Advancing RNN Transducer Technology for Speech Recognition

George Saon, Zoltán Tüske, Daniel Bolanos and Brian Kingsbury

IBM Research AI

International Conference on Acoustics, Speech and Signal Processing (ICASSP)
Toronto, June 6-11 2021

Outline

- Introduction
- Model description
- Multiplicative integration
- Training and decoding recipe
- Experiments on Switchboard 300h, conversational Spanish and Italian
- Conclusion

Introduction

- End-to-end approaches directly map an acoustic feature sequence to a sequence of characters or words without any conditional independence assumptions
- Compared to traditional approaches, E2E ASR has dramatically simplified both training and decoding pipelines
- E2E outperforms traditional ASR on a range of tasks in both high [Chiu'18] and low data regimes [Tueske'20]
- RNN-Ts [Graves'12] are excellent at handling monotonic alignment problems which is useful for streaming ASR
- Because of that, RNN-Ts are becoming the modeling paradigm of choice for many sites

Model description

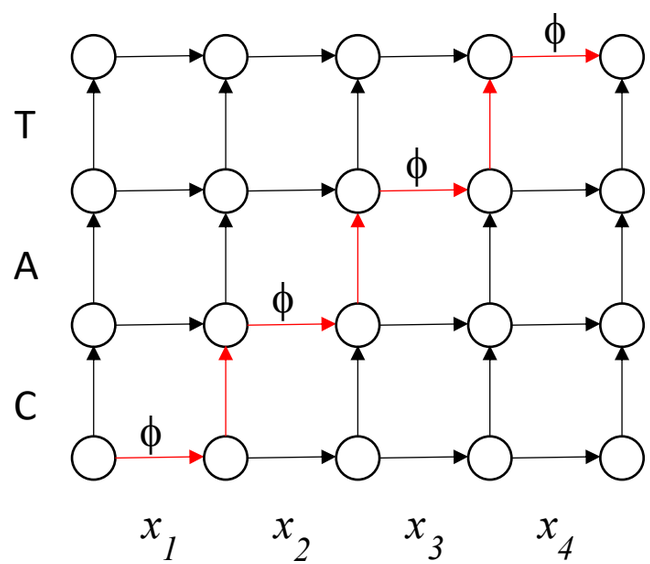
- $\mathbf{y} = (y_1, \ldots, y_U) \in \mathcal{Y}^*$ *output sequence* (typically characters, words or morphemes)
- $\mathbf{x} = (x_1, \ldots, x_T) \in \mathcal{X}^*$ *input sequence* (acoustic frames extracted from speech signal)
- RNN-T models conditional distribution of $\mathbf{y}$ given $\mathbf{x}$

$$p(\mathbf{y}|\mathbf{x}) = \sum_{\mathbf{a} \in \mathcal{B}^{-1}(\mathbf{y})} p(\mathbf{a}|\mathbf{x})$$

- $\mathbf{a} = (a_1, \ldots, a_{T+U})$ is an *alignment sequence*. $a_i \in \overline{\mathcal{Y}} = \mathcal{Y} \cup \{\phi\}$ belong to the augmented output space where ϕ (called BLANK) denotes the null output
- $\mathcal{B} : \overline{\mathcal{Y}}^* \rightarrow \mathcal{Y}^*$ strips the BLANKs from $\mathbf{a}$ such that $\mathcal{B}(\mathbf{a}) = \mathbf{y}$

Example

- $\mathbf{y} = (C, A, T)$
- $\mathbf{x} = (x_1, x_2, x_3, x_4)$
- Valid alignments are: $(\phi, C, \phi, A, \phi, T, \phi)$, $(\phi, \phi, \phi, \phi, C, A, T)$, $(C, A, T, \phi, \phi, \phi, \phi)$ ...[1]



[1] There are $\binom{T+U}{U} = \frac{(T+U)!}{T!U!} \geq \left(1 + \frac{T}{U}\right)^U$ possible alignments.

