

Speech Recognition Algorithms and Optimization Neural Transduces & LAS

Wonyong Sung
Signal Processing Systems Lab @ SNU



End-to-end Speech Recognition

- Modularized approach (Hybrid)
 - Acoustic model (CTC)
 - Language model and beam-search
 - Separate optimization
- Pure end2end approach
 - Integration of AM and LM
 - LAS (Listen Attend and Spell) (Chan et al., "Listen, Attend and Spell")
 - RNN-T (Transducer) (Graves et al., "Sequence transduction with recurrent neural networks")



Contents

- Neural Transducers
- Listen Attend Spell (LAS)
- Comparison of CTC, Neural Transducers, LAS
- Implementation Considerations

Some RNN-T slides from

https://www.cc.gatech.edu/classes/AY2021/cs7643_spring/assets/L24_rnnt_asr_tutorial_gt.pdf



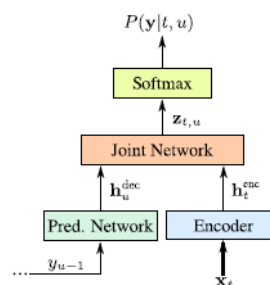
CTC Acoustic Model 의 문제

- CTC 의 independence 가정이 성능에 미치는 영향
- AM에서 output label 을 얻은 뒤에 그 결과를 좋기 위해서 LM(language model)을 이용한 beam-search decoding 을 해야 하는데, 좋은 성능을 얻기 위해서는 beam width 가 커야 한다.
- 이 beam width 가 보통 32~ 128 을 사용한다.
- RNN LM을 사용할 경우 전체 계산량의 측면에서 AM 보다 RNN LM 에 필요한 것이 훨씬 더 많다.
(N-gram LM의 경우에는 계산이 적지만 (memory look-up) 성능이 떨어지고 또 병렬처리 계산의 이점을 살리지 못한다. 갈수록 RNN 이나 Transformer LM 대비 경쟁력이 떨어진다.
- 해결 - AM과 LM 을 integrate 하여 end-to-end training 한다.



Neural Transducers

- Neural Transducers: Similar to CTC (encoder), but include LM (prediction network) to overcome the conditional independence problem of CTC
- Encoder (acoustic modeling), prediction network (LM), joint network (assemble the acoustic and prediction)
- Joint network merges the probabilities in algorithm or using feed-forward neural network.
- Graves, Alex. "Sequence transduction with recurrent neural networks." *arXiv preprint arXiv:1211.3711* (2012).

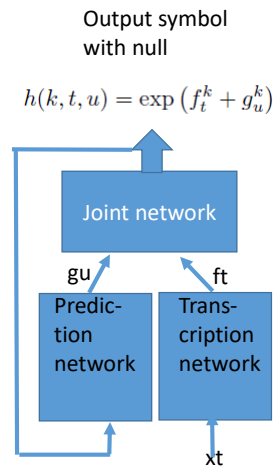


2. Neural Transducers

- Graves, Alex. "Sequence transduction with recurrent neural networks." *arXiv preprint arXiv:1211.3711* (2012).
- Rao, Kanishka, Haşim Sak, and Rohit Prabhavalkar. "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer." *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017.



- Transcription network (acoustic model): speech input x_t to its label probabilities (K+1 labels including null)
- (Text) Prediction network: a kind of language model with K+1 output labels (including null label)



Transcription network

- Acoustic input to labels.
- Bidirectional RNN: the labels include null (K+1 labels). In Graves 2012 paper, phonemes are used for labels.

$$\overleftarrow{h}_t = \mathcal{H} \left(W_{i\overleftarrow{h}} i_t + W_{\overleftarrow{h}\overleftarrow{h}} \overleftarrow{h}_{t+1} + b_{\overleftarrow{h}} \right) \quad (9)$$

$$\overrightarrow{h}_t = \mathcal{H} \left(W_{i\overrightarrow{h}} i_t + W_{\overrightarrow{h}\overrightarrow{h}} \overrightarrow{h}_{t-1} + b_{\overrightarrow{h}} \right) \quad (10)$$

$$o_t = W_{\overrightarrow{h}o} \overrightarrow{h}_t + W_{\overleftarrow{h}o} \overleftarrow{h}_t + b_o \quad (11)$$

Prediction network

- The input is one-hot encoded to represent K labels (all zero corresponds to the blank label)
- The output is the probability distribution of K+1 labels (including blank)
- The output sequence is modeled as (blank, y_1, y_2, \dots, y_U)
- The network can be built using uni-directional LSTM RNN

$$\alpha_n = \sigma(W_{i\alpha}i_n + W_{h\alpha}h_{n-1} + W_{s\alpha}s_{n-1}) \quad (4)$$

$$\beta_n = \sigma(W_{i\beta}i_n + W_{h\beta}h_{n-1} + W_{s\beta}s_{n-1}) \quad (5)$$

$$s_n = \beta_n s_{n-1} + \alpha_n \tanh(W_{is}i_n + W_{hs}) \quad (6)$$

$$\gamma_n = \sigma(W_{i\gamma}i_n + W_{h\gamma}h_{n-1} + W_{s\gamma}s_n) \quad (7)$$

$$h_n = \gamma_n \tanh(s_n) \quad (8)$$



Output distribution

Given the transcription vector f_t , where $1 \leq t \leq T$, the prediction vector g_u , where $0 \leq u \leq U$, and label $k \in \bar{\mathcal{Y}}$, define the *output density function*

$$h(k, t, u) = \exp(f_t^k + g_u^k) \quad (12)$$

where superscript k denotes the k^{th} element of the vectors. The density can be normalised to yield the conditional *output distribution*:

$$\Pr(k \in \bar{\mathcal{Y}}|t, u) = \frac{h(k, t, u)}{\sum_{k' \in \bar{\mathcal{Y}}} h(k', t, u)} \quad (13)$$

