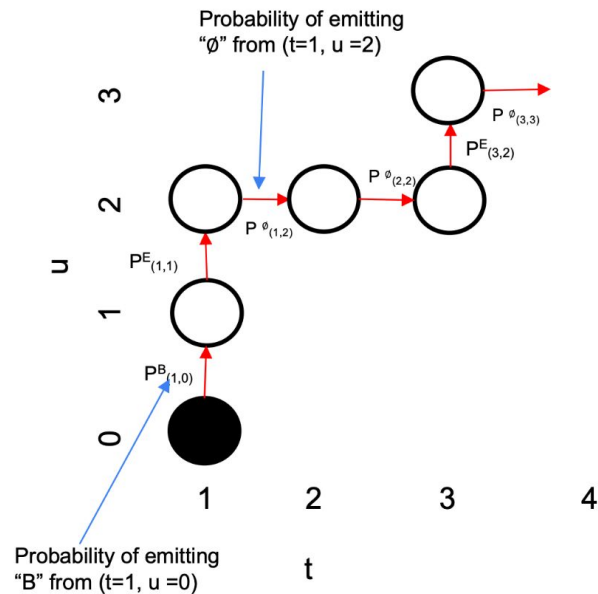
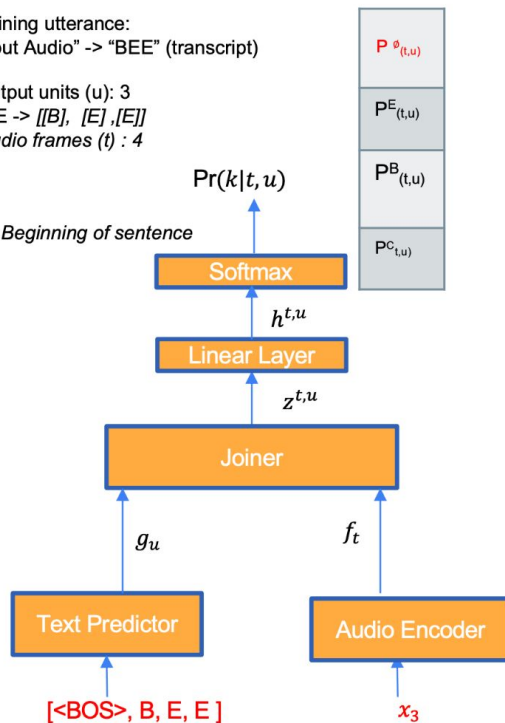
- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t): 4



RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t) : 4
- BOS: Beginning of sentence

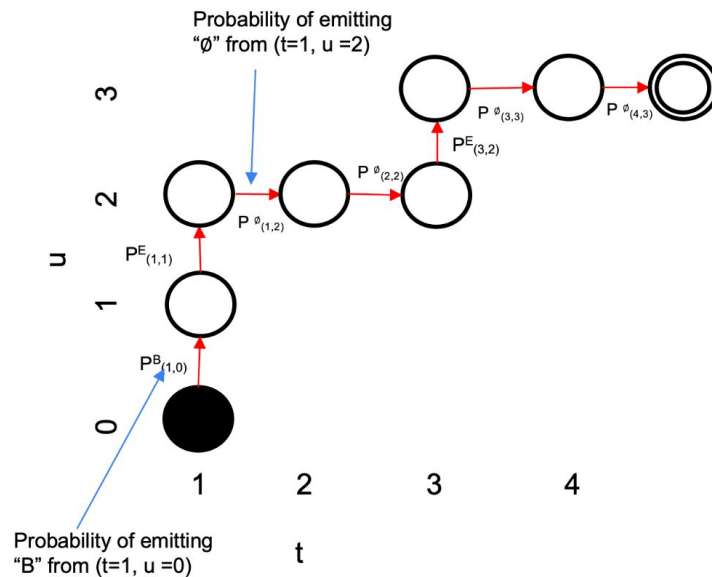
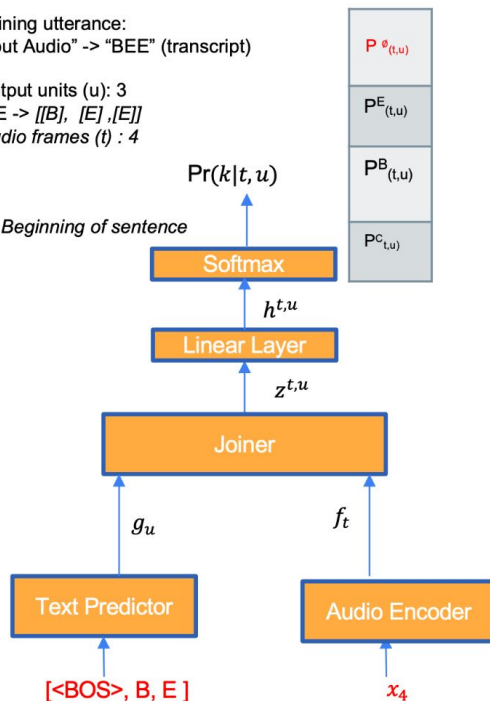


Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T

An Alignment During Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)
- #output units (u): 3
BEE -> $[[B], [E], [E]]$
- #audio frames (t) : 4



Graves et al., "Sequence transduction with recurrent neural networks"

RNN-T

RNN-T Lattice And Training

- Training utterance:
"Input Audio" -> "BEE" (transcript)

- #output units (u): 3
BEE -> [[B], [E], [E]]
- #audio frames (t) : 4

- We don't know alignment i.e.
which portion of audio aligns to what
output unit

- Probability of alignment is multiplication of probabilities assigned along
the path of alignment

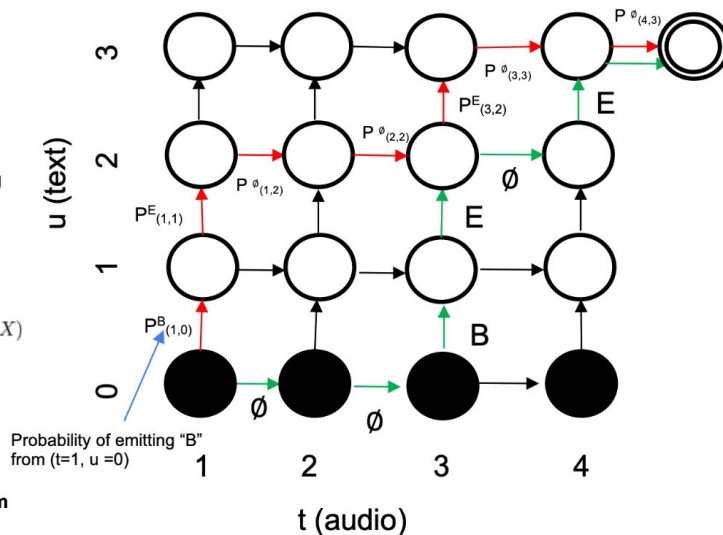
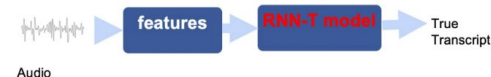
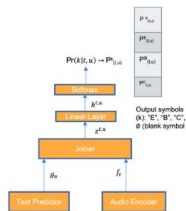
- Training