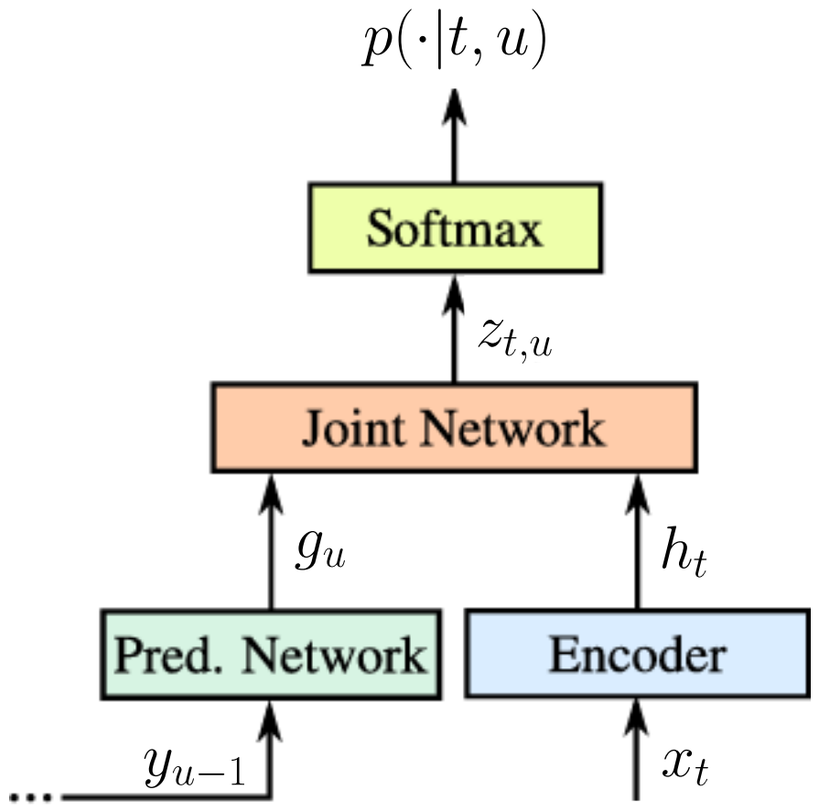
Alignment probability

$$p(\mathbf{a}|\mathbf{x}) = \prod_{i=1}^{T+U} p(a_i|h_{t_i}, g_{u_i})$$

- $\mathbf{h} = (h_1, \ldots, h_T) = Encoder(\mathbf{x})$ is an embedding of the input sequence computed by an encoder network
- $\mathbf{g} = (g_1, \ldots, g_U)$ is an embedding of the output sequence computed by a prediction network via the recursion $g_u = Prediction(g_{u-1}, y_{u-1})$
- $p(a_i|h_{t_i}, g_{u_i})$ is the predictive output distribution over $\overline{\mathcal{Y}}$ computed by a joint network commonly implemented as:

$$z_{t,u} = \mathbf{W}^{out}\tanh(\mathbf{W}^{enc}h_t + \mathbf{W}^{pred}g_u + b),$$
$$p(\cdot|t,u) = \mathrm{softmax}(z_{t,u}),\ t = 1 \ldots T,\ u = 1 \ldots U$$

RNN-T architecture



Multiplicative integration in joint network

$$z_{t,u} = \mathbf{W}^{out}\tanh(\mathbf{W}^{enc}h_t \odot \mathbf{W}^{pred}g_u + b)$$

- Introduces second order interactions between the elements of h_t and g_u
- Includes the additive terms when used in conjunction with biases:

$$(\mathbf{W}^{enc}h_t + b^{enc}) \odot (\mathbf{W}^{pred}g_u + b^{pred}) = (\mathbf{W}^{enc}h_t) \odot (\mathbf{W}^{pred}g_u) + b^{enc} \odot \mathbf{W}^{pred}g_u + b^{pred} \odot \mathbf{W}^{enc}h_t + b^{enc} \odot b^{pred}$$