To simplify notation, define

$$y(t, u) \equiv \Pr(y_{u+1}|t, u) \quad (14)$$

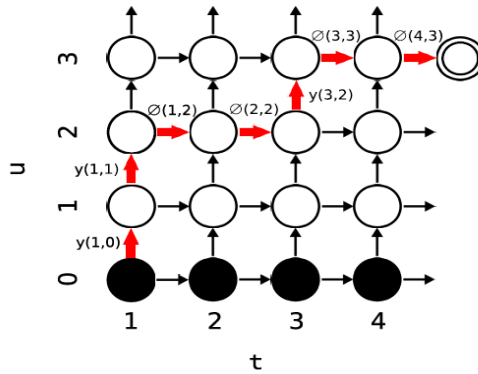
$$\varnothing(t, u) \equiv \Pr(\varnothing|t, u) \quad (15)$$

$\Pr(k|t, u)$ is used to determine the transition probabilities in the lattice shown in Fig. 1. The set of possible



Output probability lattice

- Moving horizontal direction: Probability of horizontal arrow leaving node t, u represents the prob of $\text{Null}(t, u)$
- Moving vertical direction: represents the probability of $y(t, u)$ of outputting element $u+1$ of y .



Define the *forward variable* $\alpha(t, u)$ as the probability of outputting $y_{[1:u]}$ during $f_{[1:t]}$. The forward variables for all $1 \leq t \leq T$ and $0 \leq u \leq U$ can be calculated recursively using

$$\alpha(t, u) = \alpha(t-1, u)\varnothing(t-1, u) + \alpha(t, u-1)y(t, u-1)$$

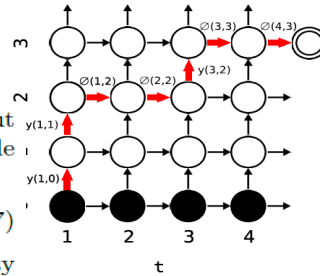
with initial condition $\alpha(1, 0) = 1$. The total output sequence probability is equal to the forward variable at the terminal node:

$$\Pr(y|x) = \alpha(T, U)\varnothing(T, U) \quad (17)$$

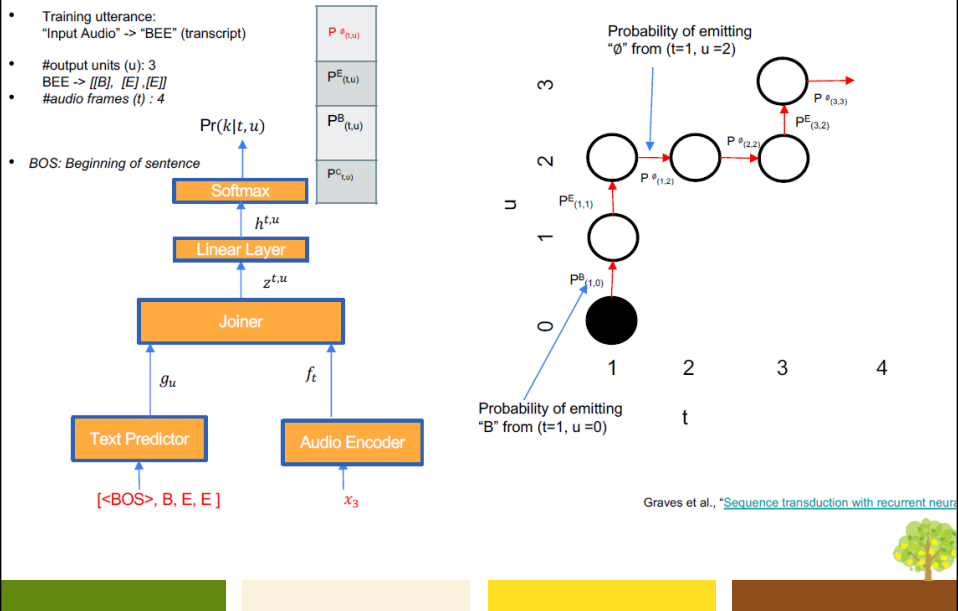
Define the *backward variable* $\beta(t, u)$ as the probability of outputting $y_{[u+1:U]}$ during $f_{[t:T]}$. Then

$$\beta(t, u) = \beta(t+1, u)\varnothing(t, u) + \beta(t, u+1)y(t, u) \quad (18)$$

