Deep Learning for Audio

Lecture 3

Pavel Severilov

Moscow Institute of Physics and Technology

2023

Outline

1. RNN-Transducer (RNN-T)
2. Language models for ASR
3. Byte-pair encoding (BPE)
4. State-of-the-art ASR

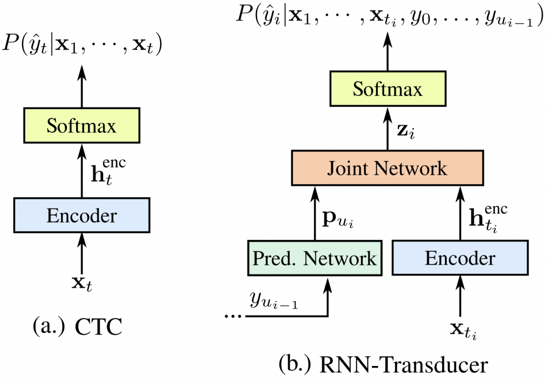
Outline

1. RNN-Transducer (RNN-T)
2. Language models for ASR
3. Byte-pair encoding (BPE)
4. State-of-the-art ASR

CTC and Attention models: recap

	CTC	**Listen, Attend and Spell: LAS**	**?**
Summary	Maximize probability of all possible CTC-paths leading to target.	Encoder-decoder architecture with attention.	???
Online	+	-	+
Context dependent	-	+	+
Multiple outputs for each input	-	-	+

RNN-T: idea



- Predictor is autoregressive: takes as input the previous outputs.
- Joiner – feedforward network, combines the encoder vector h_t and predictor vector p_u

He et al. Streaming End-to-end Speech Recognition for Mobile Devices / 2019, Google, Inc.

RNN-T: model

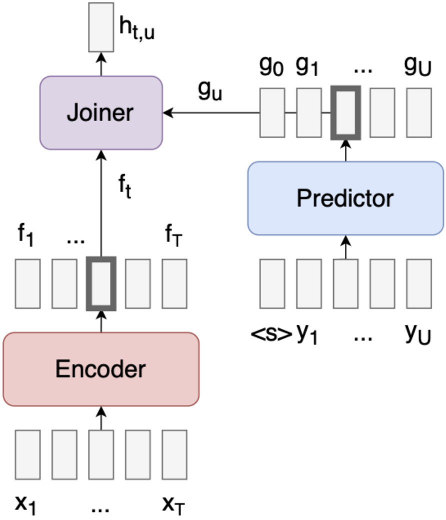


Figure: RNN-T architecture

Sequence-to-sequence learning with Transducers blog post

RNN-T: model

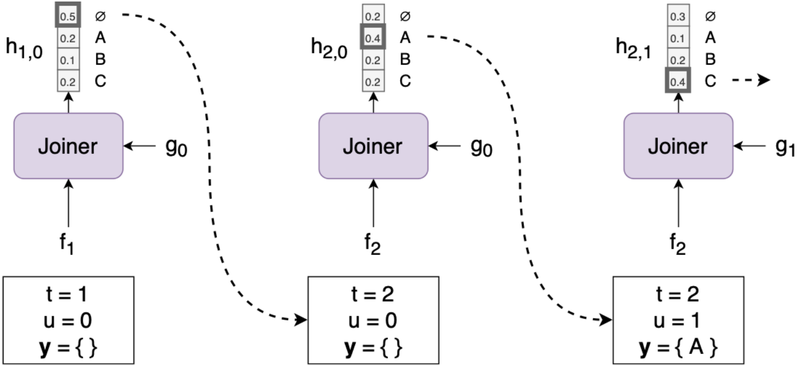


Figure: Steps example of RNN-T inference: t – audio-encoder timestamp, u – Predictor (char network) step

Sequence-to-sequence learning with Transducers blog post

RNN-T: training

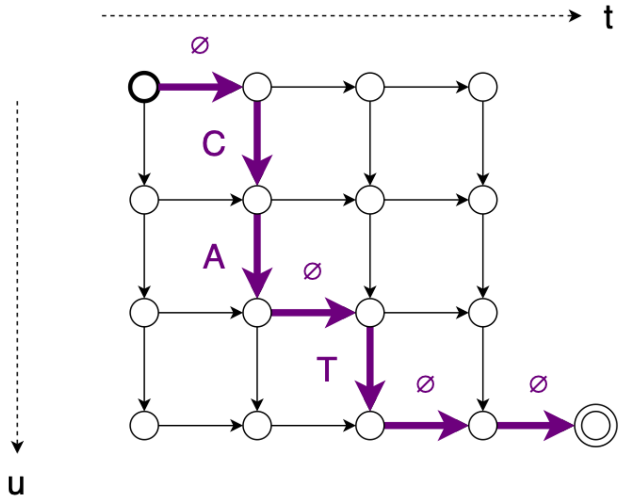


Figure: Alignment $\{\emptyset, C, A, \emptyset, T, \emptyset, \emptyset\}$ for input sequence of length $T = 4$ and an output sequence "CAT" of length $U = 3$