$$P(BEE|X) = \sum_{\text{alignment}} P(\text{alignment}, BEE|X)$$

$$P(BEE|\text{alignment}, X) = 1$$

$$\sum_{\text{alignment}} P(\text{alignment}|X)$$

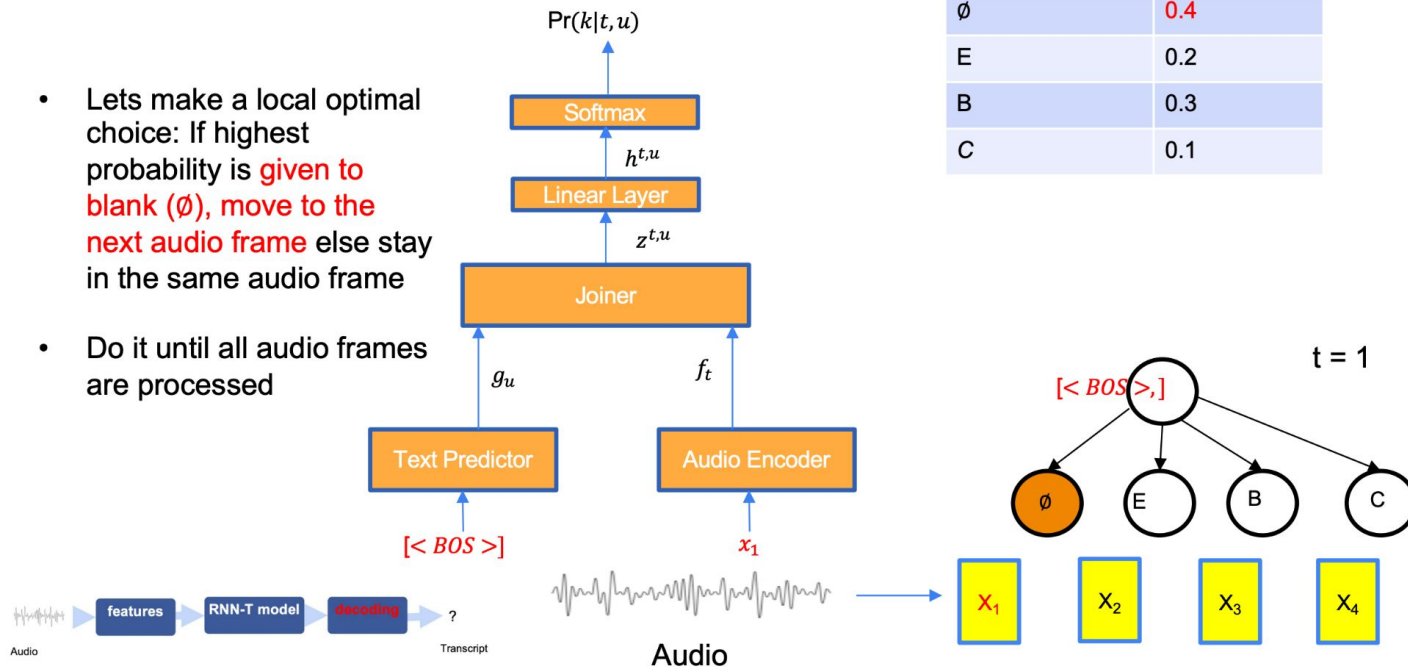
- Lattice contains all valid alignment paths(traversals). **During training, we change (optimize) neural network parameters to maximize sum of probabilities of all alignment paths**
- Computation is done efficiently through dynamic programming (slide 54 to 58)**



RNN-T

Greedy Decoding






- Lets make a local optimal choice: If highest probability is **given to blank (\emptyset)**, move to the **next audio frame** else stay in the same audio frame
- Do it until all audio frames are processed