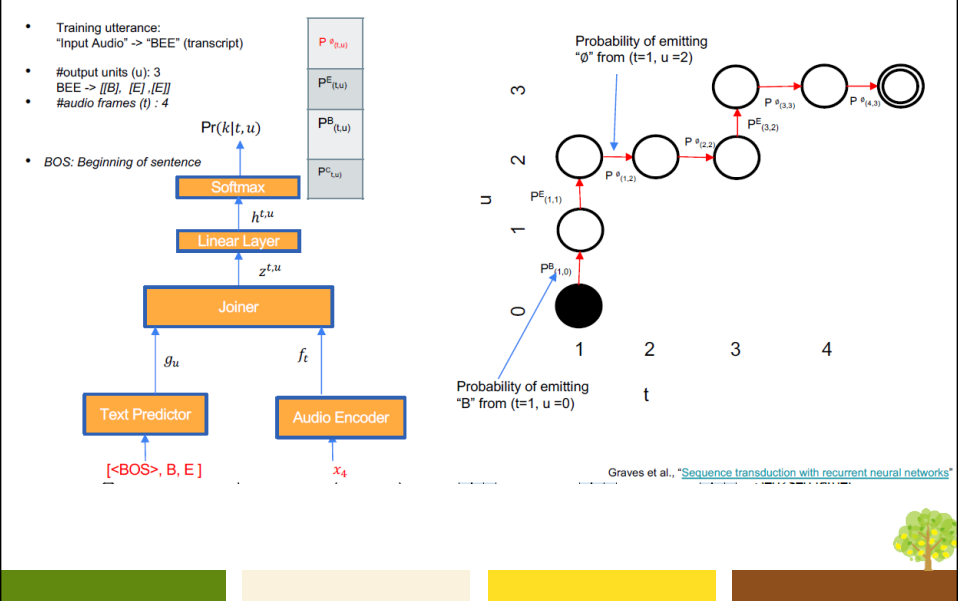
with initial condition $\beta(T, U) = \varnothing(T, U)$. From the definition of the forward and backward variables it follows that their product $\alpha(t, u)\beta(t, u)$ at any point (t, u) in the output lattice is equal to the probability of emitting the complete output sequence *if* y_u is emitted during transcription step t . Fig. 2 shows a plot



Alignment example during training (1)



Alignment example (2)



RNN-T lattice

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

A subset of hypotheses

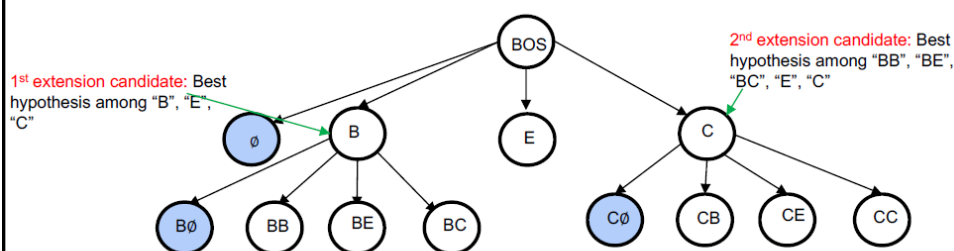
We can continue generating symbols from every hypothesis that has not yet emitted \emptyset

Beam search

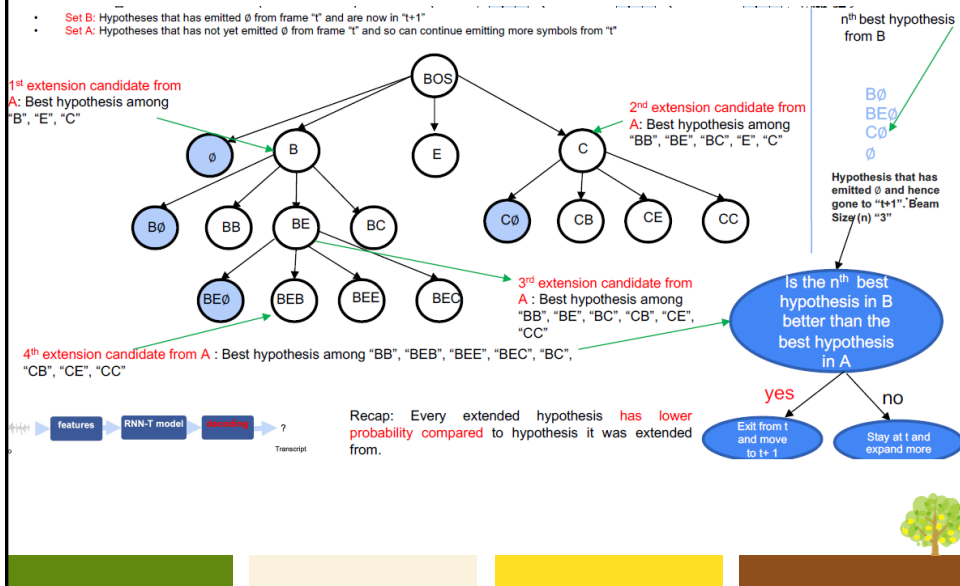
- We do not want to stay in time frame “t” forever. But we do not know when to move to “t+1” as there are many candidates that can be emitted at “t”.
- n candidate hypotheses at time frame before exiting to decode at “t”. n is the hyper parameter.