Sequence-to-sequence learning with Transducers blog post

RNN-T: training

We need to get $p(\mathbf{y}|\mathbf{x})$ as the sum of the probabilities of all possible alignments between $\mathbf{x}$ and $\mathbf{y}$

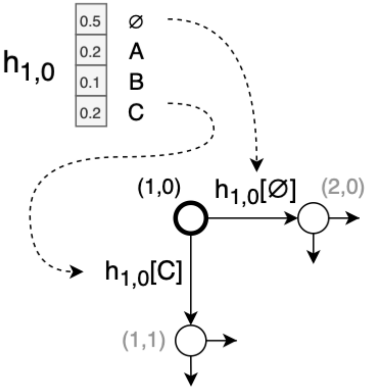


Figure:

$\mathbf{z} = \varnothing, C, A, \varnothing, T, \varnothing, \varnothing$

$\underline{p(\mathbf{z} \mid \mathbf{x}) = h_{1,0}[\varnothing] \cdot h_{2,0}[C]} \cdot h_{2,1}[A] \cdot h_{2,2}[\varnothing] \cdot h_{3,2}[T] \cdot h_{3,3}[\varnothing] \cdot h_{4,3}[\varnothing]$

Sequence-to-sequence learning with Transducers blog post

RNN-T: training

Objective: minimize the loss function $-log\, p(y|x)$

To compute the sum efficiently, compute $\alpha_{t,u}$, for $1 \leq t \leq T$ and $0 \leq u \leq U$

$$\alpha_{t,u} = \alpha_{t-1,u} \cdot h_{t-1,u}[\varnothing] \\ +\alpha_{t,u-1} \cdot h_{t,u-1}\,[y_{u-1}]$$

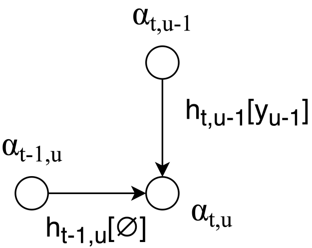


Figure: Computing $\alpha_{t,u}$ to get $p(\mathbf{y} \mid \mathbf{x}) = \alpha_{T,U} \cdot h_{T,U}[\varnothing]$

Sequence-to-sequence learning with Transducers blog post

Losses in ASR: summary

- CTC Loss
  - Pros: output can be computed parallel, very fast, goes well with LMs
  - Cons: no context dependences; often LMs are big and slow (can't do on devices)
- CrossEntropyLoss (Autoregressive Encoder-Decoder, LAS)
  - Pros: High Quality because of autoregression, often has attention (good for seq-to-seq tasks)
  - Cons: hard inference, hard to do in production (slow), not online
- RNN-T Loss (Transducer)
  - Pros: can be online, better quality because of autoregression, often used in devices
  - Cons: hard inference, hard to build in attention, need a lot of GPUs to train, output are computed not in parallel

Outline

Language models (LM): why need in ASR?

- Language models recap: a model that estimates the probability of a text.
  - N-gramms
  - Neural networks (BERT, GPT-3, ...)
  - Example:
    $P(\text{let's go two a movie}) = 0.01$
    $P(\text{let's go to a movie}) = 0.6$
- ASR problem:
  - Spelling of a word heavily depends on its context
  - Labeled audio data is difficult to obtain
- How LM helps:
  - Improves final WER
  - Improves performance for small audio datasets
  - Can be used to adapt model to new domain

LM: how to integrate in ASR?