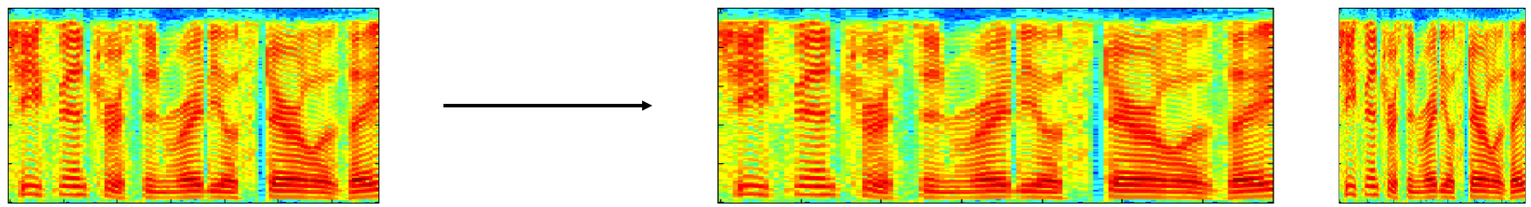
- One embedding has a gating effect on the gradient of the loss w.r.t. the other embedding i.e.

$$\frac{\partial \mathcal{L}}{\partial h_t} = g_u \odot \frac{\partial \mathcal{L}}{\partial (h_t \odot g_u)}$$
$$\frac{\partial \mathcal{L}}{\partial g_u} = h_t \odot \frac{\partial \mathcal{L}}{\partial (h_t \odot g_u)}$$

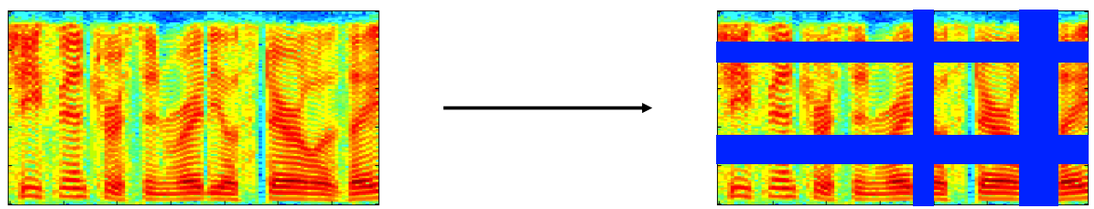
- Shown to improve RNN performance by [Wu et al.'16]

Training recipe: data augmentation/perturbation

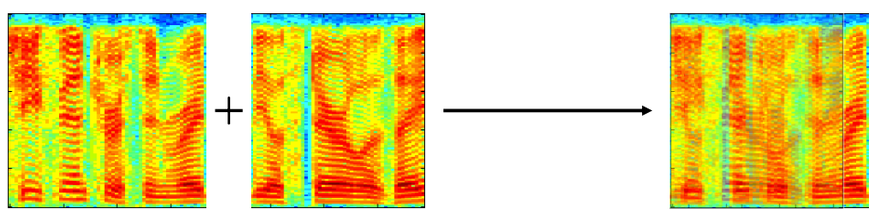
- Speed and tempo perturbation [Ko et al.'15]



- SpecAugment [Park et al.'19]



- Sequence noise injection (SNI) [Saon et al.'19]



Other techniques

- **CTC pretraining**: initialize the encoder with a model trained using CTC
- **Dropconnect**: zeros out entries randomly in the LSTM hidden-to-hidden transition matrices
- **Switchout**: Replace characters with probability $\hat{n}/U$ where $\hat{n} \in \{0, \ldots, U\}$ is sampled with $p(\hat{n}) \propto e^{-\hat{n}/\tau}$ [Wang'18]
- **I-vector speaker adaptation**: appends a speaker identity vector to the input features [Saon'13]
- **Density ratio LM fusion**: subtracts the effect of the inbuilt LM given by the prediction network [McDermott'19]

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{Y}^*}{\operatorname{argmax}} \{\log p(\mathbf{y}|\mathbf{x}) - \mu \log p^{src}(\mathbf{y}) + \lambda \log p^{ext}(\mathbf{y}) + \rho|\mathbf{y}|\}$$

where μ, λ, ρ are the weights corresponding to the source LM p^{src}, external LM p^{ext}, and label length reward $|\mathbf{y}|$.