Beam search



Beam search with beamwidth (n)



Beam search algorithm

- Set B: Hypotheses that a blank has been output from frame t
- Set A: Hypotheses that a blank has not been output from frame t
- Take the best hypothesis from A and extend it with each of the output symbols and \emptyset
- Exit the beam search at audio frame t if B has more than W (beam size) hypotheses that are more probable than most probable hypothesis in A.
- When starting the beam search at audio frame $t+1$: 1) Empty A, 2) Move all Hypotheses from B to A, 3) compute the prefix ($pref$) completion probability and, 4) do prefix accumulation.
- Prefix completion probability ($\Pr(y|\hat{y}, t)$) for proper prefixes ($\hat{y} \in pref(y)$) of each hypothesis (y) is computed by outputting the symbols of symbol index (u) from $(|y| + 1)$ to $(|y|)$ at audio frame (t). (slide 60 has more context about it)

$$\Pr(y|\hat{y}, t) = \prod_{u=|\hat{y}|+1}^{|y|} \Pr(y_u|y_{[0:u-1]}, t)$$

- Prefix accumulation for each hypothesis in A is done by following:

for y in A do
 $\Pr(y) \leftarrow \sum_{\hat{y} \in pref(y) \cap A} \Pr(\hat{y}) \Pr(y|\hat{y}, t)$
 end for

Algorithm 1 Output Sequence Beam Search

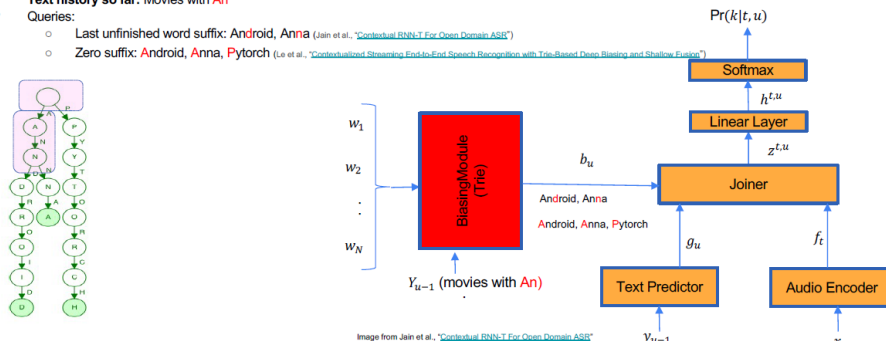
Initialise: $B = \{\emptyset\}$; $\Pr(\emptyset) = 1$
 for $t = 1$ to T do
 $A = B$
 $B = \{\}$
 for y in A do
 $\Pr(y) \leftarrow \sum_{\hat{y} \in pref(y) \cap A} \Pr(\hat{y}) \Pr(y|\hat{y}, t)$
 end for
 while B contains less than W elements more probable than the most probable in A do
 $y^* =$ most probable in A
 Remove y^* from A
 $\Pr(y^*) = \Pr(y^*) \Pr(\emptyset|y^*, t)$
 Add y^* to B
 for $k \in \mathcal{Y}$ do
 $\Pr(y^* + k) = \Pr(y^*) \Pr(k|y^*, t)$
 Add $y^* + k$ to A
 end for
 end while
 Remove all but the W most probable from B
 end for
 Return: y with highest $\log \Pr(y)/|y|$ in B



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

Utilizing context specific biasing modules

- Train RNN-T model using: **Utterance specific context words** (Android, Anna, Pytorch) along with **Audio, True Transcript**
- Biasing Module:** Built using Trie (data structure) of all utterance specific context words. Trie is used to find what words from context list can be finished from last unfinished word in text history so far
- Text history so far:** Movies with **An**
- Queries:
 - Last unfinished word suffix: **Android, Anna** (Jain et al., "Contextual RNN-T For Open Domain ASR")
 - Zero suffix: **Android, Anna, Pytorch** (Le et al., "Contextualized Streaming End-to-End Speech Recognition with Trie-Based Deep Biasing and Shallow Fusion")