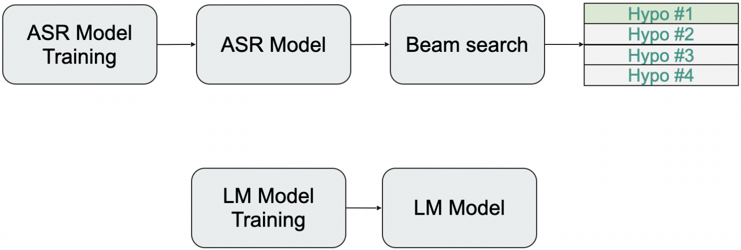


Figure: ASR pipeline VS Language models pipeline

LM: final hypothesis rescoring

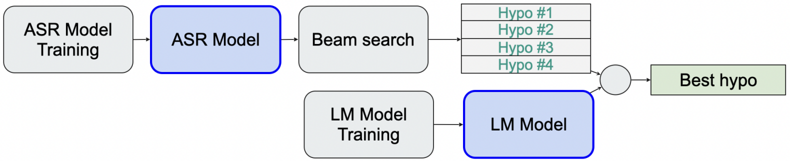


Figure: Final hypothesis rescoring: rescore beam-search output with LM probs

$$\boldsymbol{y}^* = \arg\max_{\boldsymbol{y}} \log p(\boldsymbol{y} \mid \boldsymbol{x}) + \lambda \log p_{LM}(\boldsymbol{y}) + \beta \cdot \mathrm{len}(\boldsymbol{y})$$

$\mathrm{len}(\boldsymbol{y})$ – function of word length, anti-penalty for long words

LM: shallow fusion

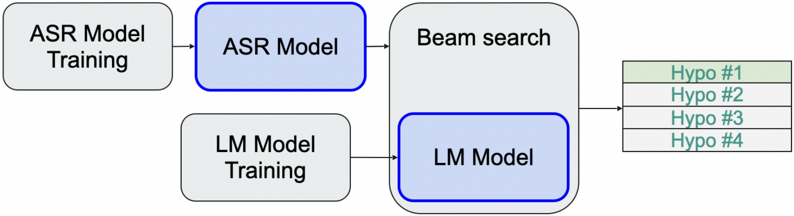


Figure: Shallow fusion: use LM rescoring after each beam search step

Practice:

- requires much more LM runs
- use light LM for shallow fusion
- use heavy LM for second-pass rescoring

LM: Deep fusion

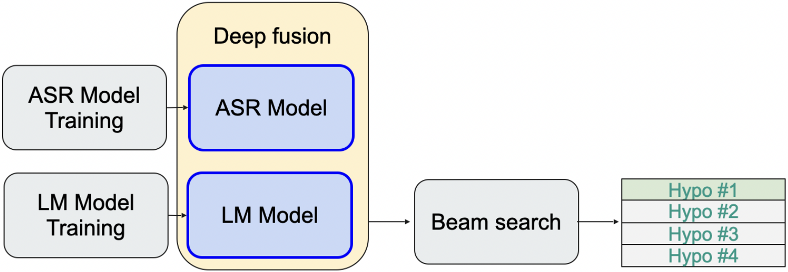


Figure: Deep fusion: integrates the external LM into the encoder-decoder model (ASR) by fusing together the hidden states of the external LM and the decoder

---

Toshniwal, Shubham et al. "A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition." 2018 IEEE

LM: Cold fusion

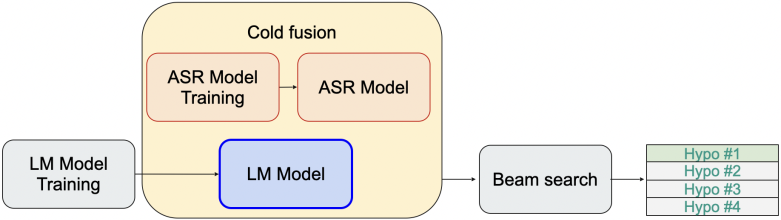


Figure: Cold fusion: train like in Deep fusion, but jointly with ASR model

Toshniwal, Shubham et al. "A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition." 2018 IEEE

LM in ASR: comparison of approaches

Model	SWB	CH	Full
LAS	17.1	27.9	22.6
Shallow Fusion	**15.6**	**26.6**	**21.1**
Deep Fusion	16.3	27.2	21.7
Cold Fusion	16.3	27.3	21.8

Table: Word error rates (%) on Eval2000 for the LAS baseline model and fusion approaches. SWB=Switchboard, CH=CallHome, Full=Eval2000.

Toshniwal, Shubham et al. "A Comparison of Techniques for Language Model Integration in Encoder-Decoder Speech Recognition." 2018 IEEE

Outline

1. RNN-Transducer (RNN-T)
2. Language models for ASR
3. Byte-pair encoding (BPE)
4. State-of-the-art ASR

BPE: motivation & idea

- Motivation: a lot of characters have different pronunciation in different contexts
- Idea: let's use n-gramms as tokens in addition
- Advantages:
  - Less decoder steps $\rightarrow$ faster training and inference
  - Better generalization $\rightarrow$ better WER

BPE: algorithm

1. Each character – token
2. Most popular n-gram: add new token
3. Replace n-gram with a new token
4. Restrict maximum length of tokens
5. New vocabulary = all characters + new tokens

