AMMI course on Speech Recognition

Week 1: Gabriel Synnaeve, Neil Zeghidour,
Week 2: Laurent Besacier, Emmanuel Dupoux

Logistics

- Those slides (to click on links!) at https://tinyurl.com/AMMIASR
- Recorded lectures at: https://tinyurl.com/AMMIASRyoutube
- Labs on a zoom meeting, so that we start with a Q&A on the course
- Labs on Colab, first lab here: https://colab.research.google.com/drive/1md_45aJyiksrb_K_EOzn2S5-nn6soEDN?usp=sharing
- Chat on Campuswire https://campuswire.com/c/G327BAF92/feed

Grade

- Quiz: at the end of week 2, roughly 50-60% of the grade
- Labs: roughly 40-50% of the grade
- Project (week 2+): bonus points

Structure of the Labs for Week 1

- There are basically 2 labs:
- One in PyTorch where you have to code some things from the courses -> AMMI_labs_PyTorch (they start Monday, Wednesday, Friday).
- One in C++ where you mostly just have to use existing code -> wav2letter (it starts Tuesday / Thursday).

- The one in PyTorch is a required assignment
  - Turn in an html copy of your Colab (jupyter notebook) with the outputs of all cells
- The one in C++ is bonus

Good Tutorials and Books to Read

- Speech and Language Processing, **Jurafsky and Martin**
  3rd edition draft:
  https://web.stanford.edu/~jurafsky/slp3/ed3book.pdf
- HTK Book https://www.danielpovey.com/files/htkbook.pdf

- A guide to convolution arithmetic for DL https://arxiv.org/abs/1603.07285
- CS231n: Convolutional Neural Networks for Visual Recognition
  http://cs231n.stanford.edu/

Good Papers to Read to Accompany the Course

- https://www.microsoft.com/en-us/research/wp-content/uploads/2016/02/HintonDengYuEtAl-SPM2012.pdf
- https://arxiv.org/abs/1412.5567
- https://arxiv.org/abs/1812.06864
- (Along with course 3:) https://cs.nyu.edu/~mohri/pub/hbka.pdf
- + what is mentioned in the labs

Week 1 overview

Monday (today): Introduction to Automatic Speech Recognition

- Speech tasks: ASR, KWS, TTS, speaker ID, speech enhancement, source separation...
- Why is it difficult? Spontaneous vs. read speech, large/open vocab., single word vs continuous, background noise, accents, dialects etc.
- Acoustics and Linguistics absolute basics: sound, phonemes, words.
- Brief history of speech rec.: HMM, DNN, E2E.
- ASR: sound->features->acoustic model->loss->LM (decoding)
- ASR metrics: WER/CER, inference latency, throughput, RTF

Monday’s Lab (in PyTorch):

- Q&A
- Setting up the PyTorch-based Google Colab
- Implement a ConvNet with 1D convolutions
- Visualize features
- Implement a classification training loop
- Implement WER

Tuesday’s Lab:

- Q&A
- Continue Monday’s Lab if needed
- Setup Thursday’s Lab

Wednesday: Speech Features and Acoustic Models [Neil Zeghidour]

- Q&A **11am Accra/1pm Kigali**
- Speech and speech features
- Modeling speech with Hidden Markov Models
- Modeling speech features with Gaussian Mixture Models and Neural Networks
- Handling variability in gender, speaker identity, and noise conditions
- Waveform-based acoustic modeling

Wednesday’s Lab (in PyTorch):

- Implement classic speech features
- Redo the training loop of lab 1
- Implement a very simple wav-based acoustic model

Thursday’s Lab (wav2letter, C++):

- Continue Wednesday’s lab if needed
- wav2letter (a state-of-the-art end-to-end ASR system) usage tutorial
  - How to use a pretrained model to transcribe speech
  - How to train your acoustic model in wav2letter (without writing new code)

Friday: End-to-End Training

- No alignment in most datasets
- Historically: align with multiple stages of training (HMM)
- CTC (and other sequence criterion)
- seq2seq and other end-to-end losses
- Phonemes, word pieces, words
- Language models and decoding

Friday’s Lab (back to PyTorch):

- Q&A
- Speech **recognition** for real
- Implement CTC
- Implement beam-search decoding

Course 1: Introduction to Speech Recognition

Gabriel Synnaeve

Plan

- Speech tasks: ASR, KWS, TTS, speaker ID, speech enhancement, source separation...
- Why is it difficult? Spontaneous vs. read speech, large/open vocab., single word vs continuous, background noise, accents, dialects etc.
- Acoustics and Linguistics absolute basics: sound, phonemes, words.
- Brief history of speech rec.: HMM, DNN, E2E.
- ASR: sound->features->acoustic model->loss->LM (decoding)
- ASR metrics: WER/CER, inference latency, throughput, RTF

A note about notation

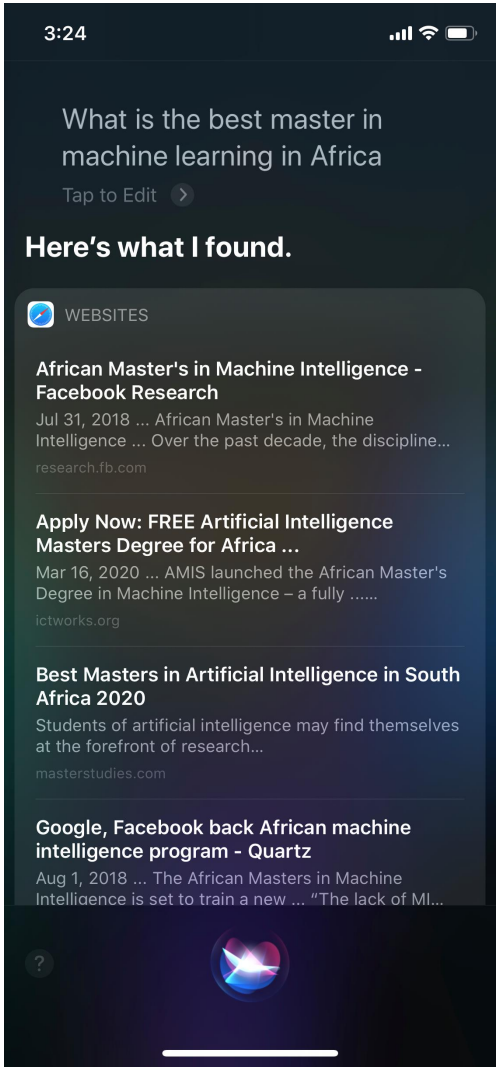
- in machine learning (supervised learning): x is the input (features), y the labels, “yhat” ($\hat{y}$) the prediction of the model.

- in Bayesian modeling: O are the input features (**observations**), S are the **states**. Discriminative models (CRFs) are P(O, S) = P(S | O)P(O), while generative models (HMMs) are P(O, S) = P(O | S)P(S).

- WER = Word Error Rate

Speech recognition: what for?

- Historically, the easiest data input/output format for computers is text.
- **Human machine interface**: dictation, assistants (e.g. Apple Siri, Amazon Echo, Google Home, etc.), OS control...
- **Transcribing** as an end goal: e.g. voice messages transcription and automatic video captioning.
- **Indexing**: search of content
- **Accessibility**: (e.g. transcription for hearing impaired, or dictation for motor impairments).

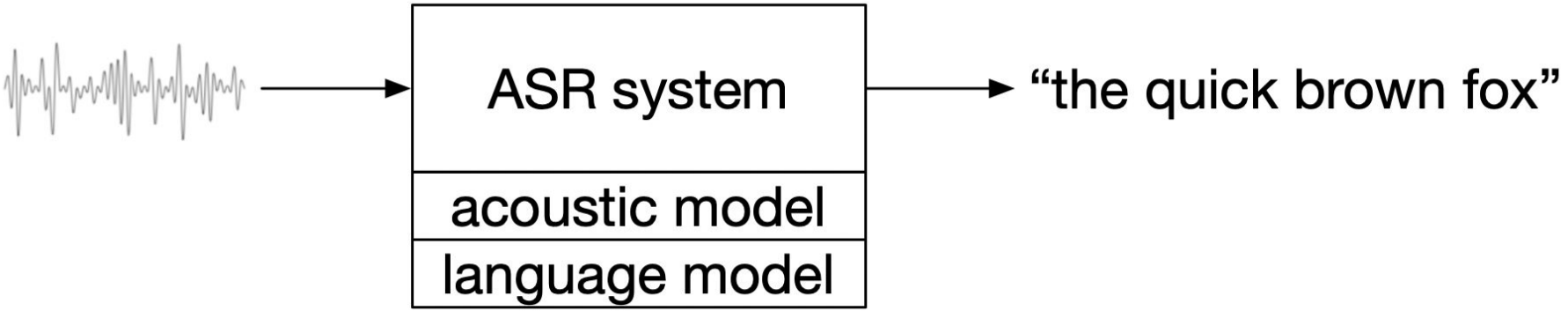


Human machine interface:

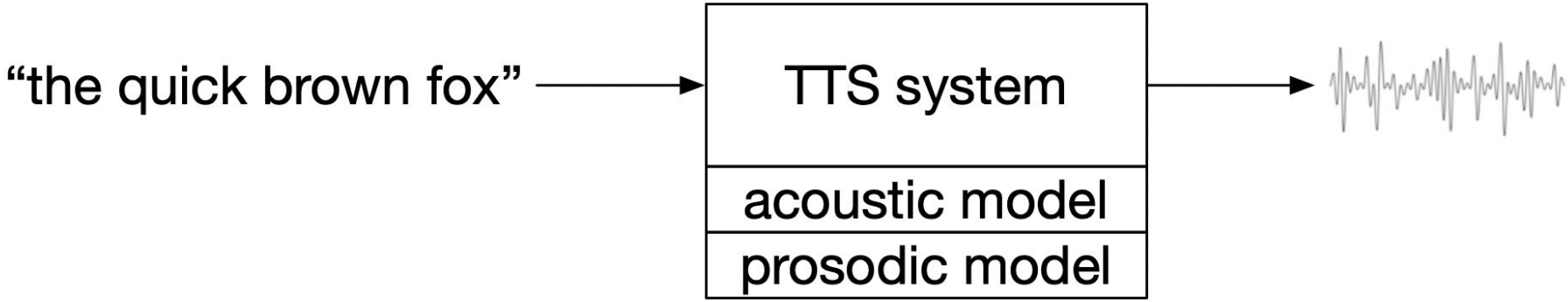
- dictation,
- assistants (e.g. Apple Siri, Amazon Echo, Google Home, etc.),
- OS control...

Transcribing as an end goal: e.g. voice messages transcription and automatic video captioning, accessibility.

ASR (automatic) speech rec.
(a.k.a. STT, speech to text)



TTS, text to speech, a.k.a. speech synthesis



- Generating speech from text (there is also STS).
- Useful for human machine interface, accessibility (e.g. screen readers), assistive technology, art (singing).

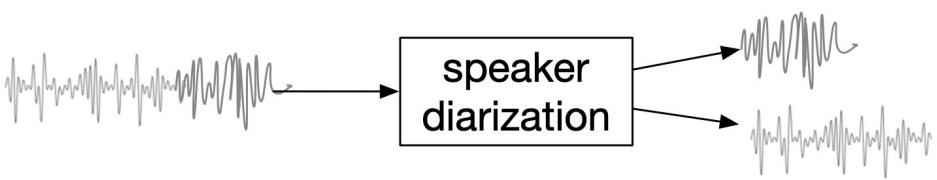
Keyword Spotting (KWS)

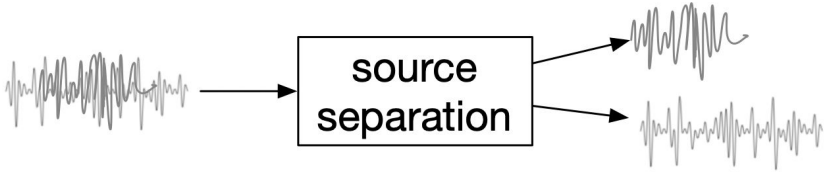
- Finding specific words in a large volume of (potentially noisy) speech.
- Useful for indexing, searching large recordings

Speaker identification/recognition

- Identifying “who speaks”, can be done with a known text (transcribed, “text-dependent”) or without.
- Useful for:
  - Human machine inferface
  - Improving ASR performance (adapt to the speaker)
  - Diarization / source separation
- Relies a lot on the signal
- Different from speaker verification / authentication
  - Can be used for biometrics (authentication)

Separation tasks





- Speaker diarization: “who speaks and when”, think of a meeting when multiple persons speak.
  - What good is perfect ASR if we don’t know who said what?
  - Some similarities with speaker ID.
- Source separation: “which source said what”, and more generally applicable, e.g. in music.
  - Extremely hard in the general case (no assumptions, single microphone)
  - Feasible if you have a microphone array and want to separate speech from other noises for instance.

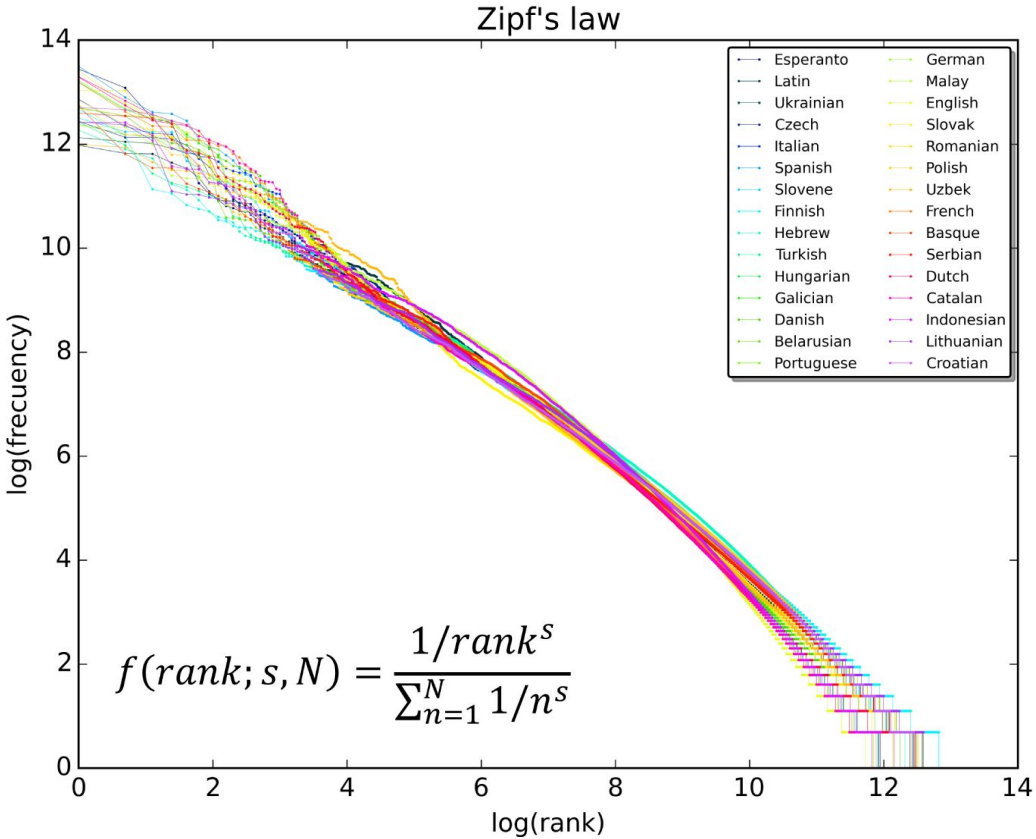
Speech enhancement / Denoising

- Removing background noise (street noise, dogs barking, keyboard typing, background TV/radio...)
- From clean speech + noise, produce cleaned speech.
- Objective and subjective measures:
  - PESQ: Perceptual Evaluation of Speech Quality
  - STOI: Short-Time Objective Intelligibility
  - (S)SNR: (Segmental) Signal Noise Ratio
  - MOS: Mean Opinion Score (human judgement)
- Applications in video conferencing, hearing aids, noise-robust ASR...

Why is it difficult?

- Input signal: 16kHz → output: a few words per second
- **Language** is difficult: diversity, variation, sparsity, ambiguity. Large or open vocabulary, accents, dialects.
- **Style**: spontaneous, conversational, read speech. Single word vs continuous speech
- Background **noise**, recording conditions, microphone.
- **Speaker** differences (can be adapted to): age, congestion...
- Speech recognition is rarely the end task (only for transcription), **downstream tasks** are imperfectly defined or unsolved.