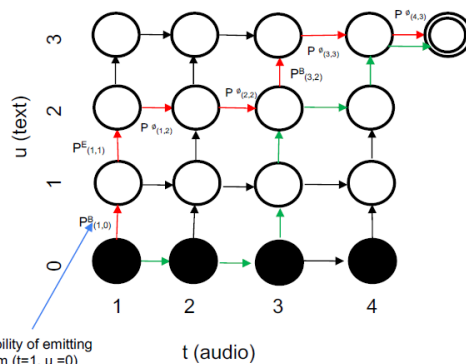


Training – lattice with complete set of paths

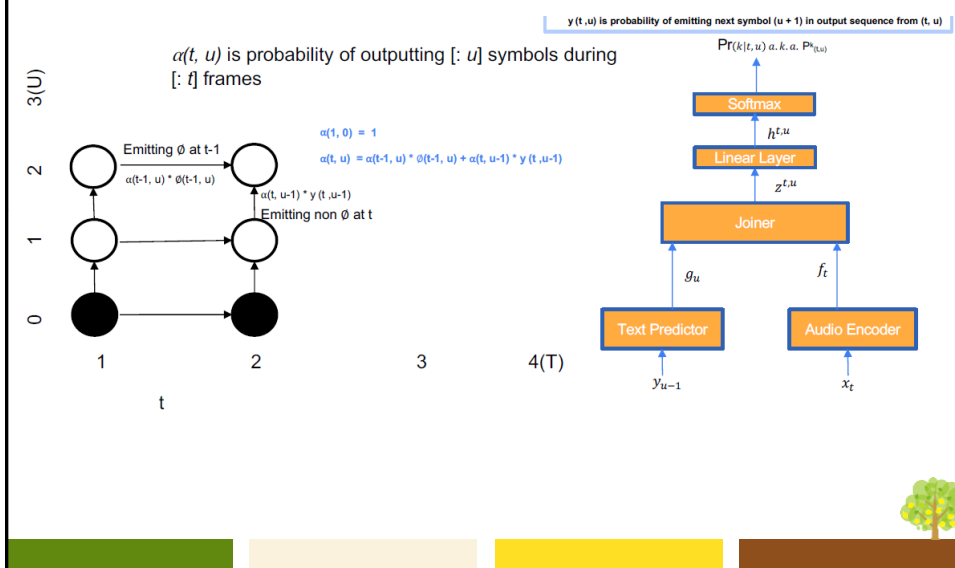
$$P(BEE|X) = \sum_{\text{alignment}} P(\text{alignment}, BEE|X)$$

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

- A naïve implementation is computationally heavy: Use dynamic programming to efficiently compute this.



Training with forward and backward variables



EXPLORING ARCHITECTURES, DATA AND UNITS FOR STREAMING END-TO-END SPEECH RECOGNITION WITH RNN-TRANSDUCER

Kanishka Rao, Haşim Sak, Rohit Prabhavalkar

Google Inc.,
 Mountain View, CA, U.S.A.
 {kanishkarao, hasim, prabhavalkar}@google.com

- Rao, Kanishka, Haşim Sak, and Rohit Prabhavalkar. "Exploring architectures, data and units for streaming end-to-end speech recognition with RNN-transducer." *Automatic Speech Recognition and Understanding Workshop (ASRU), 2017 IEEE*. IEEE, 2017.

RNN - Transducer

- Encoder – CTC pre-trained acoustic to grapheme or word-piece prob.
- Prediction network – Pre-trained LM
- Joint network: Fully connected neural network

$$\mathbf{h}_t^{\text{enc}} = f^{\text{enc}}(x_t),$$

$$\mathbf{h}_u^{\text{dec}} = f^{\text{dec}}(y_{u-1}).$$

$$z_{t,u} = f^{\text{joint}}(\mathbf{h}_t^{\text{enc}}, \mathbf{h}_u^{\text{dec}})$$

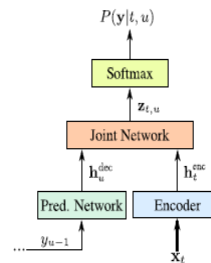


Fig. 1. The RNN-T model. The model consists of an encoder network, which maps input acoustic frames into a higher-level representation, and a prediction and joint network which together correspond to the decoder network. The decoder is conditioned on the history of previous predictions.

Inference

- If the predicted label, y_u , is non-blank, then the prediction network is updated with that label as input to generate the next output label prob $p(y|t, u+1)$.
- If a blank label is predicted then the next acoustic frame, x_{t+1} , is used to update the encoder while retain the same prediction network output.
- Alternatively, updating the encoder and the prediction network (LM)

Pre-training

- Grapheme based model: pre-training helps improving the WER (13.9% to 13.2% for voice search, 8.4% to 8.0% for voice dictation)

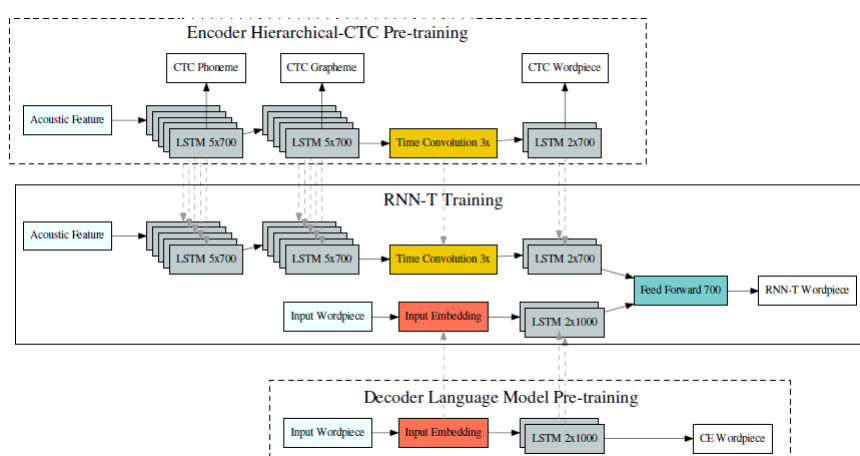
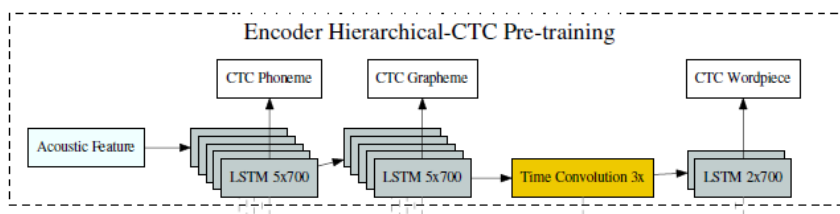