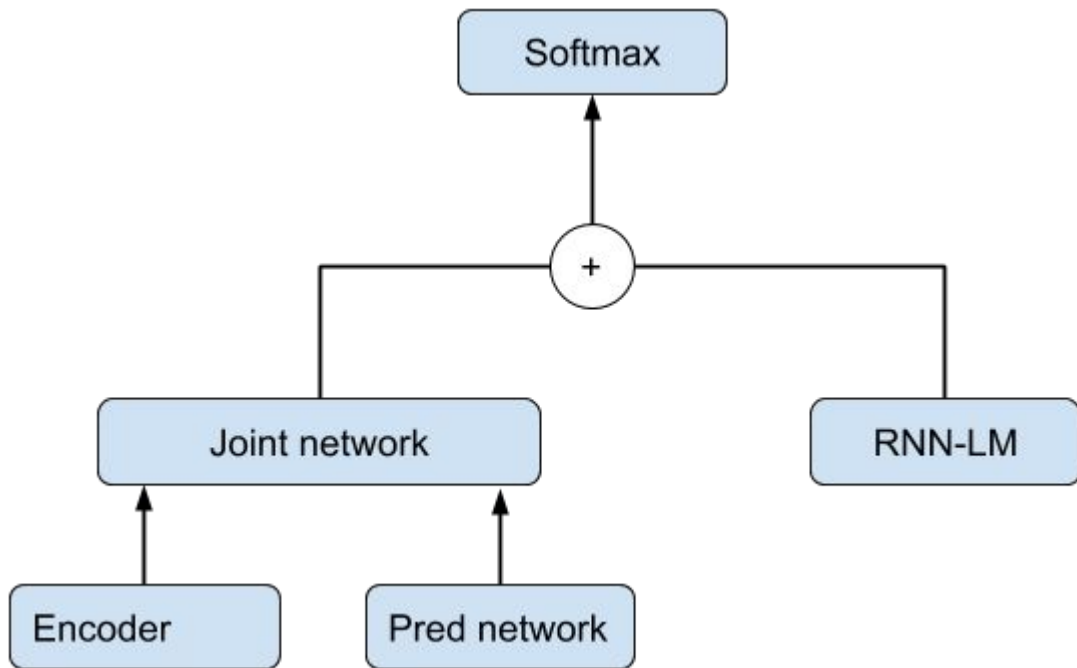
RNN-T

- RNN-T introduces audio-module and text-module
- Audio module encodes spectrogram, text module encodes predicted text
- Aggregate prediction using joint network
- Simpler rules than CTC
- Autoregressive model
- Comparable accuracy
- Better for streaming

RNN-T is used in practice to deploy on device or CPU ASR. Audio and Text encoders must be unidirectional

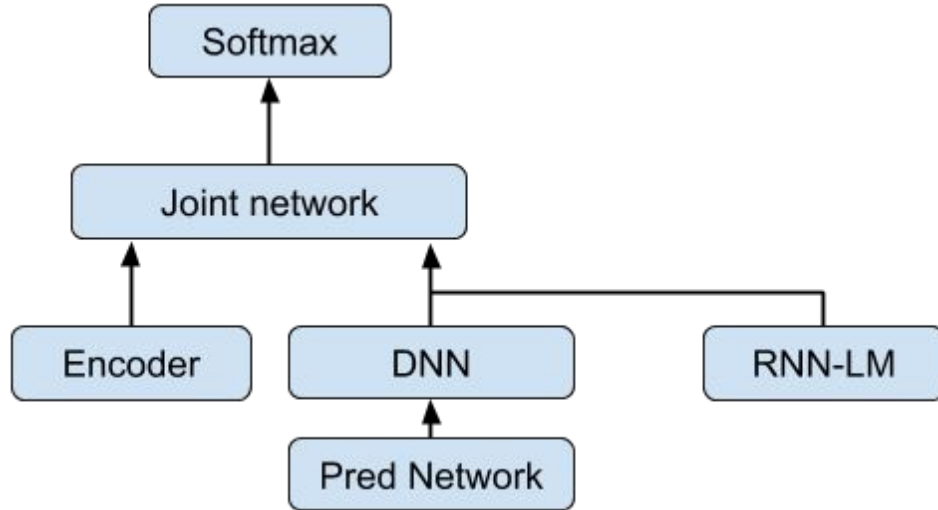
12	ContextNet (M)	2	×	ContextNet: Improving Convolutional Neural Networks for Automatic Speech Recognition with Global Context	 	2020	
13	Transformer Transducer	2.0	×	Improving RNN Transducer Based ASR with Auxiliary Tasks		2020	Transformer
14	Conformer (M)	2	×	Conformer: Convolution-augmented Transformer for Speech Recognition	 	2020	Conformer