Iteration	Sequence	Vocabulary
0	a b a b c a b c	{a, b, c}
1	ab ab c ab c	{a, b, c, ab}
2	ab abc abc	{a, b, c, ab, abc}
3	ababc abc	{a, b, c, ab, abc, ababc}
4	ababcabc	{a, b, c, ab, abc, ababc, ababcabc}

Table: BPE: example for sequence {ababcabc}

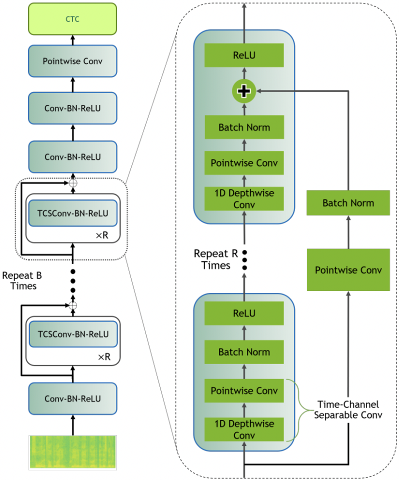
Outline

1. RNN-Transducer (RNN-T)
2. Language models for ASR
3. Byte-pair encoding (BPE)
4. State-of-the-art ASR

ASR: SOTA models

- RNN-T (2018, Google)
- MoChA (2018, Google) (Tricks to make LAS online)
- wav2vec (2019, Facebook AI Research) (SSL, use WAV not spectrograms)
- Jasper (2019, Nvidia) (Encoder: CNN ; Loss: CTC)
- QuartzNet (2019, Nvidia) (Encoder: TDS CNN ; Loss – CTC)
- ContextNet (2020, Google) (Encoders: CNN and LSTM ; Loss – RNN-T)
- wav2vec2 (2020, Facebook AI Research)
- Conformer (2020, Google) (Encoder: Convolutions + Transformers,)
- Whisper (2022, OpenAi) (Encoder: Transformer; Loss - multitask)

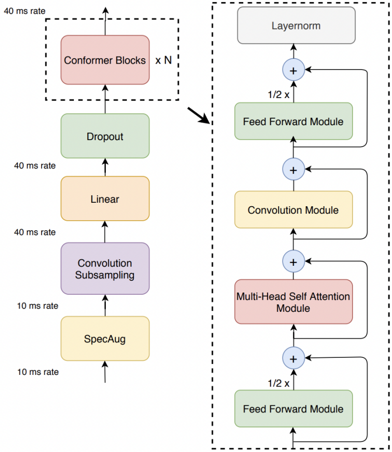
QuartzNet



- Multiple blocks with residual connections (no transformers)
- Block: 1D time-channel separable convolutional layers + batch normalization + ReLU
- Loss: CTC (can be RNN-T)
- very fast: 2500 SPS on V100 ($\approx$ 45mins in a second)

Kriman, Samuel et al. "QuartzNet: Deep Automatic Speech Recognition with 1D Time-Channel Separable Convolutions", 2020 ICASSP

Conformer



- Combines convolution and transformers blocks
- Loss: CTC, Transducer
- 1900 SPS on V100 ($\approx$ 30mins in a second)

Gulati, Anmol et al. "Conformer: Convolution-augmented Transformer for Speech Recognition", 2020 Interspeech

Whisper

- ASR system trained on 680,000 hours of multilingual and multitask data collected from the web
- Shows that large and diverse dataset leads to improved robustness to accents, background noise and technical language
- Can translate text into English
- 99 languages in train

Radford, Kim et al. "Robust Speech Recognition via Large-Scale Weak Supervision", 2022 IEEE