Table 1. Word error performance on the voice-search and dictation tasks for various RNN-T trained with graphemes and wordpieces with various architectures and pre-training. Also shown for each model is which types of training data are included: acoustic, pronunciation or text. The baseline is a state-of-the-art conventional speech recognition system with separate acoustic, pronunciation and language models trained on all available data. The parameters for the baseline system include 20 million weights from the acoustic model network, 0.2 million for each word in the pronunciation dictionary and the 100 million n-grams in the language model.

in the language model.										
Units	Layers		Pre-trained		Training Data Used				WER(%)	
	Encoder	Decoder	Encoder	Decoder	Acoustic	Pronunciation	Text	Params	VS	IME
RNN-T										
Graphemes	5x700	2x700	no	no	yes	no	no	21M	13.9	8.4
Graphemes	5x700	2x700	yes	no	yes	no	no	21M	13.2	8.0
Graphemes	8x700	2x700	yes	no	yes	no	no	33M	12.0	6.9
Graphemes	8x700	2x700	yes	no	yes	yes	no	33M	11.4	6.8
Graphemes	8x700	2x700	yes	yes	yes	yes	yes	33M	10.8	6.4
Wordpieces-1k	12x700	2x700	yes	yes	yes	yes	yes	55M	9.9	6.0
Wordpieces-10k	12x700	2x700	yes	yes	yes	yes	yes	66M	9.1	5.3
Wordpieces-30k	12x700	2x1000	yes	yes	yes	yes	yes	96M	8.5	5.2
Baseline										
-	-	-	-	-	yes	yes	yes	120.2M	8.3	5.4



Why RNN-T from a product point of view

- Single deployable non modularized neural model, all components of ASR in one model.
- Allows compact on-device streaming ASR. Does not need a decoder graph which can be large. Unlimited words in vocab. Standard on-device ASR choice across industry.



An All Neural On-Device Speech Recognizer
Published: March 12, 2019
Presented by: Johnathan Huh, Google Fellow, Speech Team

Image from <https://ai.googleblog.com/2019/03/an-all-neural-on-device-speech.html>

- Achieves comparable accuracy and compute with much smaller size model compared to modularized (hybrid) systems for production when training data is the same.

Table 3. Comparison of Hybrid model with RNN-T

Test Set	System	WER	Throughput	rtf@40
vid-clean	hybrid	14.0	55	.70
vid-clean	RNN-T	14.0	63	.60
vid-noisy	hybrid	20.7	55	.71
vid-noisy	RNN-T	21.0	65	.60

Example Study

Image from Jain et al., "RNN-T For Latency Controlled ASR With Improved Beam Search"

https://www.cc.gatech.edu/classes/AY2021/cs7643_spring/assets/L24_rnnt_asr_tutorial_gt.pdf



3. Listen Attend and Spell

- Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." *arXiv preprint arXiv:1409.0473* (2014).
- Chan, W., Jaitly, N., Le, Q., & Vinyals, O. (2016, March). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on* (pp. 4960-4964). IEEE. (Google Brain)



Review - Neural machine translation

NEURAL MACHINE TRANSLATION BY JOINTLY LEARNING TO ALIGN AND TRANSLATE

Dzmitry Bahdanau
Jacobs University Bremen, Germany

Kyunghyun Cho Yoshua Bengio*
Université de Montréal

- 음성인식도 일종의 번역
 - 음성(encoder) -> text (decoder)
 - 음성은 text 에 비해서 엄청 길다 (frame 수가 많다)
 - Alignment 가 중요 (attention 필요)

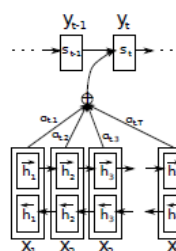


Figure 1: The graphical illustration of the proposed model trying to generate the t -th target word y_t given a source sentence (x_1, x_2, \dots, x_T) .



The context vector c_i is, then, computed as a weighted sum of these annotations h_j :

$$c_i = \sum_{j=1}^{T_x} \alpha_{ij} h_j. \tag{5}$$

The weight α_{ij} of each annotation h_j is computed by

$$\alpha_{ij} = \frac{\exp(e_{ij})}{\sum_{k=1}^{T_x} \exp(e_{ik})}, \tag{6}$$


where

$$e_{ij} = a(s_{i-1}, h_j)$$

is an *alignment model* which scores how well the inputs around position j and the output at position i match. The score is based on the RNN hidden state s_{i-1} (just before emitting y_i , Eq. (4)) and the j -th annotation h_j of the input sentence.

We parametrize the alignment model a as a feedforward neural network which is jointly trained with all the other components of the proposed system. Note that unlike in traditional machine translation,

tration of the \hat{c} model trying to generate the t -th target word y_t given a source sentence (x_1, x_2, \dots, x_T) .