Switchboard 300 h experiments

- Architecture:
  - Encoder: 6 BiLSTM layers with 640 cells per layer per direction
  - Prediction network: 1 unidir LSTM with 768 cells
  - Joint network: projects both embeddings to 256 dim, combines them then applies `tanh`
  - Output layer: joint vector is projected to 46 dim followed by `softmax` (45 characters + BLANK)
- Features:
  - 40-dim SI log-Mel 10ms frames mean and variance normalized per speaker, augmented with Δ and $\Delta\Delta$ coefficients
  - Every two consecutive frames are stacked and every other stacked frame is skipped
  - Features are augmented with 100-dim i-vectors $\Rightarrow$ 340-dim vectors every 20 ms
- Training:
  - 20 epochs of AdamW with one cycle learning rate policy [Smith & Topin'17]
  - Curriculum learning with utterances sorted by increasing length (unstable sort)

Ablation study

	WER SWB	WER CH	Average
No Speed/tempo	7.6	15.2	11.4
No SpecAugment	6.9	15.5	11.2
No DropConnect	7.0	14.8	10.9
No Density ratio	6.9	14.4	10.7
No CTC encoder init	6.6	14.1	10.4
No I-vectors	6.6	13.9	10.3
No SNI	6.6	13.8	10.2
No Switchout	**6.3**	**13.1**	**9.7**
Baseline	6.4	13.4	9.9

(SWB = Switchboard testset, CH = CallHome testset)

Additive and multiplicative model combination

- Log-linear score combination with density ratio LM fusion:

$$\mathbf{y}^* = \underset{\mathbf{y} \in \mathcal{H}_+(\mathbf{x}) \cup \mathcal{H}_\odot(\mathbf{x})}{\operatorname{argmax}} \{\alpha \log p(\mathbf{y}|\mathbf{x}; \theta_+) + \beta \log p(\mathbf{y}|\mathbf{x}; \theta_\odot) - \mu \log p^{src}(\mathbf{y}) + \lambda \log p^{ext}(\mathbf{y}) + \rho|\mathbf{y}|\}$$

where $\mathcal{H}_+(\mathbf{x}), \mathcal{H}_\odot(\mathbf{x})$ are the n-best hypotheses generated by the additive and multiplicative RNN-Ts.

Model	No ext LM			With ext LM		
	SWB	CH	Avg	SWB	CH	Avg
$+$	8.0	15.6	11.8	6.4	13.4	9.9
$\odot$	8.1	15.5	11.8	6.3	13.1	9.7
Combined	7.5	14.4	11.0	**5.9**	**12.5**	**9.2**

Experiments on conversational Spanish 780 h

- Internal call center recordings of Castillian and latin American dialects from 4000 speakers
- Similar setup to SWB 300h except for different output layer and no i-vectors
- Test set contains 4 hours of speech, 113 speakers and 31.6k words

Model/technique	Training data	WER
Initial experiment	250 h	34.8
+ DropConnect+seq. noise	250 h	27.6
+ Speed/tempo	780 h	25.0
+ CTC encoder pretraining	780 h	23.6
+ Multiplicative integration	780 h	22.7
+ SpecAugment	780 h	21.9
+ AdamW+OneCycleLR	780 h	20.8
+ Shallow LM fusion	780 h	20.3
+ Density ratio LM fusion	780 h	20.0

Experiments on conversational Italian 900 h

- Scripted dialogs from several domains such as banking, insurance, telco, retail with a balanced demographic and dialectal distribution
- Similar setup to SWB 300h except for different output layer and no i-vectors
- Test set from Mozilla CommonVoice[2] Italian corpus with 1.2 hours of audio, 764 utterances and 6.6K reference words

	Bidirectional		Unidirectional	
	$+$	$\odot$	$+$	$\odot$
No external LM	17.8	17.6	28.0	26.2
Shallow fusion	15.2	13.9	22.9	21.4
Density ratio fusion	13.6	12.7	20.9	18.6

[2] https://commonvoice.mozilla.org

Conclusion

- Multiplicative integration of acoustic and prediction network embeddings outperforms additive integration and shows good model complementarity
- Careful application of data augmentation/perturbation, model regularization, speaker adaptation, external LM fusion, and model combination leads to excellent performance when applied to RNN-Ts with a very simple architecture and character outputs
- Improving the architecture and training recipe for other neural transducers (e.g. conformer transducers) is currently underway