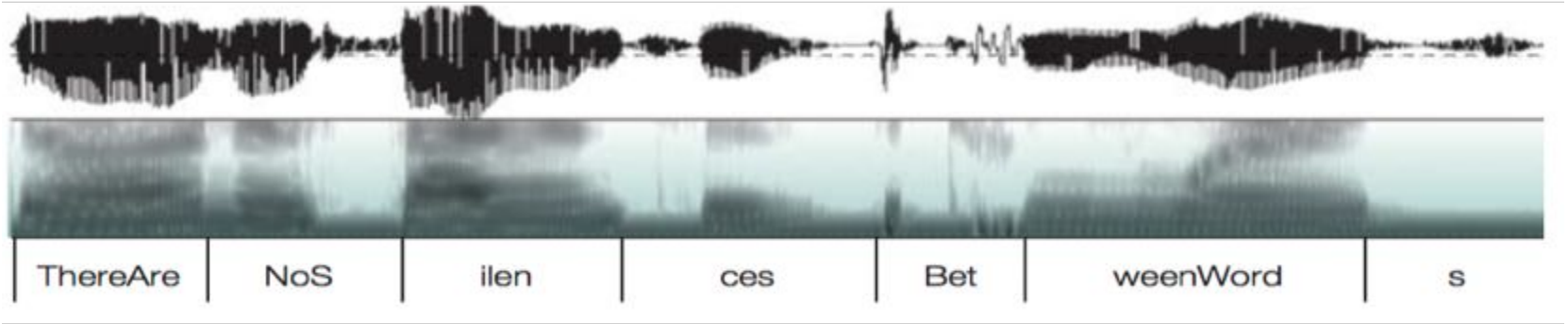
Zipf law



A plot of the rank versus frequency for the first 10 million words in 30 Wikipedias (dumps from October 2015) in a log-log scale.

https://en.wikipedia.org/wiki/Zipf%27s_law

therearenosilencesbetweenwords*



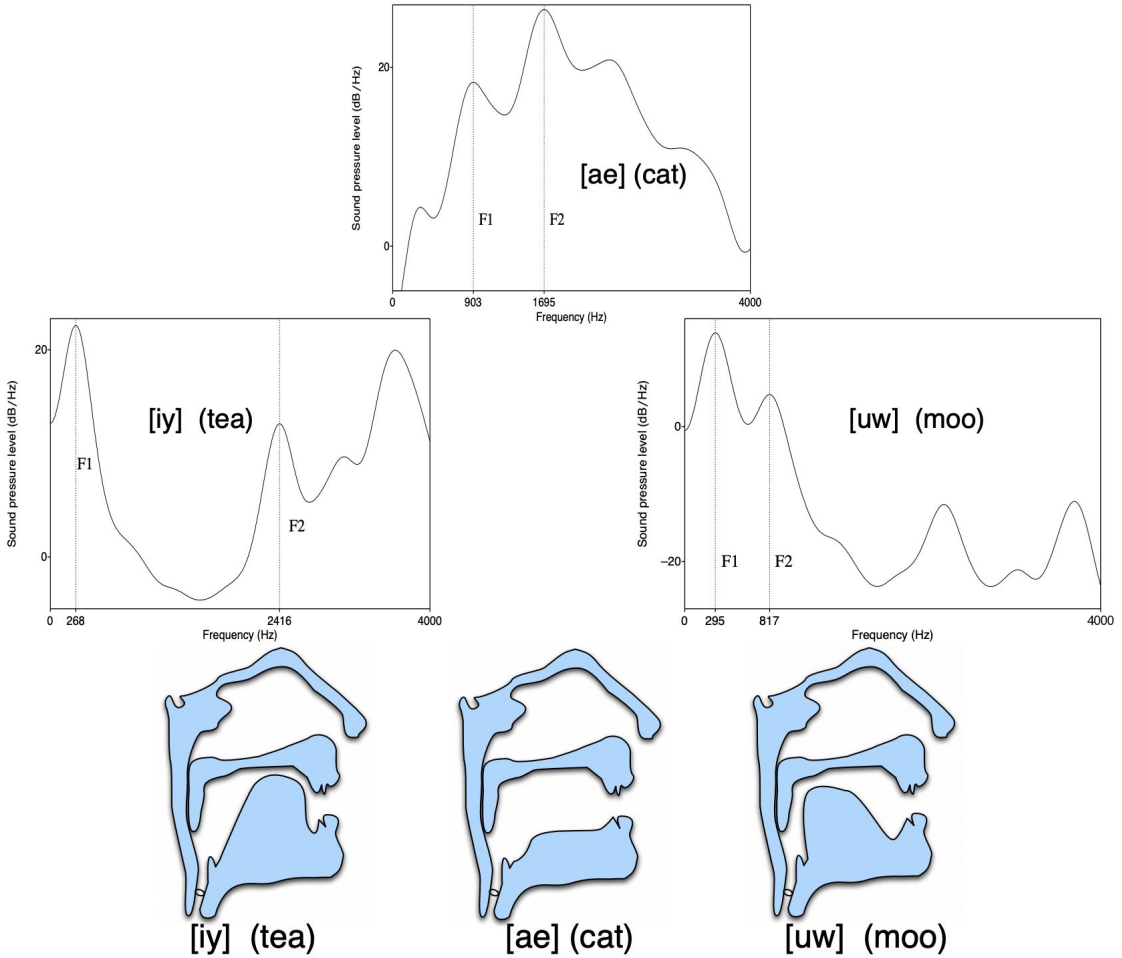
[Kuhl et al. 2004]

*theyareinyourheadbecauseyouknowthislexicon

Phones and Phonemes

- Phoneme = to speech what letters are to written language.
- Phones = acoustic realization of phonemes. 35-75ms duration for consonants and 60-250ms duration for vowels. Good approximation 70-100ms
- PER = Phone Error Rate, or evaluated in classification

Phones and Phonemes



[Jurafsky & Martin 2017]

High level
Context

Signal
Low level

Linguistics in a hurry

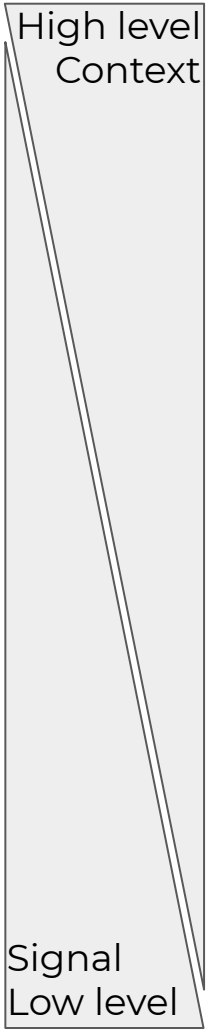
Phonological level

International Phonetic Alphabet

[aɪ pʰiː eɪ]

Graphemic level

enough, cough, draught, although, brought, through, thorough, hiccough



Morphological level

walked = walk+ed, unwashable = un+wash+able
recognition = re+cogn+ition

Phonological level

International Phonetic Alphabet

[aɪ pʰiː eɪ]

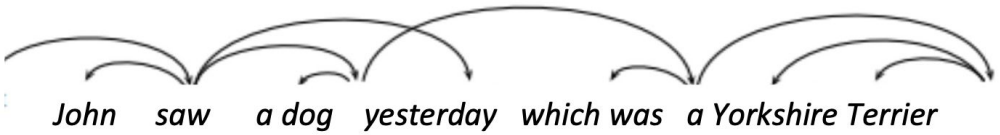
Graphemic level

enough, cough, draught, although, brought, through, thorough, hiccough

High level
Context

Signal
Low level

Syntactic level



Morphological level

walked = walk+ed, unwashable = un+wash+able
recognition = re+cogn+ition

Phonological level

International Phonetic Alphabet

[aɪ pʰiː eɪ]

Graphemic level

enough, cough, draught, although, brought, through, thorough, hiccough

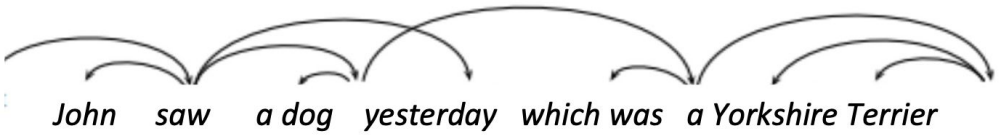
High level
Context

Signal
Low level

Semantic level

The landlord$_{SPEAKER}$ has not yet **REPLIED**$^{Communication\_response}$ in writing$_{MEDIUM}$ to the tenant$_{ADRESSEE}$ objecting the proposed alterations$_{MESSAGE}$.*DNI*$_{TRIGGER}$

Syntactic level



Morphological level

walked = walk+ed, unwashable = un+wash+able
recognition = re+cogn+ition

Phonological level

International Phonetic Alphabet

[aɪ pʰiː eɪ]

Graphemic level

enough, cough, draught, although, brought, through, thorough, hiccough

High level
Context

Signal
Low level

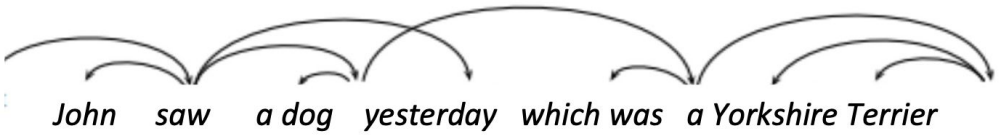
Linguistic context

— *You know what? John gave Peter a Christmas present yesterday*
— *Wow, was he surprised? What was it like?*
— *Surprisingly good. He spent quite a bit on it.*

Semantic level

The landlord$_{SPEAKER}$ has not yet **REPLIED**$^{Communication\_response}$ in writing$_{MEDIUM}$ to the tenant$_{ADRESSEE}$ objecting the proposed alterations$_{MESSAGE}$.*DNI*$_{TRIGGER}$

Syntactic level



Morphological level

walked = walk+ed, unwashable = un+wash+able
recognition = re+cogn+ition

Phonological level

International Phonetic Alphabet

[aɪ pʰiː eɪ]

Graphemic level

enough, cough, draught, although, brought, through, thorough, hiccough

High level
Context

Signal
Low level

Extra-linguistic context

*Found **him** in the street inside a bag. I think **he** is happy with his new life*

http://9gag.com/gag/azVnEwp/found-him-in-the-street-inside-a-bag-i-think-he-is-happy-with-his-new-life

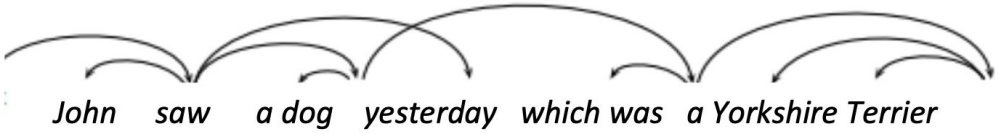
Linguistic context

— You know what? John gave Peter a Christmas present yesterday
*— Wow, was **he** surprised? What was **it** like?*
*— Surprisingly good. **He** spent quite a bit on it.*

Semantic level

The landlord$_{\text{SPEAKER}}$ has not yet **REPLIED**$^{\text{Communication\_response}}$ in writing$_{\text{MEDIUM}}$ to the tenant$_{\text{ADRESSEE}}$ objecting the proposed alterations$_{\text{MESSAGE}}$.*DNI*$_{\text{TRIGGER}}$

Syntactic level



Morphological level

walked = walk+ed, unwashable = un+wash+able
recognition = re+cogn+ition

Phonological level

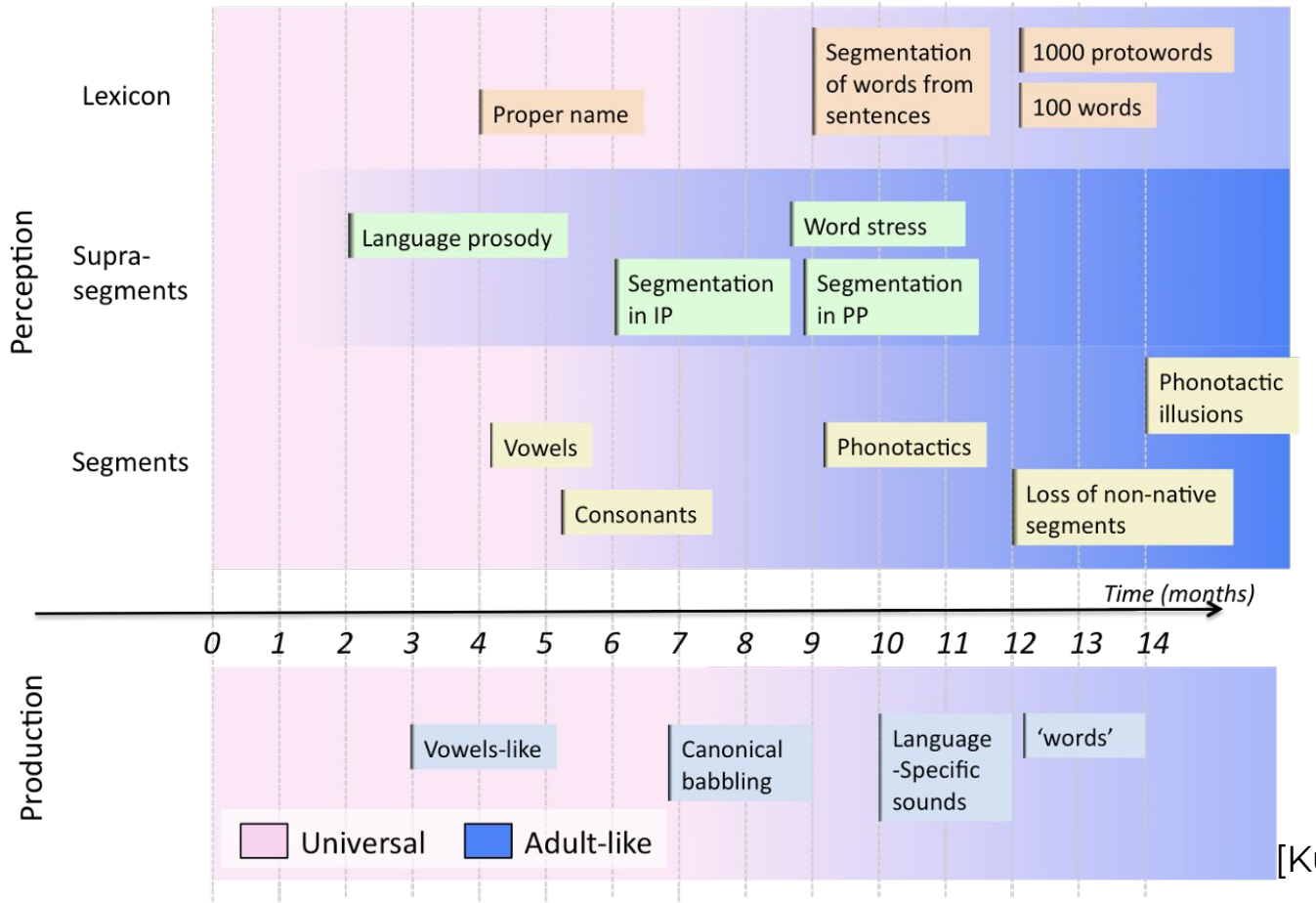
International Phonetic Alphabet

[aɪ pʰiː eɪ]

Graphemic level

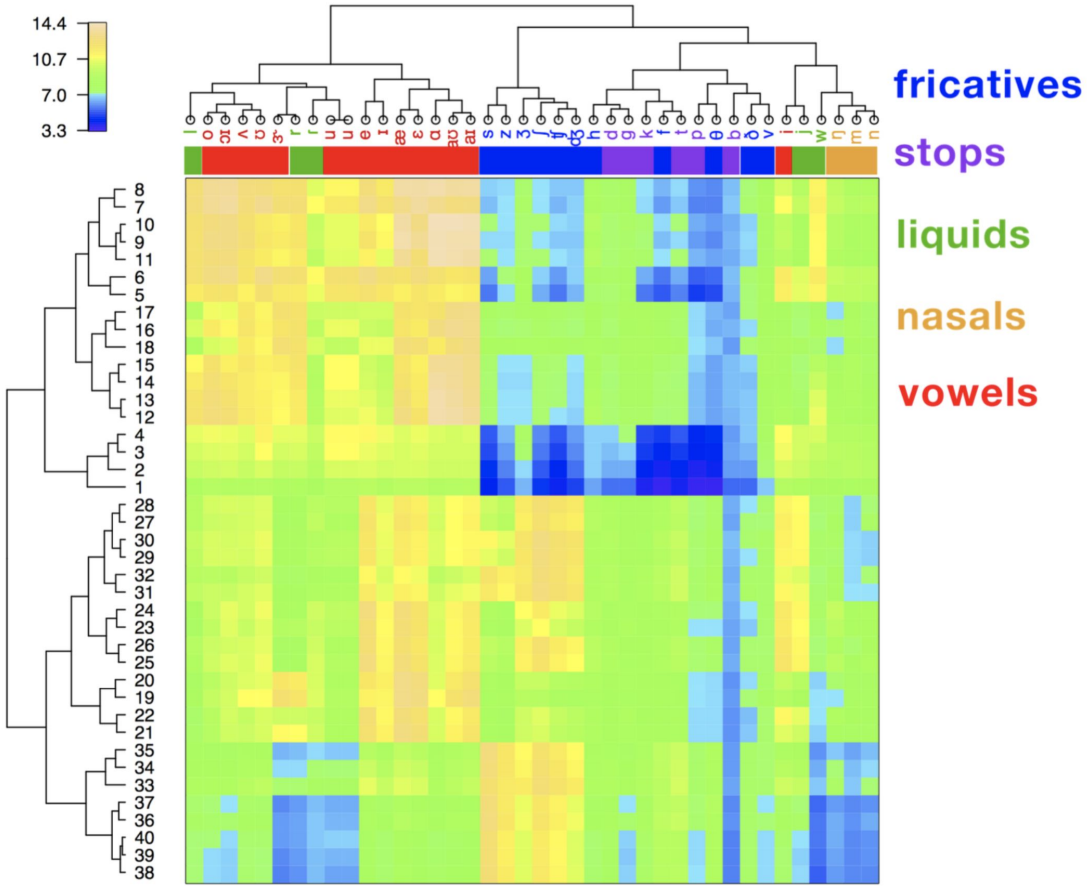
enough, cough, draught, although, brought, through, thorough, hiccough

Language acquisition in babies



[Kuhl et al. 2004]

Bi-clustering of phonemes with learned features



A brief history of ASR

- Caveat: this is a quick tour, with many ellipsis.
- ASR research builds on ancient statistical modeling and linguistic foundations that are omitted.