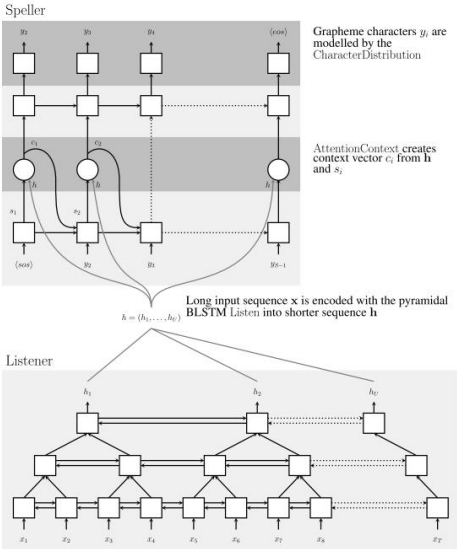


Listen, Attend and Spell

Conditional RNN

Attention

Pyramidal RNN (3):
1/8로 축소



Speller


Grapheme characters y_i are modelled by the CharacterDistribution.

AttentionContext creates context vector c_i from h and s_i .

Long input sequence x is encoded with the pyramidal BLSTM Listen into shorter sequence h .

Listener

Figure 1: Listen, Attend and Spell (LAS) model: the listener is a pyramidal BLSTM encoding our input sequence x into high level features h , the speller is an attention-based decoder generating the y characters from h .



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

Listener

- Listener compresses the input \mathbf{x} by using pyramid bi-directional RNN (LAS is basically bi-directional)
- Pyramid structure conducts time domain scaling for complexity reduction (up to $1/32$)

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

Long input sequence \mathbf{x} is encoded with the pyramid BLSTM Listen into shorter sequence \mathbf{h}

Listener

Attend and Spell

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

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

$$P(y_i|\mathbf{x}, y_{<i}) = \text{CharacterDistribution}(s_i, c_i)$$

CharacterDistribution is an MLP with softmax

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

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

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

Speller

Long input sequence \mathbf{x} is encoded with the pyramid BLSTM Listen into shorter sequence \mathbf{h}

Grapheme characters y_i are modelled by the CharacterDistribution

AttentionContext creates context vector c_i from \mathbf{h} and s_i

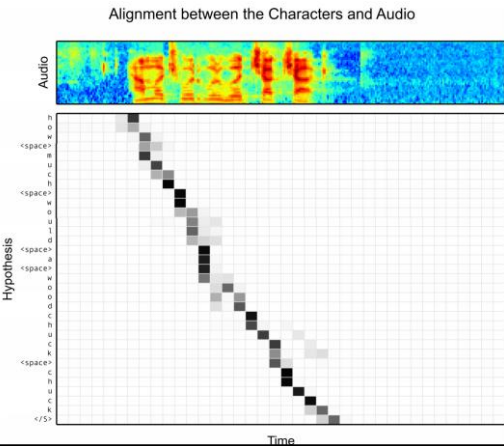
At each time step, i , the attention mechanism, AttentionContext generates a context vector, c_i encapsulating the information in the acoustic signal needed to generate the next character. The attention model is content based - the contents of the decoder state s_i are matched to the contents of h_u representing time step u of \mathbf{h} , to generate an attention vector α_i . α_i is used to linearly blend vectors h_u to create c_i .

Specifically, at each decoder timestep i , the AttentionContext function computes the scalar energy $e_{i,u}$ for each time step u , using vector $h_u \in \mathbf{h}$ and s_i . The scalar energy $e_{i,u}$ is converted into a probability distribution over time steps (or attention) α_i using a softmax function. This is used to

Listen, Attend and Spell

- Not state-of-the-art
 - Comparable results.
- Less Beam search
 - No dictionary.
 - Smaller beam width (32)
 - Decoding does not need LV
 - Attention complexity is proportional to the input length

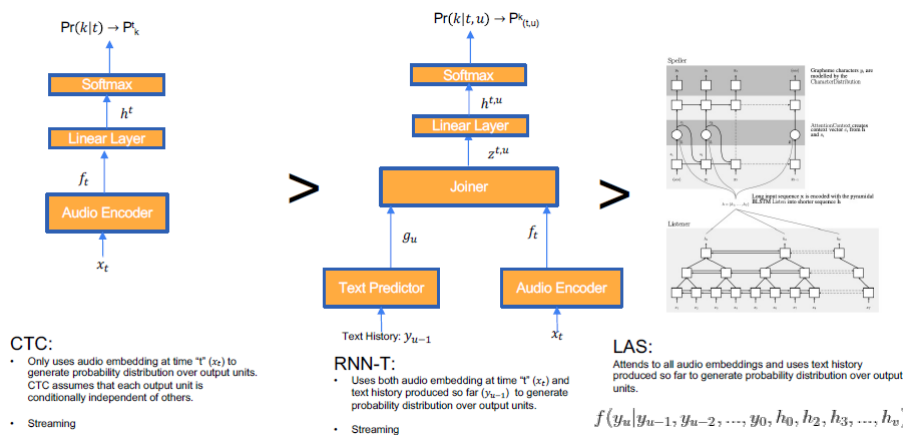
Model	Clean WER	Noisy WER
CLDNN-HMM [20]	8.0	8.9
LAS	16.2	19.0
LAS + LM Rescoring	12.6	14.7
LAS + Sampling	14.1	16.5
LAS + Sampling + LM Rescoring	10.3	12.0