Whisper: training data

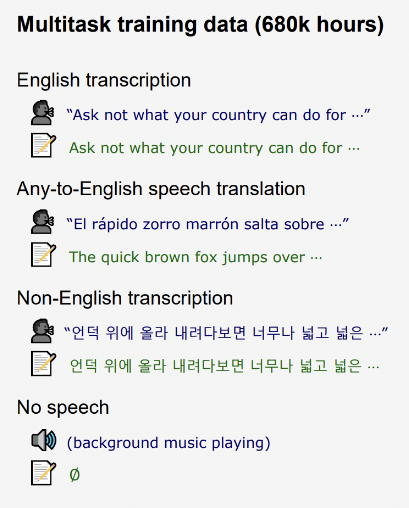
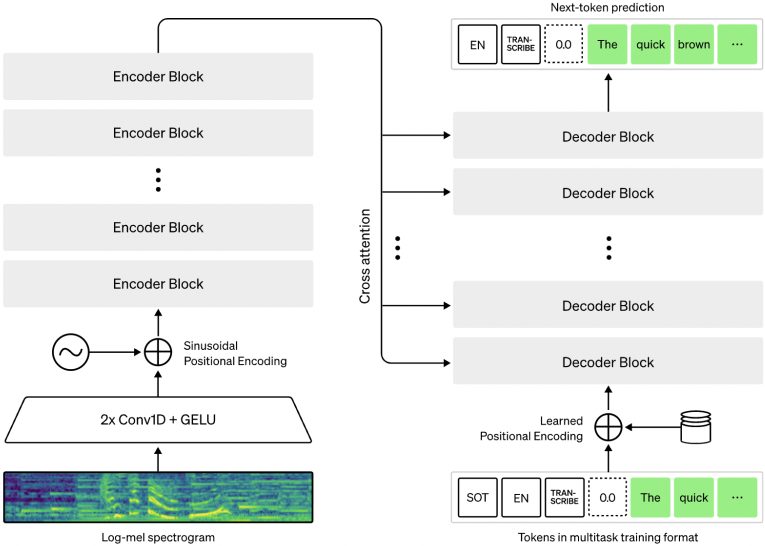
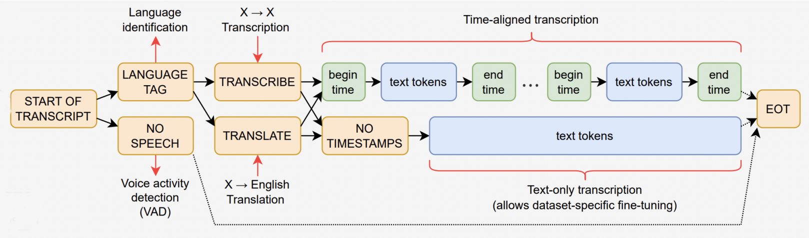


Figure: 17% Multilingual Speech Recognition (117k hours), 18% Translation (126k hours), 65% English Speech Recognition (438k hours)

Whisper: architecture



Whisper: multitask learning



Whisper: results

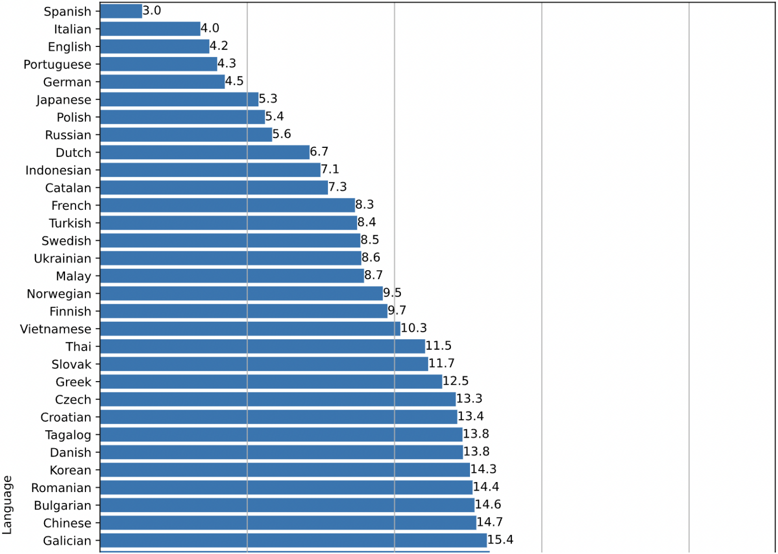


Figure: WER, %, Fleurs dataset

SOTA in production cookbook

- Use CTC/CrossEntropy/Transducer loss
- Add LMs or N-grams
- Models:
  - Quartznet: very fast, worse quality
  - Conformer: medium speed, medium quality
  - wav2vec 2.0: slow, best quality
  - Whisper: ? most robust ?
  - Ansambles

SOTA ASR: summary

Model	Average WER	LS Clean	LS Other
Human	-	5.83	12.69
Kaldi	-	8.01	22.49
DeepSpeech2 (CTC)	-	5.15	12.73
LAS	-	3.2	9.8
[PwC] QuartzNet	-	2.69	7.25
[PwC] Conformer large	-	1.9	3.9
[HF] wav2vec2-large	14.47	1.73	3.74
[HF] fastconformer_ctc_xxlarge	8.34	1.69	3.4
[HF] whisper-large-v2	8.16	2.87	5.16
[HF] fastconformer_transducer_xlarge	**8.06**	**1.5**	**2.88**

Average WER: on datasets AMI, Earnings22, Gigaspeech, ~~LibriSpeech, SPGISpeech~~, Tedlium, Voxpopuli, Common Voice

HuggingFace Open ASR Leaderboard