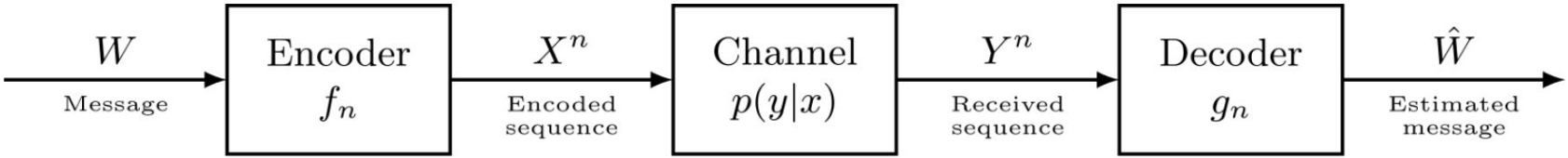
Beginnings: handcrafted systems

1952: Bell Labs, “Audrey”, recognized single digits from formants, tuned for a single s

1962: IBM, “Shoebox”, recognized 16 words

Modeling and statistical methods

1948: Noisy-channel coding theorem, Shannon



1966: Linear predictive coding (LPC) for speech coding, from NTT

1970-80s: development of Hidden Markov Models (HMMs)

1976: Jelinek “Continuous Speech Recognition by Statistical Methods”

1983: Bahl et al. “Maximum likelihood approach to continuous speech recognition”

HMMs (mostly HMM-GMM)

1976: Jelinek “Continuous Speech Recognition by Statistical Methods” (he later said “Every time I fire a linguist, the performance of the speech recognizer goes up”)

1980s: HMM + Gaussian and early GMM emissions

1990s: complex, tied states, HMM-GMMs

Late 90s, early 2000s: discriminative training of HMMs

Deep Learning

1989: Bottou et al., TDNN (1D ConvNet) for ASR

1989: Bourlard et al., RNN for ASR

1995: LeCun & Bengio, ConvNets for ASR

2009: Mohamed et al, HMM-DBN

2012: several improvements on HMM-DNN

2014: scalable and scaled end-to-end training

2016: attention and seq2seq losses

Common datasets (today)

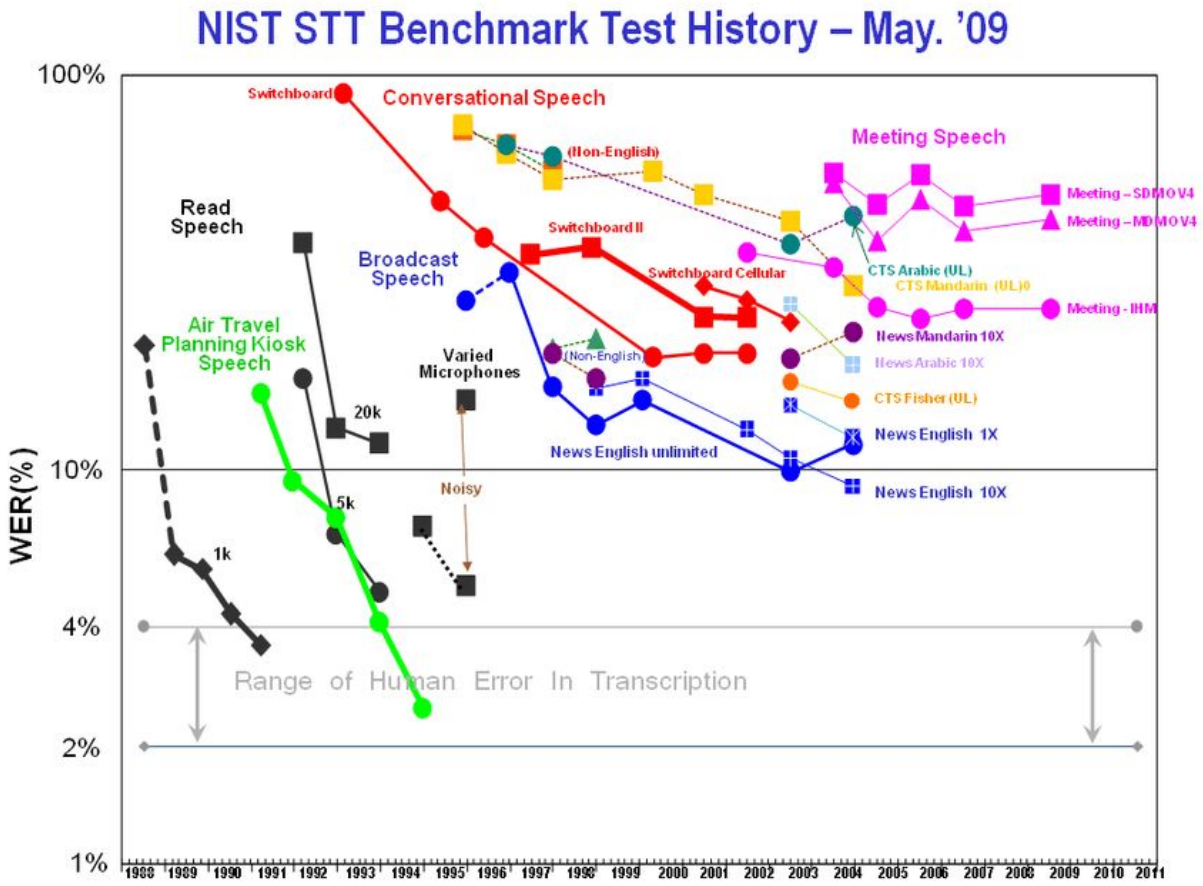
- TIMIT (4h)
- Wall Street Journal (80h)
- LibriSpeech (1000h)
- SwitchBoard + Fisher (300h + 2000h)
- TED-LIUM (450h)
- CommonVoice (1000+h in 40 languages)
- Private datasets (companies)

English Only

=> English centric

- “Resource rich” languages: English, Chinese, French, German, Spanish, Japanese...

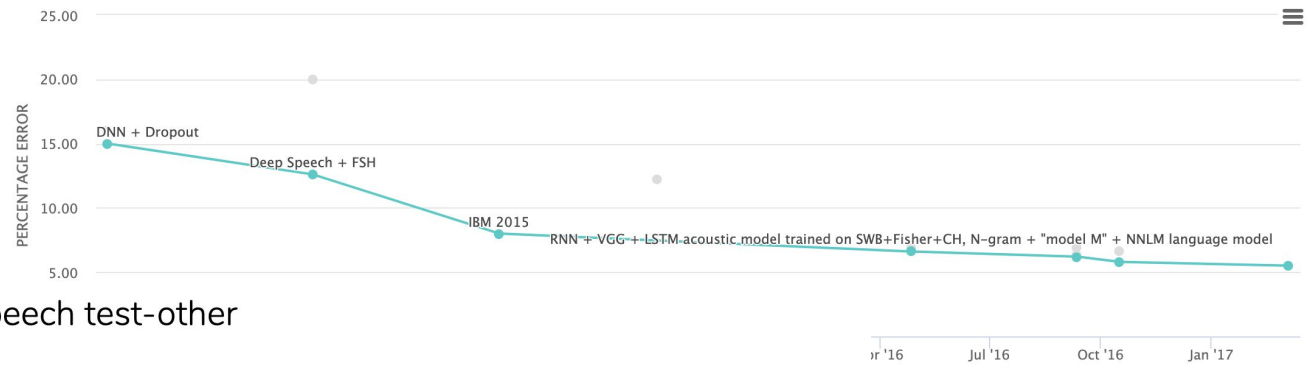
Performance over time (pre-DL)



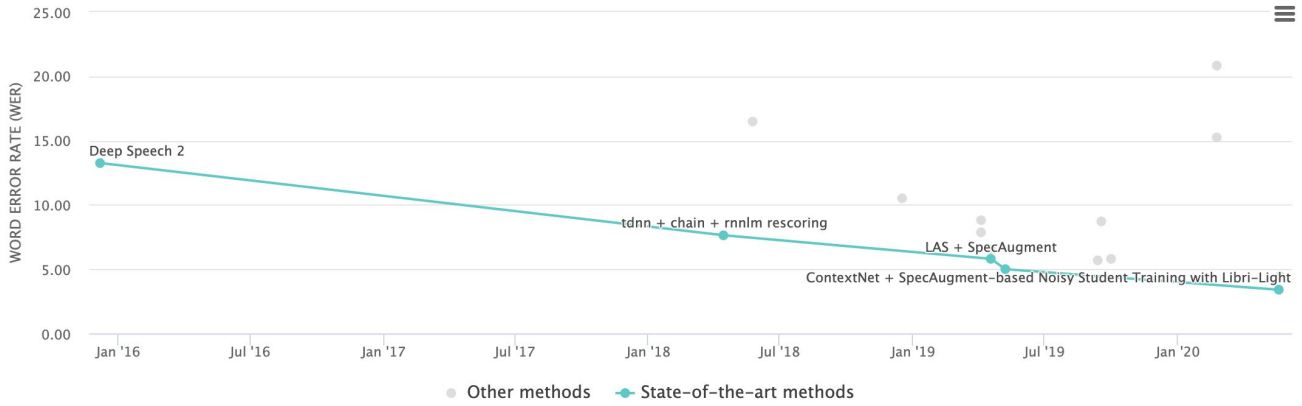
[Laurens van der Werf, 2012]

Performance over time (DL era)

Speech Recognition on Switchboard + Hub500



Speech Recognition on LibriSpeech test-other



[Papers with code]

“Deep Learning”: 3 major changes

- Improvements in acoustic models (AMs)
  - Phone recognition (PER): in 2014, ≈30% relative improvement over best HMM-GMM, nowadays about 45% relative improvement
  - ASR: on English read speech, sufficiently large corpus for DL to shine: in 2015 ≈38% relative improvement. Nowadays (combined with LM improvements) >80%.
- Improvements in language models (LMs)
  - Mikolov et al. 2010: WER on WSJ, 18% relative improvement
  - PPL of Transformer LM vs. 4gram >60% relative improvement
- End-to-end training
  - End-to-end trained systems caught up with HMM-DNN in 2019
  - Making training across datasets, languages easier

https://github.com/syhw/wer_are_we/

What does it mean to be “superhuman”???

- Why do speech-recognition based service still make errors?
  - Lack of leveraging the context for transcription to disambiguate
  - Models that are downstream in the pipeline make errors
- Narrow benchmarks
- Models don’t generalize well outside domains they are trained on
- Results depend a lot of the language model too
- There are noisy speech datasets and competition, few are naturally noisy, results there are not “superhuman” yet.

The ASR metric: Word Error Rate

$$WER = \frac{S + D + I}{N} = \frac{S + D + I}{S + D + C}$$

- It's just the edit distance! (substitutions + deletions + insertions)
- Can be computed by dynamic programming:

$$D_{0,0} = 0$$
$$D_{i,0} = i, 1 \leq i \leq m \quad \text{otherwise: } D_{i,j} = \min \begin{cases} D_{i-1,j-1} & +0 \text{ if } u_i = v_j \\ D_{i-1,j-1} & +1 \text{ (Replacement)} \\ D_{i,j-1} & +1 \text{ (Insertion)} \\ D_{i-1,j} & +1 \text{ (Deletion)} \end{cases}$$
$$D_{0,j} = j, 1 \leq j \leq n$$

ASR metrics that measure speed

- **Throughput**: amount (duration) of audio processed per second, can be batched, higher is better.
- **Latency** (also, lookahead): how much time after a given frame will it be transcribed, lower is better.
- Real-time factor (**RTF**): ratio of ASR processing time to utterance duration, lower is better.

Variable metrics, depend on a lot of parts in the system, usually interested in average and 90% quantile values.

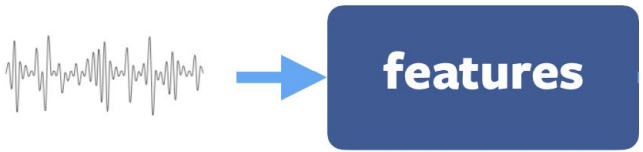
To be able to do online (“live”) speech recognition, a system must have an RTF ≤ 1 and a minimal/small latency.

The ASR pipeline

- Two main components:
  - Acoustic model: sounds to some discrete representation (e.g. words).
    P(sound | phoneme) or P(sound | word)
  - Language model(s): discrete (e.g. words) to “transcription” (full sentence).
    $P(word_{k+1} \mid word_k)$
- Historically long pipelines:
  Speech -> features -> GMM -> HMM -> LM (over senones states)
  Speech -> phonemes -> words -> LM -> sentence
- Now trending towards simpler/shorter paths:
  Speech -> neural network -> sentence