RNN-T Shallow Fusion



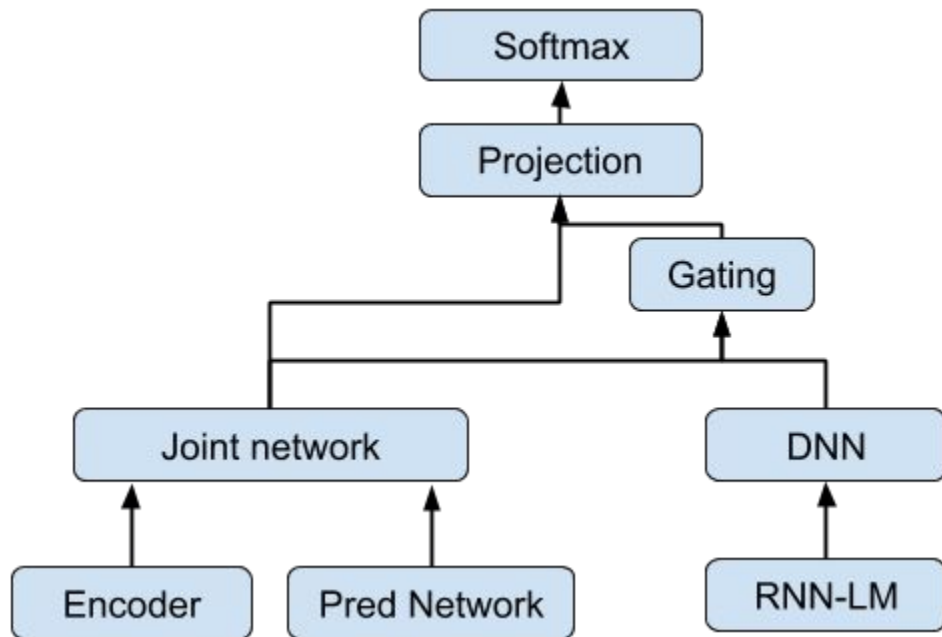
- Add language model to prediction similar to LAS

RNN-T Early shallow fusion



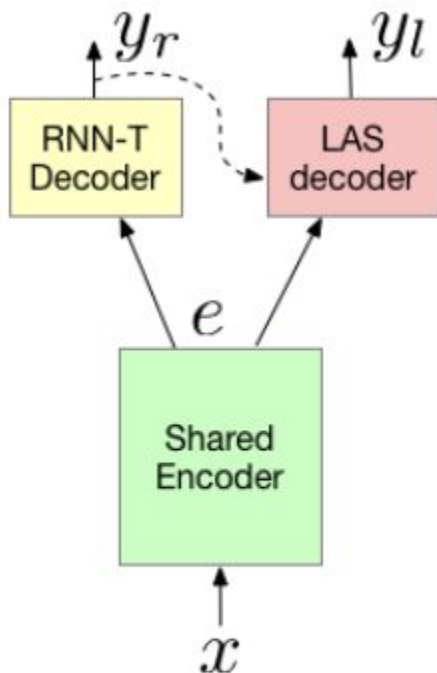
- We can also add language model to prediction network
- Though works at google <https://arxiv.org/pdf/2012.06749.pdf> shows that prediction network size is not that important

RNN-T Cold Fusion



- Cold fusion RNN-T and language model
- Train together

Two-Pass architecture



- Train RNN-T and LAS together
- Rescore with LAS head in the end

Table 2: *WER results, LAS Rescoring.*

Exp-ID	Decoding	SU	LU
<i>B0</i>	RNN-T	6.9	4.5
<i>B1</i>	LAS-only	5.4	4.5
<i>E1</i>	Beam Search	6.1	4.8
<i>E2</i>	Rescoring	6.2	4.1

Figure 1: *Two-Pass Architecture*

Next time

- Wav2Vec2.0
- Multi model systems