Comparison of CTC, RNN-T, and LAS



Conditional independence assumption



EXPLORING NEURAL TRANSDUCERS FOR END-TO-END SPEECH RECOGNITION

Eric Battenberg, Jitong Chen, Rewon Child, Adam Coates, Yashesh Gaur, Yi Li, Hairong Liu, Sanjeev Satheesh, David Seetapun, Anuroop Sriram, Zhenyao Zhu
Baidu Silicon Valley AI Lab

- Battenberg, Eric, et al. "Exploring neural transducers for end-to-end speech recognition." *arXiv preprint arXiv:1707.07413* (2017 July).

CTC, Neural Transducer, LAS Comparison

- Conditional independence between predictions at different time steps, given audio. (-) CTC makes this assumption, but (+) RNN-Transducers and Attention models do not.
- The alignment between input and output units is monotonic. A reasonable assumption for the ASR task, which enables models to do streaming transcription. (+) CTC and RNN-Transducers make this assumption, but (-) Attention models do not.
- Hard vs Soft alignments. the attention mechanism models a soft alignment between each output step and every input step.
- CTC – insufficient model, Attention – surplus model



Fisher SWBD results

Architecture		SWBD WER	CH WER
Published	Iterated-CTC [29]	11.3	18.7
	BLSTM + LF MMI [21]	8.5	15.3
	LACE + LF MMI ⁴ [28]	8.3	14.8
	Dilated convolutions [25]	7.7	14.5
	CTC + Gram-CTC [17]	7.3	14.7
	BLSTM + Feature fusion[23]	7.2	12.7
Ours	CTC [17]	9.0	17.7
	RNN-Transducer		
	Beam Search NO LM	8.5	16.4
	Beam Search + LM	8.1	17.5
	Attention		
	Beam Search NO LM	8.6	17.8
	Beam Search + LM	8.6	17.8

- LM is paired with the dataset (not text only) (it does not improve much)
- Attention and RNN-Transducer both use a beam width of 32.



DeepSpeech Corpus

- 10,000 hours in the diverse set of scenarios, such as far field, noise accents

Model	Dev	Test
CTC [4]		
Greedy decoding	23.03	-
Beam search + LM (beam=2000)	15.9	16.44
RNN-Transducer		
Greedy decoding	18.99	-
Beam search (beam=32)	17.41	-
+ LM rescoring	15.6	16.50
Attention		
Greedy decoding	22.67	-
Beam search (beam=256)	18.71	-
+ Length-norm weight	19.5	-
+ Coverage cost	18.9	-
+ LM rescoring	16.0	16.48

Table 3. Comparison of WER obtained by different transduction models on the DeepSpeech dataset which has a mismatch between training and test distributions.



Forward only encoders

- For streaming applications, forward only encoders are needed.
- Replacing the bidirectional layers with forward-only recurrent layers. Note that while this immediately makes CTC and RNN-Transducer models deployable, attention models still need to be able to process the entire utterance before outputting the first character.

formance or improves training. In our experiment, we replace every layer of 256 bidirectional LSTM cells in the encoder with a layer of 512 forward-only LSTM cells.

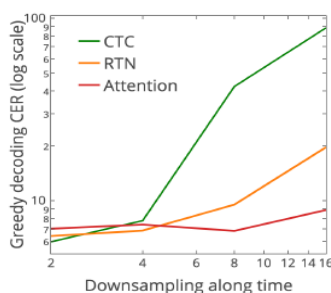
Model Decoding	Bidirectional		Forward-only Beam + LM
	Greedy No LM	Beam + LM	
CTC	15.73	10.08	13.78
RNN-Transducer	15.29	14.05	22.38
Attention	14.99	14.07	19.19

Table 5. WER of baseline models on WSJ eval'92 set. On smaller datasets, RNN-Transducers and Attention models do not have enough data to learn a good implicit language model and therefore perform poorer compared to CTC even after rescoring with an external LM (RNN-Transducers and Atten-



Down-sampling in the encoder

- Down sampling reduces the arithmetic/memory complexity.
- Since RNN-Transducers and attention models can output multiple characters for the same encoder time step, we expect RNN-Transducers to be as robust as attention models as we increase the amount of pooling in the encoder. While Figure 2 shows that they are fairly robust compared the CTC models, we find that attention models are significantly more robust.



2. Effect of increasing the frame-rate on WER

Comparison of CTC, RNN-T, and Attention models.

- In the bidirectional setting, all three models perform roughly the same. In forward direction only, CTC best.
- CTC - simplify the training process but still require to be decoded with large language models. Large beam width.
- RNN Transducers have simple decoding process with no extra hyper-parameters tuning, which leads us to believe that RNN-Transducers present the next generation of end-to-end speech models.
- Attention also simplify the decoding process and require the language models to be introduced only in a post processing stage to be equally if not more effective. Strength in down sampling.