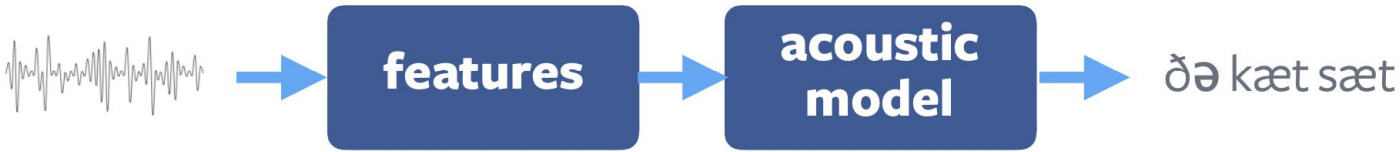
The ASR pipeline: acoustic modeling

The ASR pipeline: acoustic modeling



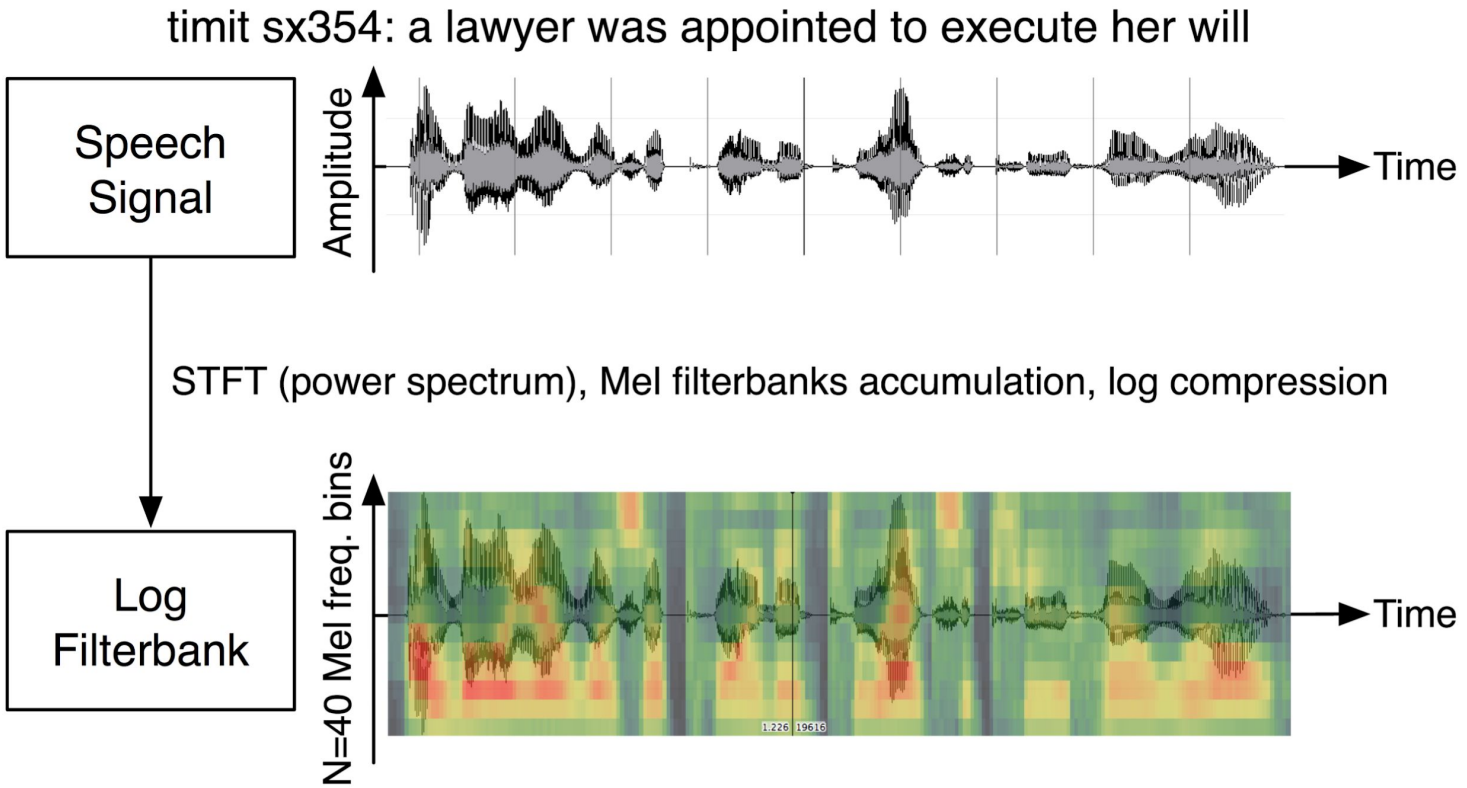
The ASR pipeline: acoustic modeling

The ASR pipeline: acoustic modeling



The ASR pipeline: acoustic modeling

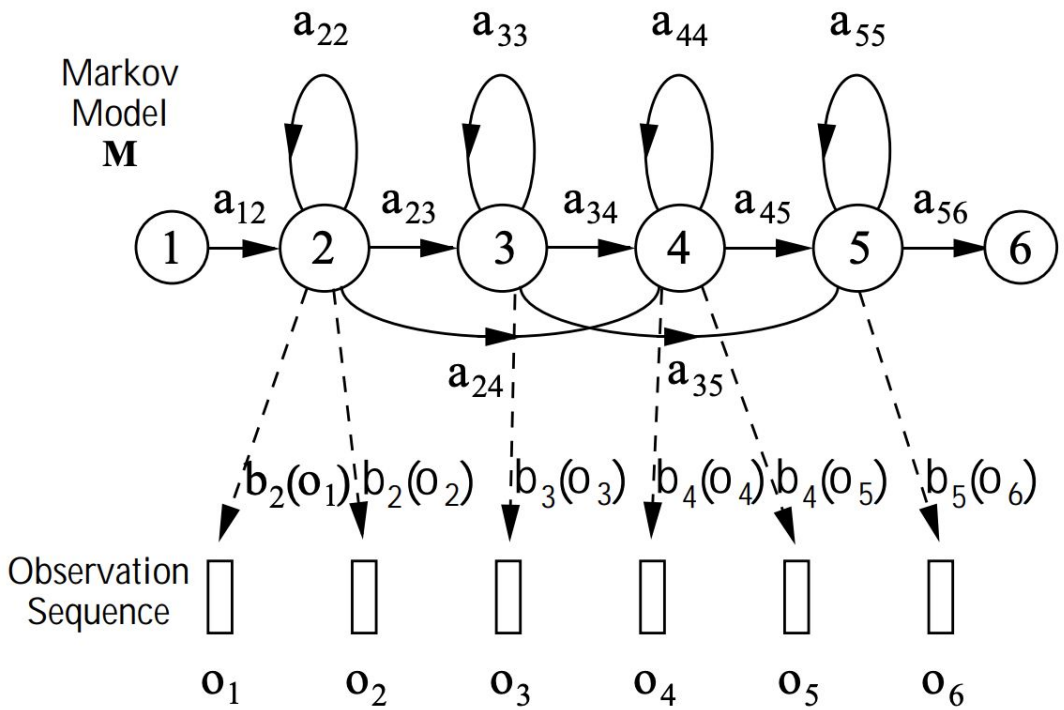
Signal processing / featurizing

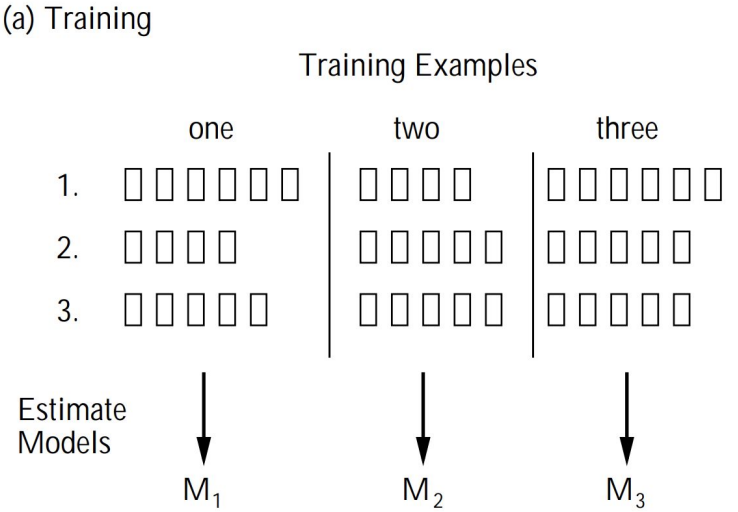


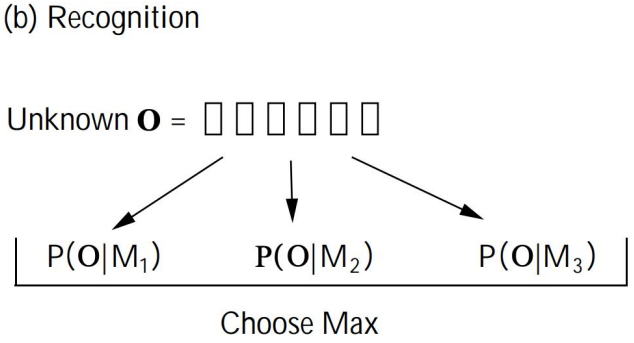
Acoustic model

- For going from speech features to phones
- Hidden Markov Model with Gaussian Mixture Model of emissions (HMM-GMM): $P(\text{speech} \mid \text{phone}_k)P(\text{phone}_k \mid \text{phone}_{k-1})$
- $P(\text{phone}_k \mid \text{phone}_{k-1})$ = HMM, c.f. [Rabiner 1989]
- $P(\text{speech} \mid \text{phone}_k)$ = GMM
- $P(\text{word}_p \mid \{\text{phone}_k \ldots \text{phone}_{k+K}\})$ = decoding process

HMM-GMM [HTK book]

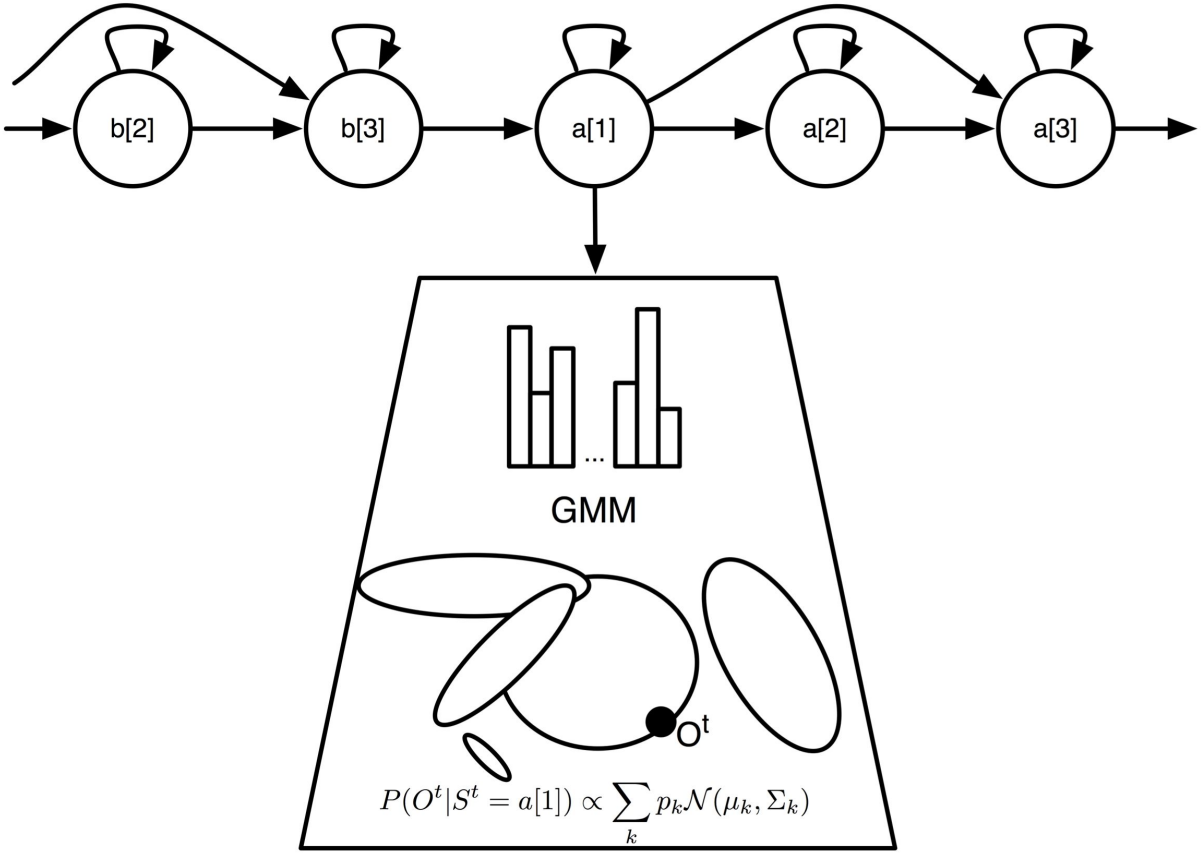




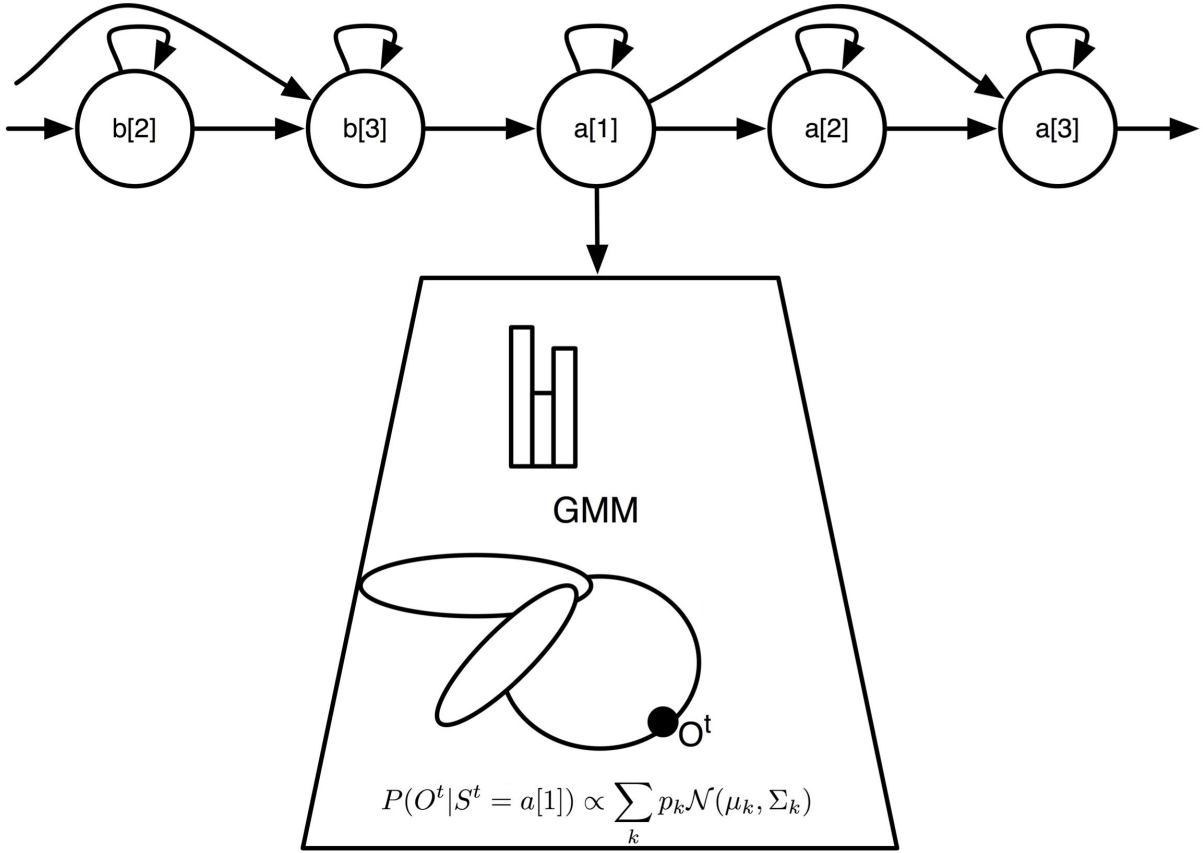


(Senones, tied states, silence model, omitted.)

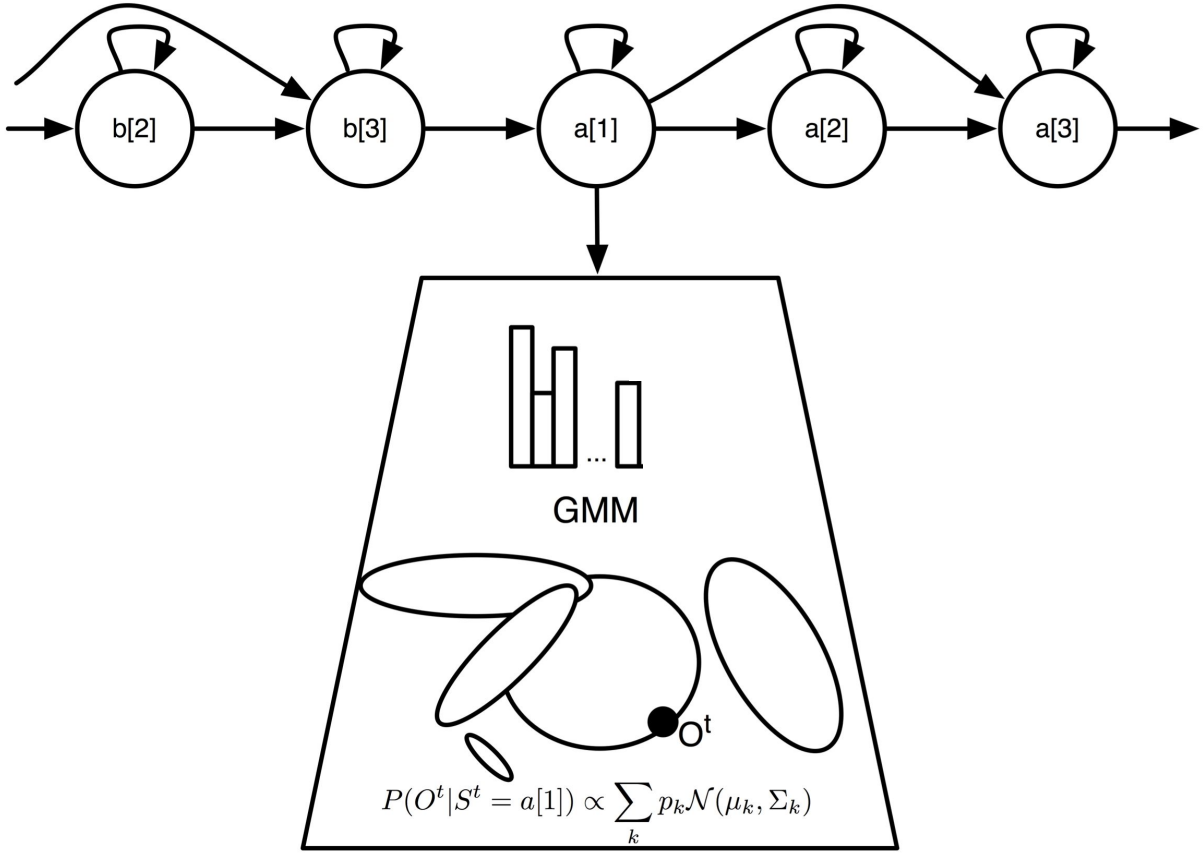
HMM-GMM


b[2]
b[3]
a[1]
a[2]
a[3]
GMM
O^t
$P(O^t|S^t = a[1]) \propto \sum_k p_k \mathcal{N}(\mu_k, \Sigma_k)$

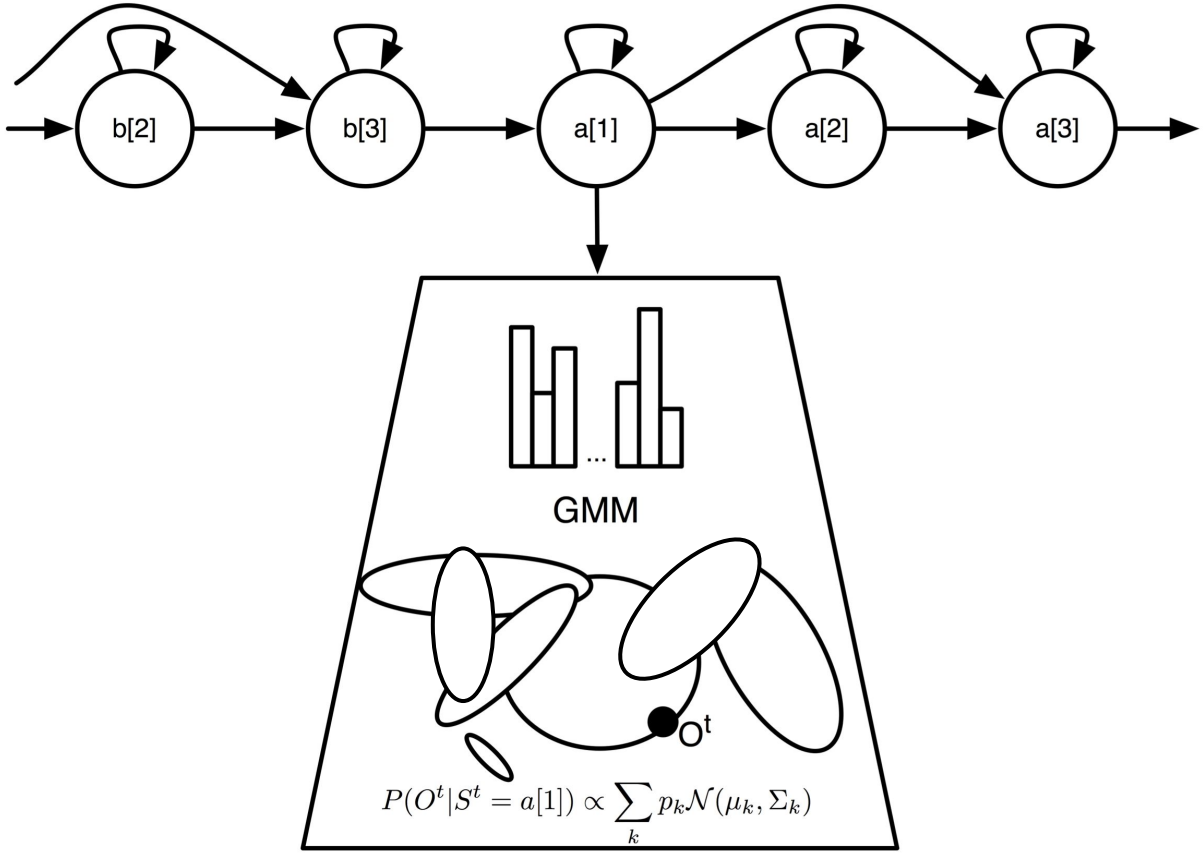
HMM-GMM (training step 1/3)



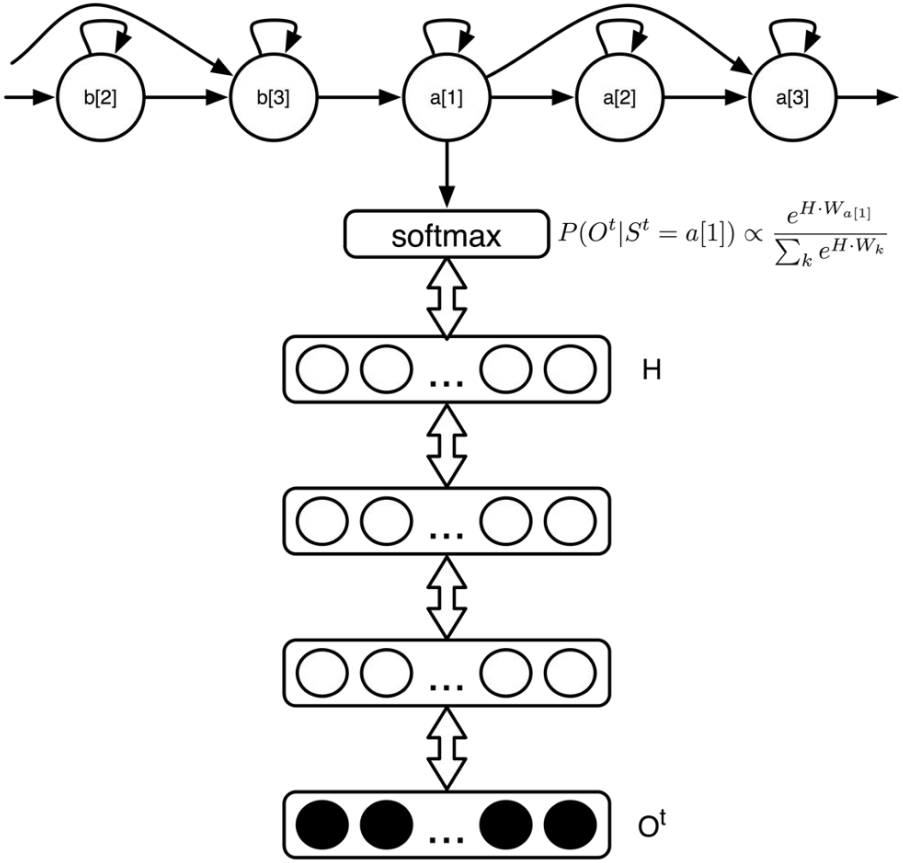
HMM-GMM (training step 2/3)



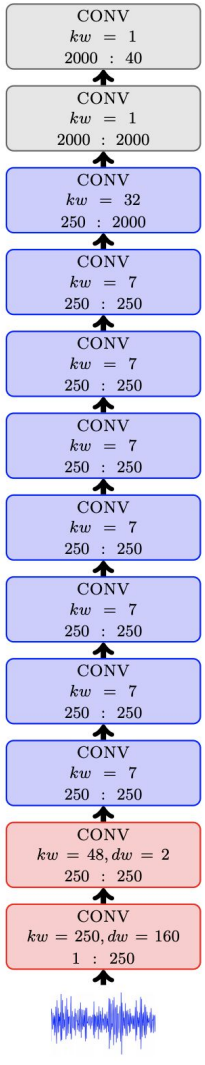
HMM-GMM (training step 3/3)



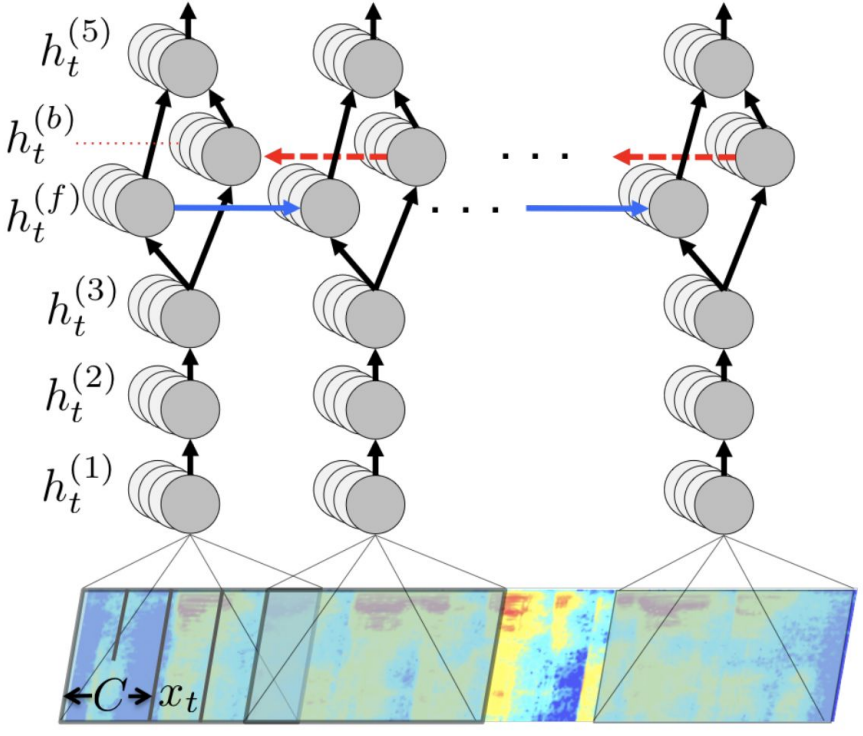
HMM-DNN


b[2]
b[3]
a[1]
a[2]
a[3]
softmax
$P(O^t|S^t = a[1]) \propto \frac{e^{H \cdot W_{a[1]}}}{\sum_k e^{H \cdot W_k}}$
H
O^t

Examples of modern acoustic models: ConvNets & RNNs

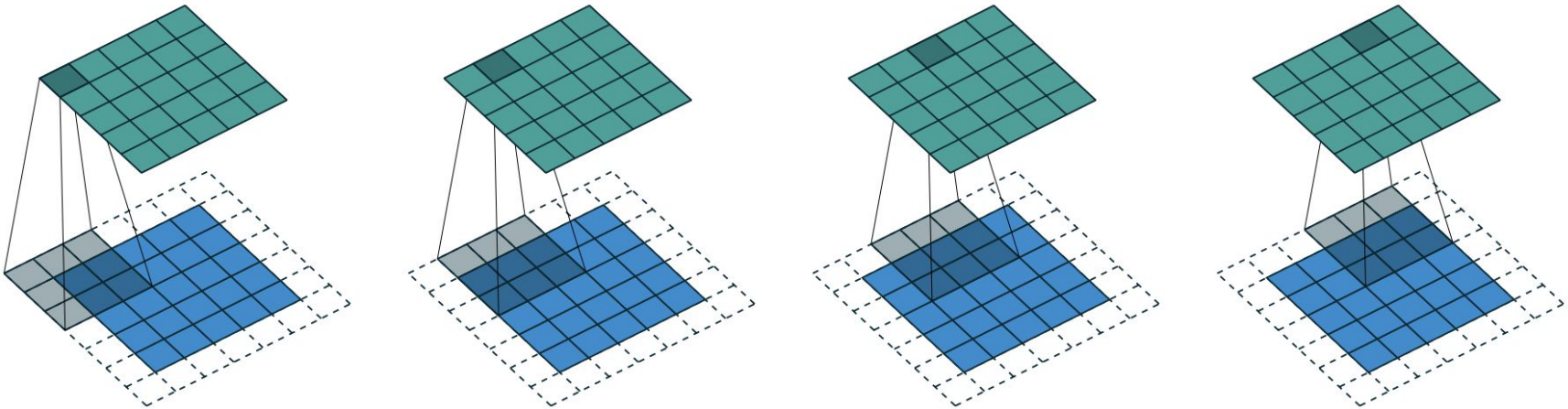


[wav2letter 2016]



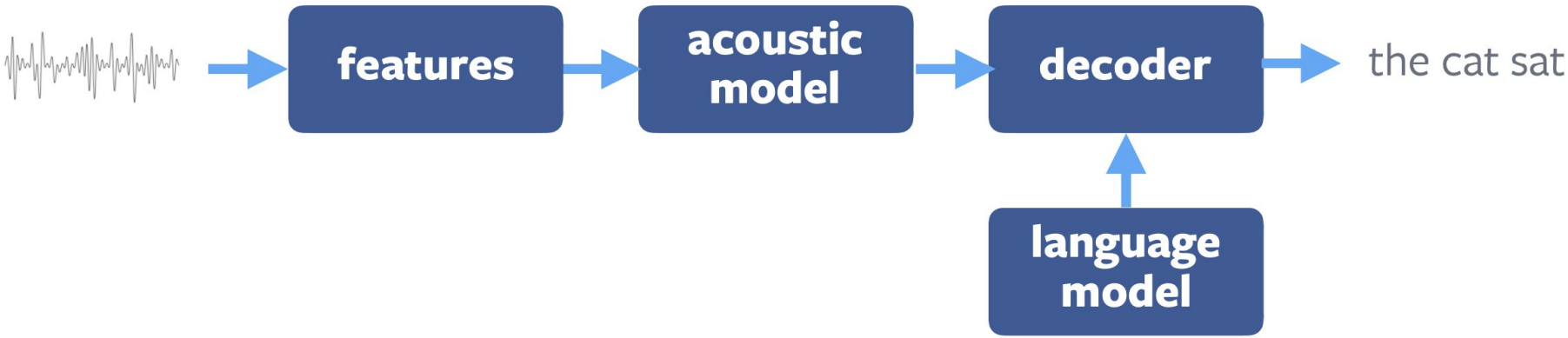
[Deep Speech 2014]

Convolution example

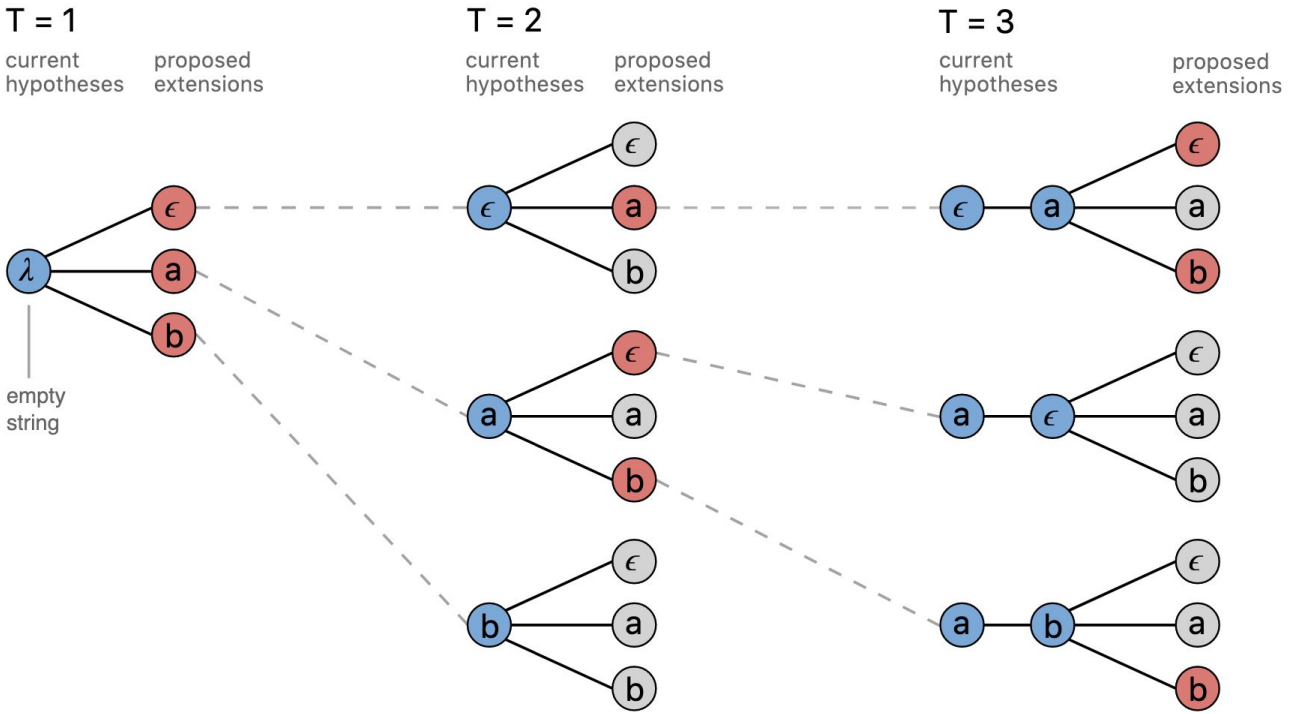


Convolving a 3 × 3 kernel over a 5 × 5 input using half padding and unit strides (i.e., i = 5, k = 3, s = 1 and p = 1). Figure from [Dumoulin & Visin 2018]

The ASR pipeline: decoding

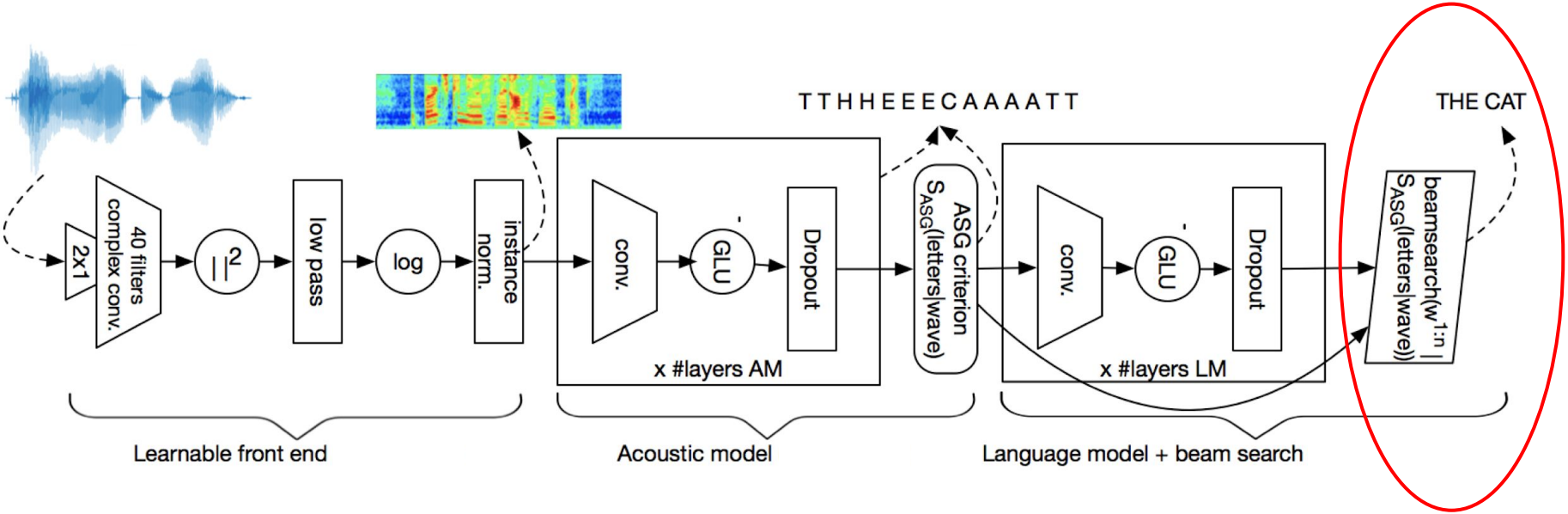


Beam search

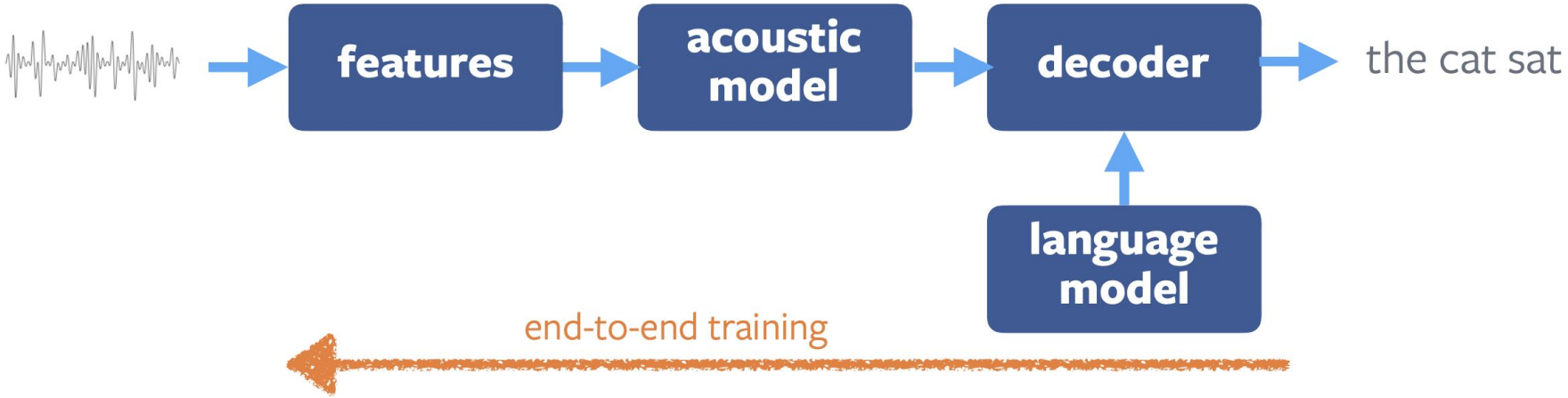


A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

“wav2letter”-like model

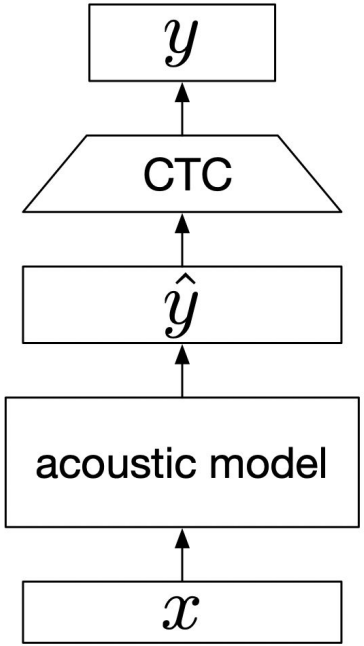


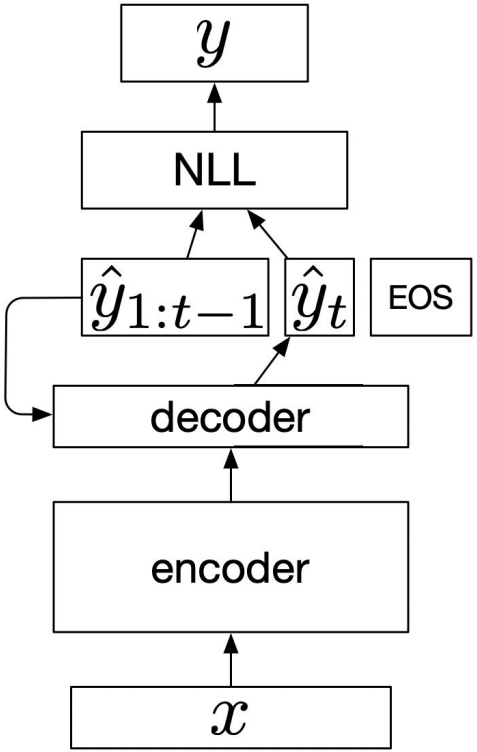
The ASR pipeline: end-to-end training



Deep learning losses: CTC, seq2seq...

x, y, yhat are sequences, but not of the same length





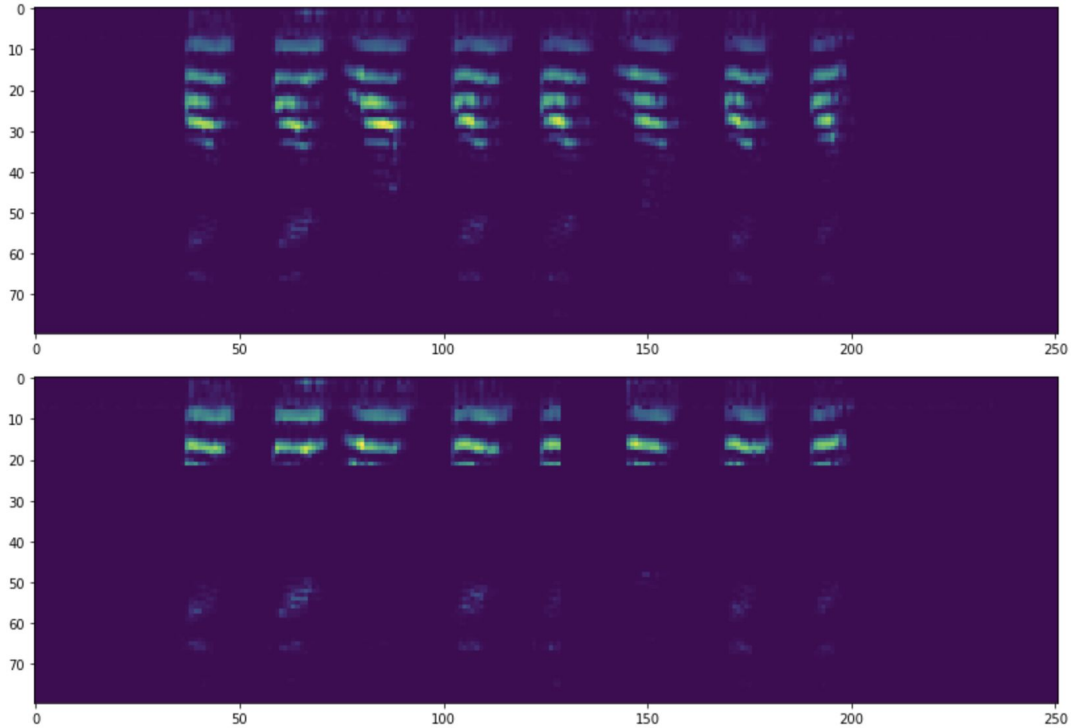
What is really end-to-end?

- From the **raw waveform**?
- Direct to letters/graphemes?
- Direct to words?
- LM outside the AM or not?
- No beam search?
- Trained without any **alignment**?

Data augmentation

To combat overfitting, it is often useful to:

- Vary the tempo, vary the pitch
- Add noises, add echos (reverberation)
- Drop some of the inputs, a recently very succesful structured drop out is SpecAugment (right).



Leveraging unlabeled speech audio

-

What we covered

- Speech tasks: ASR, KWS, TTS, speaker ID, speech enhancement, source separation...
- Why is it difficult? Spontaneous vs. read speech, large/open vocab., single word vs continuous, background noise, accents, dialects etc.
- Acoustics and Linguistics absolute basics: sound, phonemes, words.
- Brief history of speech rec.: HMM, DNN, E2E.
- ASR pipeline: sound->features->acoustic model->loss->LM (decoding)
- ASR metrics: WER/CER, inference latency, throughput, RTF

Course 2: Speech features and Acoustic Models

Neil Zeghidour

Slides at:
https://tinyurl.com/MVAspeechsignal

Lab 3: wav2letter Tutorial

Gabriel Synnaeve

What is this tutorial?

- Optional
- No code to write
- You can run decoding from wav2letter with an ngram language model:
  - Decoding a SOTA model
  - Or decoding in real time
- **If you have an NVIDIA GPU** (or a few), you can train an acoustic model with wav2letter.

Preparation: Download and Install Docker w2l

- Docker
  - For Mac https://hub.docker.com/editions/community/docker-ce-desktop-mac/
  - For Windows https://hub.docker.com/editions/community/docker-ce-desktop-windows/
  - For Linux https://docs.docker.com/engine/install/ (select your Linux distribution)
- **If you have an NVIDIA GPU**, install https://github.com/NVIDIA/nvidia-docker/
- Run the adequate command from https://github.com/facebookresearch/wav2letter/wiki/Building-Running-with-Docker#using-pre-built-docker-images
- `sudo docker exec -it w2l bash`
- `cd /root/wav2letter/build`

Preparation: Download model + data

- wget https://dl.fbaipublicfiles.com/wav2letter/sota/2019/librispeech/models/am/librispeech-train%2Bdev-unigram-10000-nbest10.lexicon
- wget https://dl.fbaipublicfiles.com/wav2letter/sota/2019/librispeech/models/am/librispeech-train-all-unigram-10000.tokens
- wget https://dl.fbaipublicfiles.com/wav2letter/sota/2019/decoder-unigram-10000-nbest10.lexicon
- wget https://dl.fbaipublicfiles.com/wav2letter/sota/2019/librivox/models/am/am_transformer_ctc_librivox_dev_other.bin
- wget https://dl.fbaipublicfiles.com/wav2letter/lexicon_free/librispeech/models/lm/lm_librispeech_kenlm_word_4g_200kvocab.bin
- wget http://www.openslr.org/resources/12/dev-other.tar.gz && tar xvzf dev-other.tar.gz

Get some transcriptions

- See this page https://github.com/facebookresearch/wav2letter/wiki/Beam-Search-Decoder

- This is just computing Viterbi: `wav2letter/build/Test \`
  `--am path/to/train/am.bin \`
  `--maxload 10 \`
  `--test path/to/test/list/file`

- This is decoding: `wav2letter/build/Decode --am path/to/train/am.bin --test path/to/test/list/file --maxload 10 --nthread_decoder 2 --show --showletters --lexicon path/to/the/lexicon/file --uselexicon true --lm path/to/lm/file --lmtype kenlm --decodertype [wrd, tkn] --beamsize 100 --beamsizetoken 100 --beamthreashould 20 --lmweight 1 --wordscore 0 --eosscore 0 --silscore 0 --unkscore 0 --smearing max`

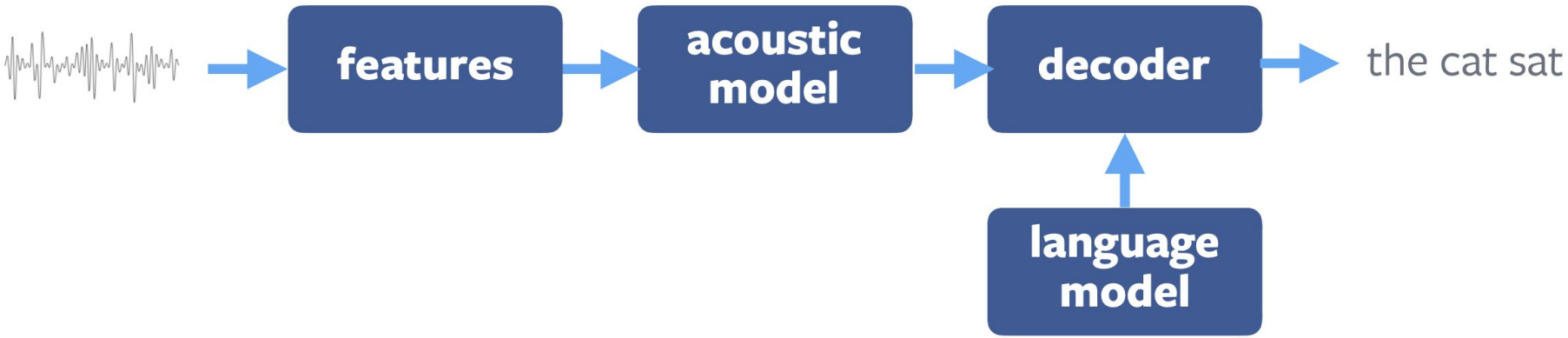
Further tutorials

- Train on LibriSpeech clean (**only if you have at least one GPU**)
  https://github.com/facebookresearch/wav2letter/tree/master/tutorials/1-librispeech_clean
- Decode in real-time
  https://github.com/facebookresearch/wav2letter/wiki/Inference-Run-Examples

Course 3: End-to-End Speech Recognition

Gabriel Synnaeve

Recap: the ASR pipeline



Why end-to-end training?

- It's simpler!
  - Simpler to optimize
  - Simpler to apply to different languages
  - Simpler to scale up
- (Controversial opinion) It's more hackable for research!
  - Better transfer
  - Easier to do more tasks
  - Easier to lift some of the supervision
  - Modeling
  - Loss functions

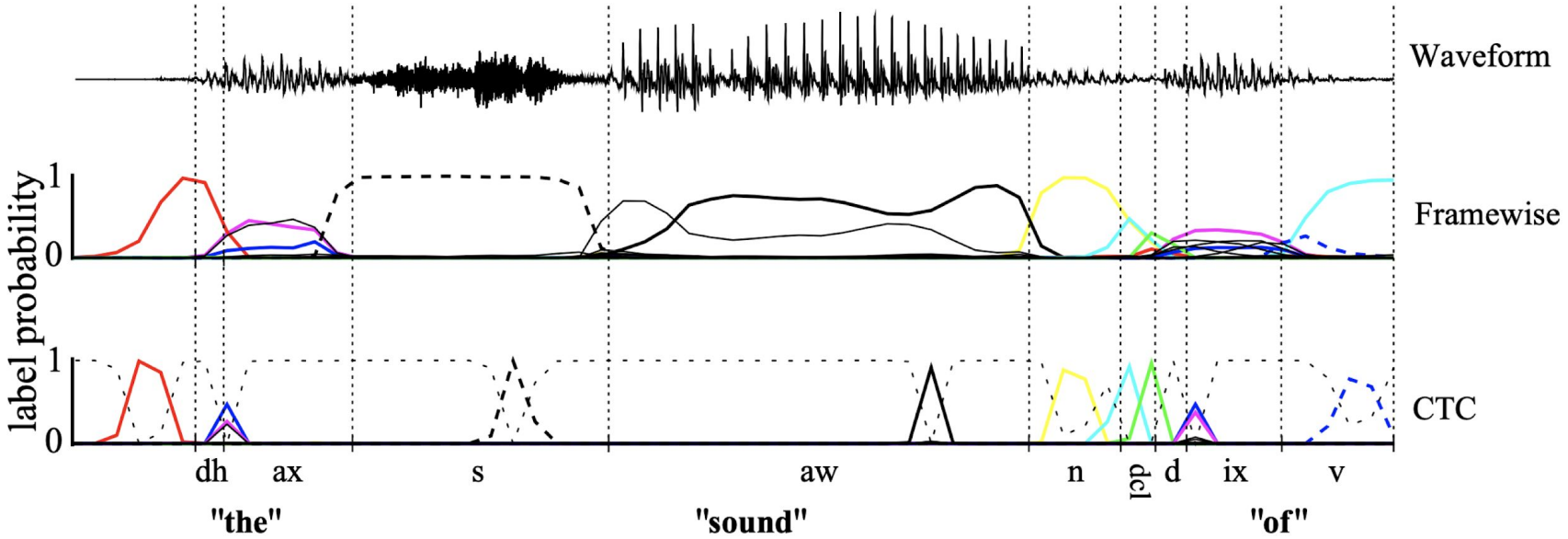
Plan for today:

- Sequence modeling and losses:
  - CTC, ASG
  - Seq2seq, RNN-T
- Leveraging an LM:
  - Beam search
  - Fusion
- Research results

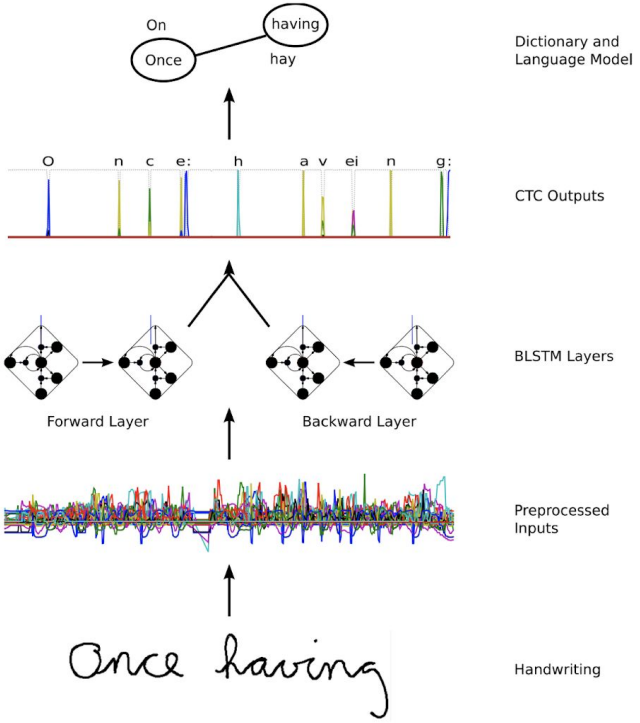
CTC: Connectionist Temporal Classification

Alex Graves, Santiago Fernandez, Faustino Gomez, Jürgen Schmidhuber (2006)

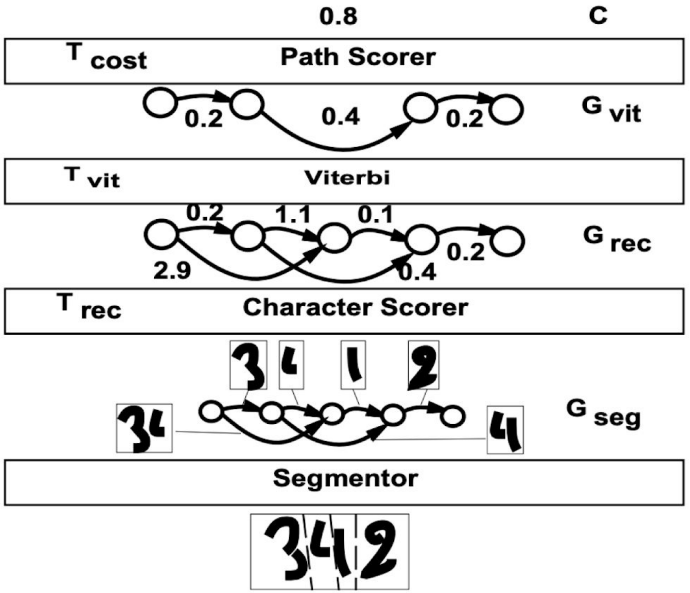
- Originally for RNNs for phone recognition and handwritten recognition
- No alignment necessary!



Handwritten recognition



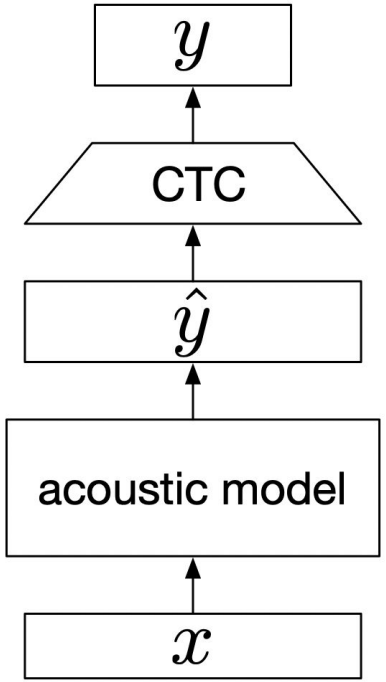
A Novel Connectionist System for Unconstrained Handwriting Recognition *Alex Graves, Marcus Liwicki, Santiago Fernandez Roman Bertolami, Horst Bunke, Jurgen Schmidhuber* (2008)



Global Training of Document Processing Systems using Graph Transformer Networks
Léon Bottou, Yoshua Bengio, Yann LeCun (1997)

CTC in the ASR Pipeline

x, y, yhat are sequences, but not of the same length



CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target,
  token dictionary is {a, c, t}
- Just sum the framewise
  scores of labels that correspond
  to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent
  given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$

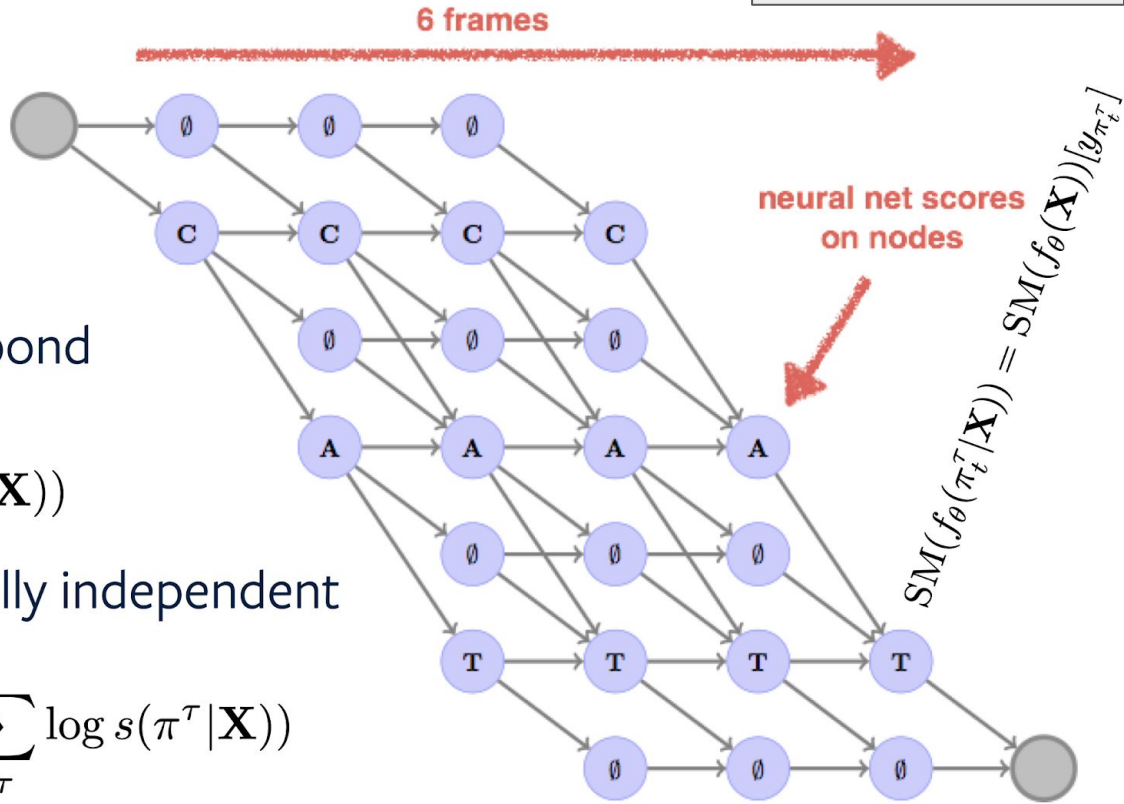
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



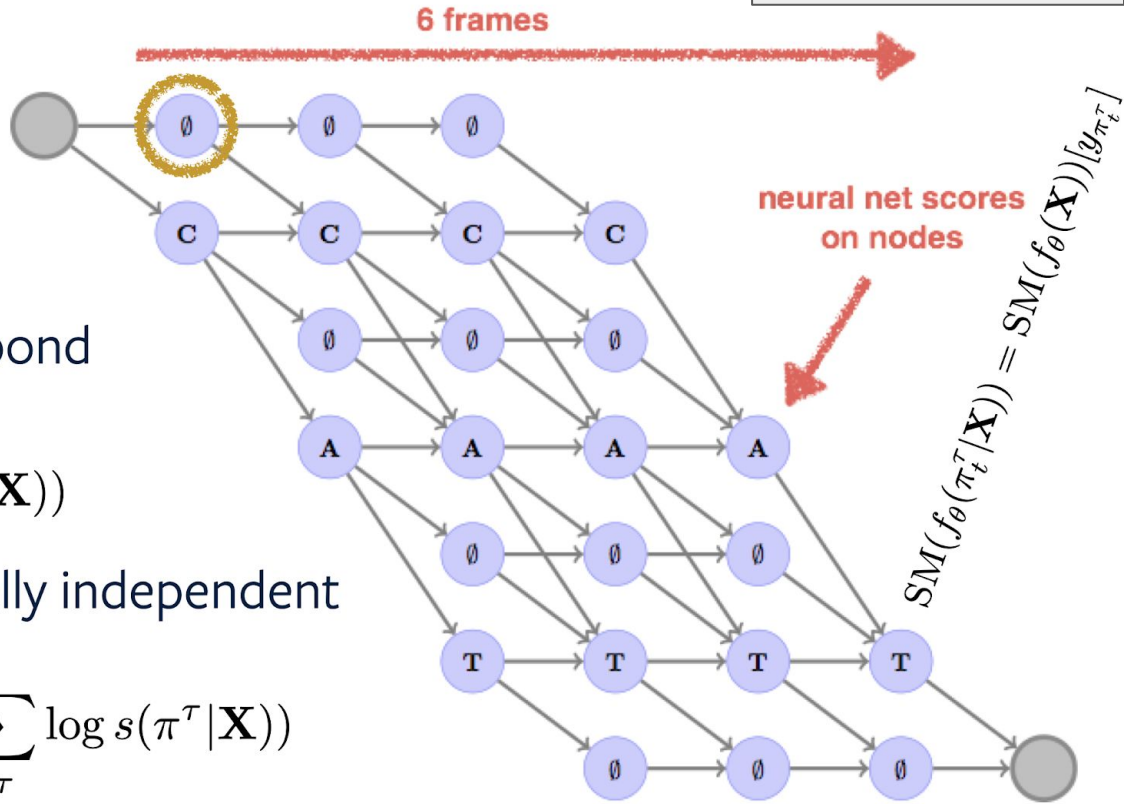
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



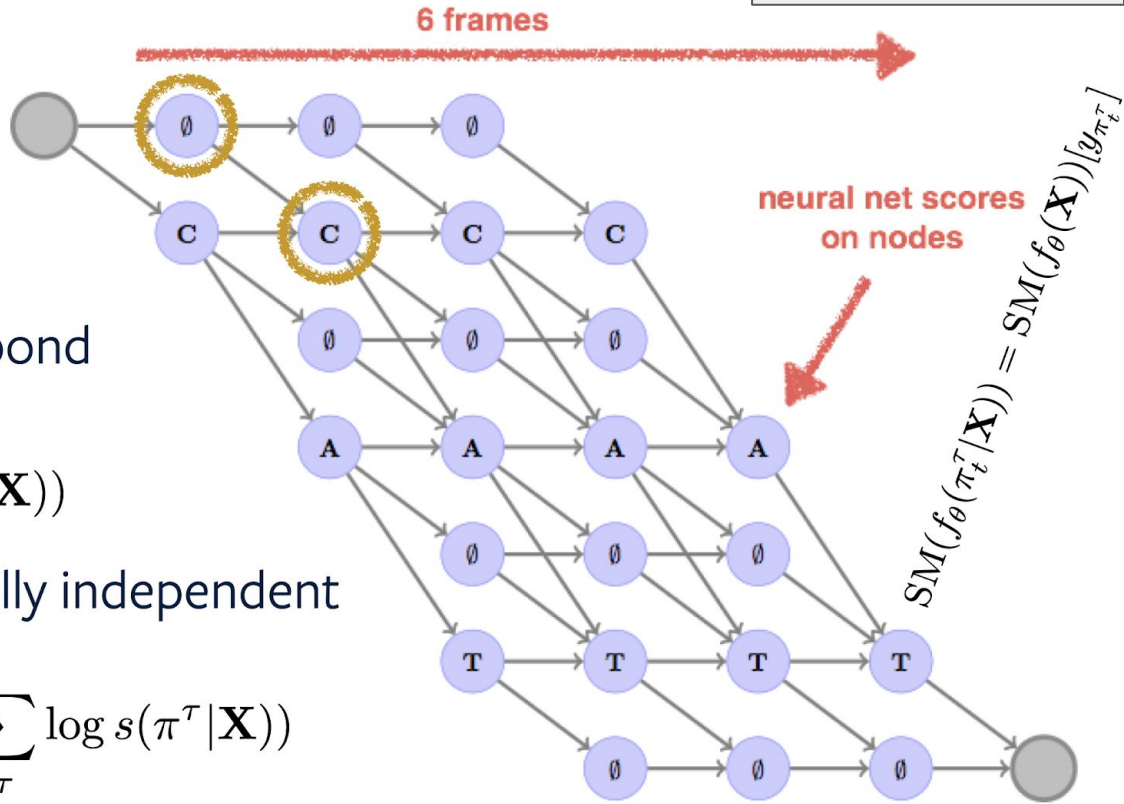
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



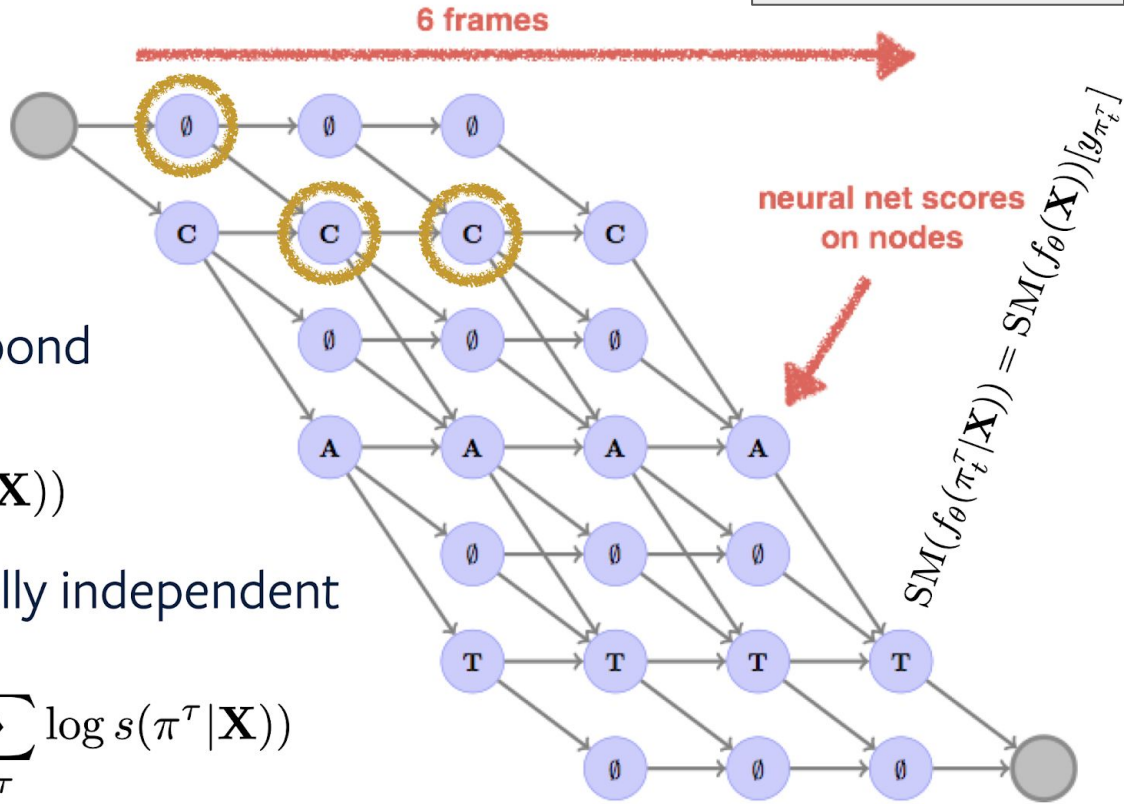
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



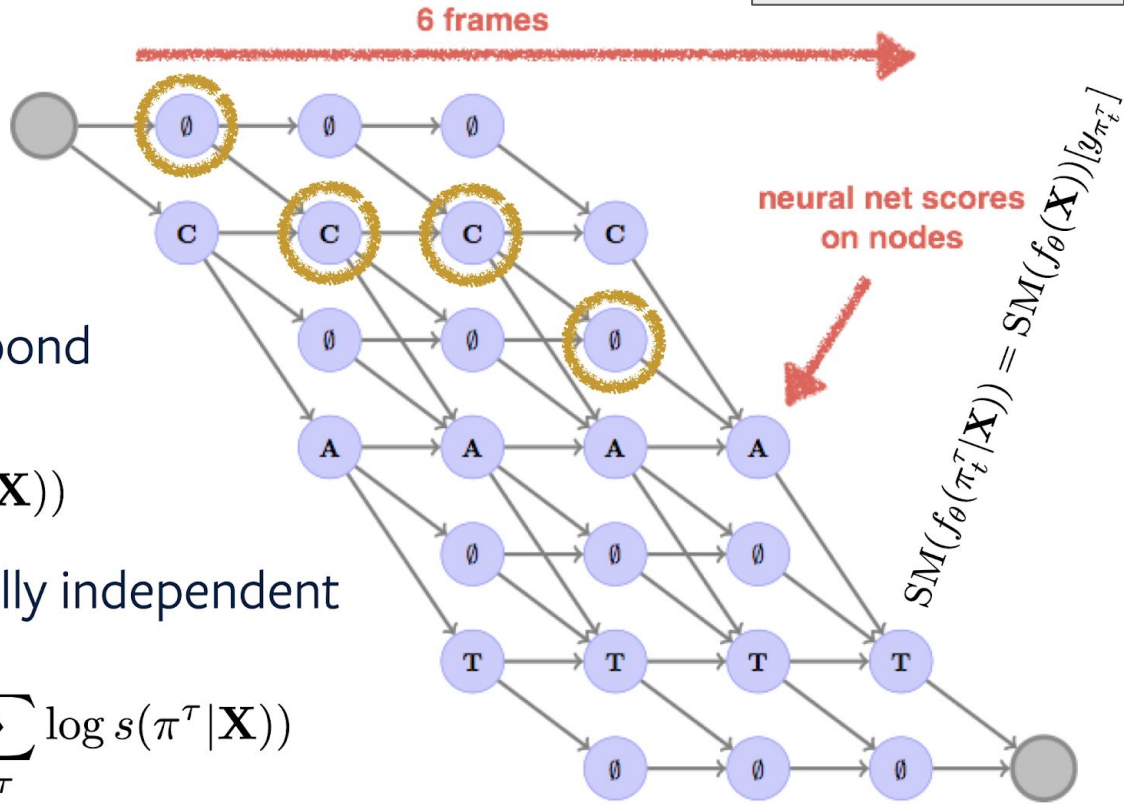
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



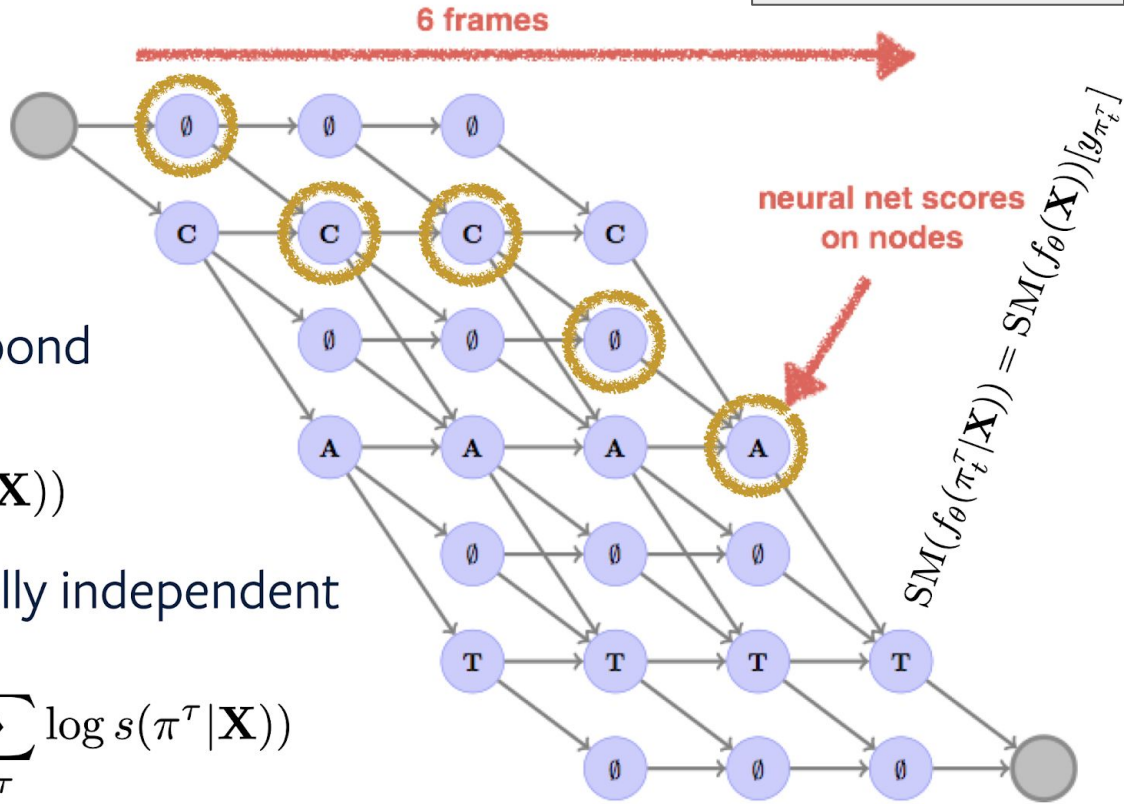
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



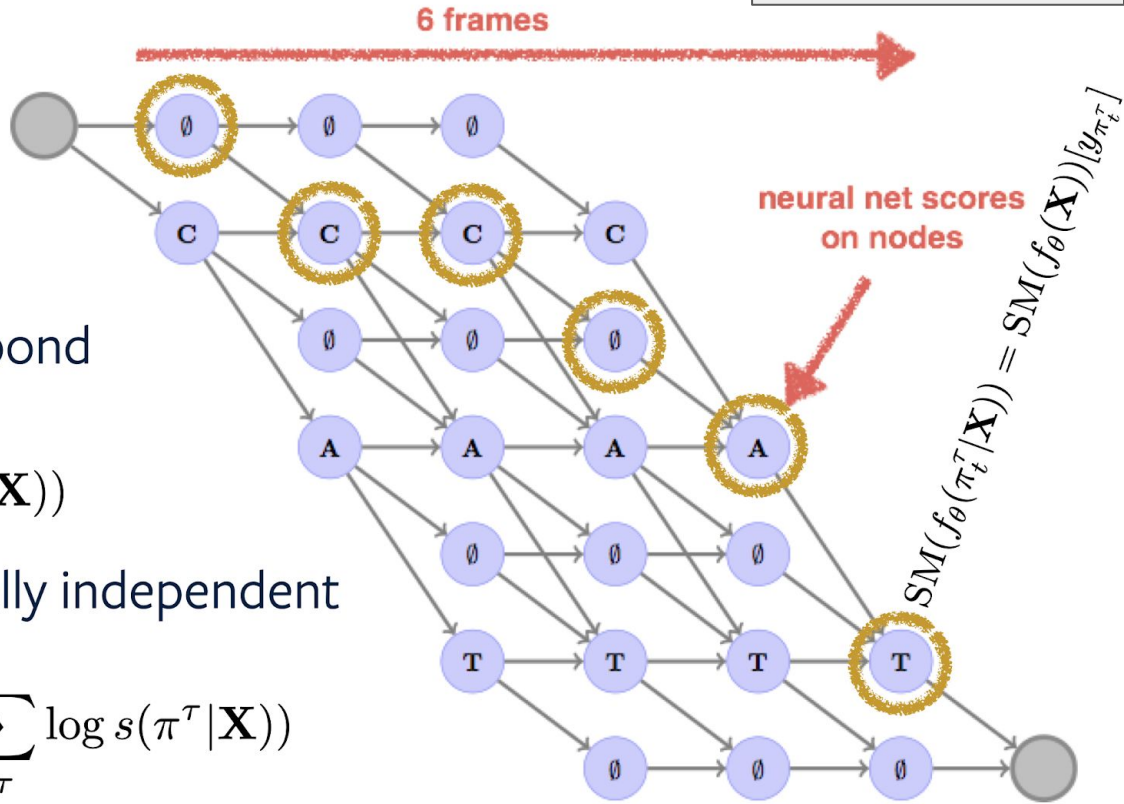
CTC explained

scores = frame normal

- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



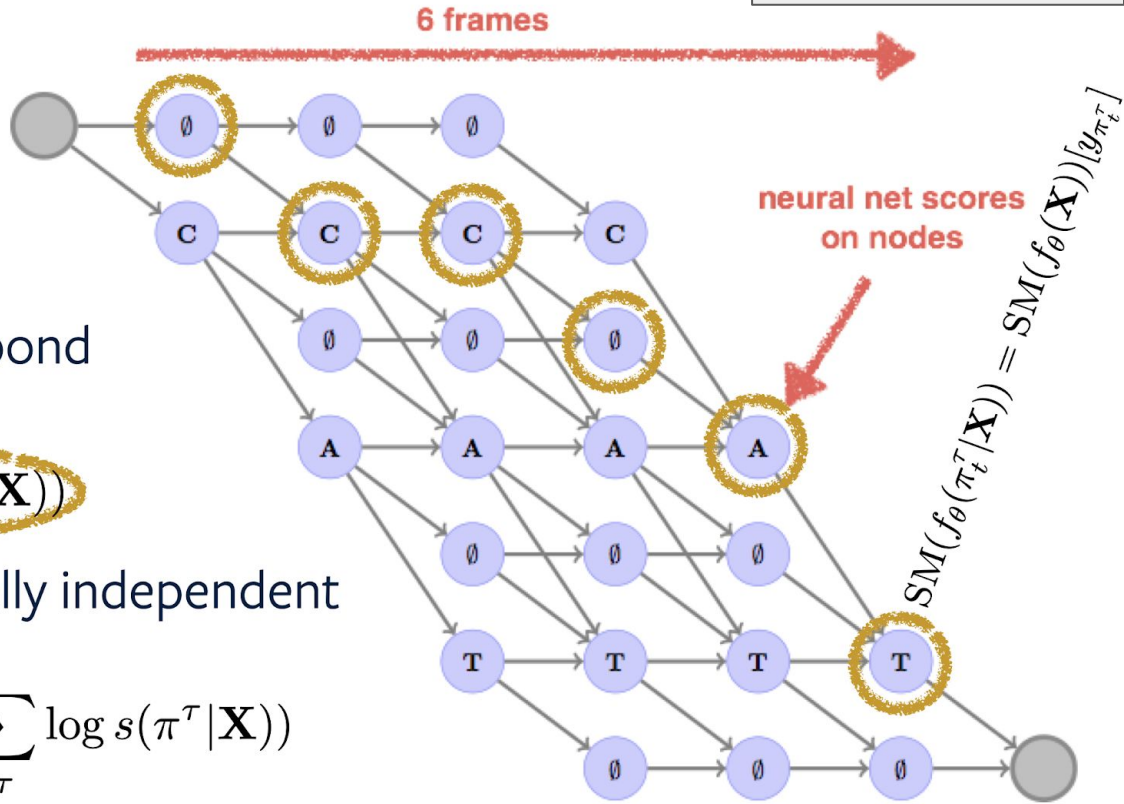
CTC explained

scores = frame normal

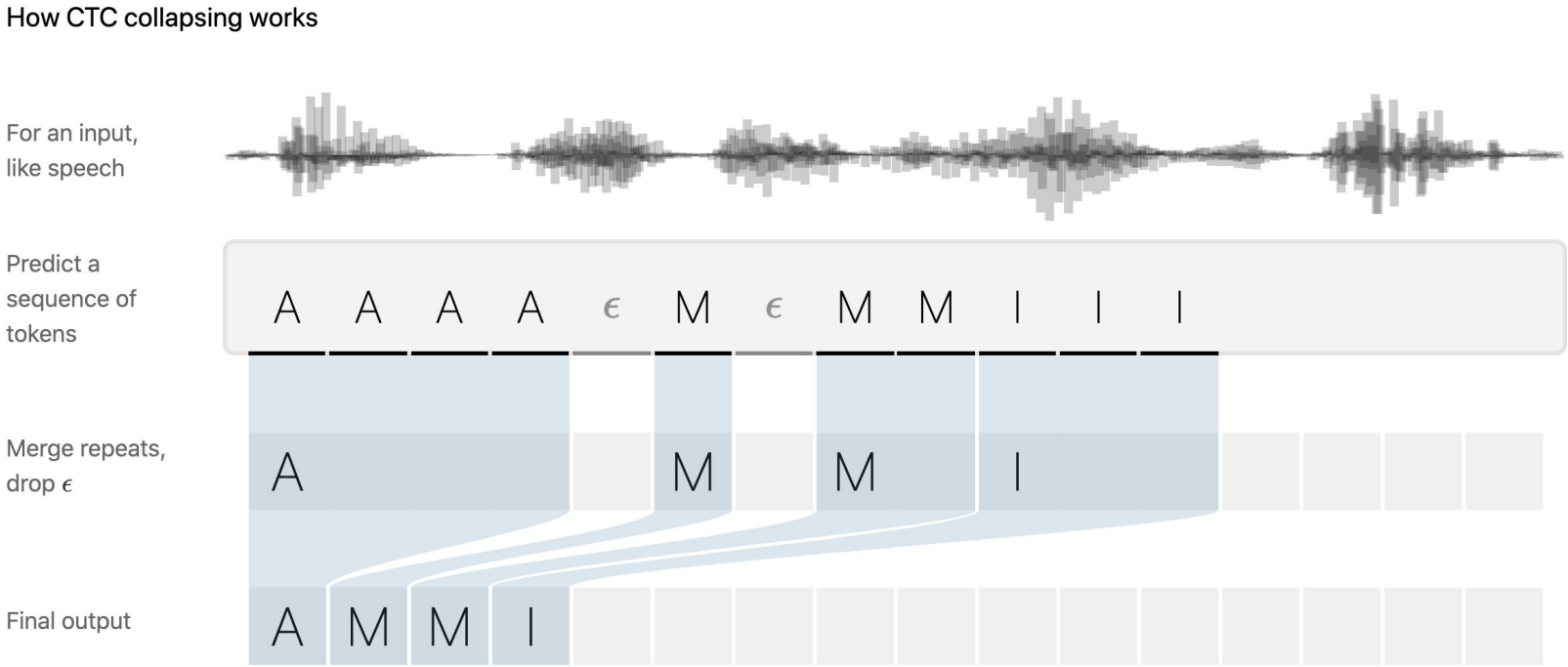
- Blank labels = ø
- Say "cat" is the target, token dictionary is {a, c, t}
- Just sum the framewise scores of labels that correspond to the transcription τ:

$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

- Note: Tokens are conditionally independent given the input
- Loss (log-sum-exp): $-\exp(\sum_\tau \log s(\pi^\tau|\mathbf{X}))$



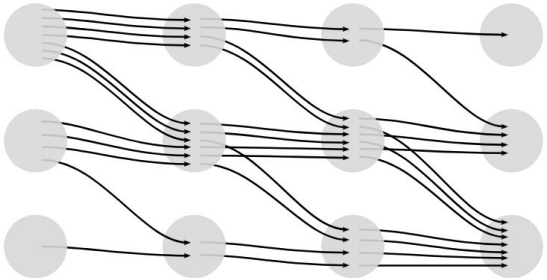
Explained by Awni Hannun: https://distill.pub/2017/ctc/



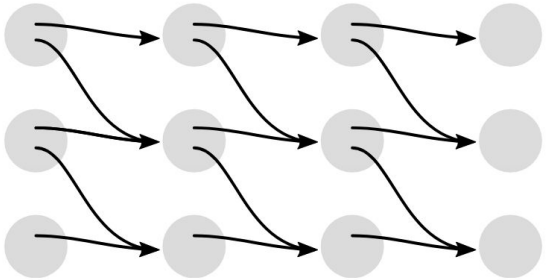
CTC: Loss function

$$p(Y \mid X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t \mid X)$$

The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.



Summing over all alignments can be very expensive.



Dynamic programming merges alignments, so it's much faster.

CTC: Computing it by dynamic prog.

- We can have an ϵ before or after any token in Y, so we work with Z, two cases.

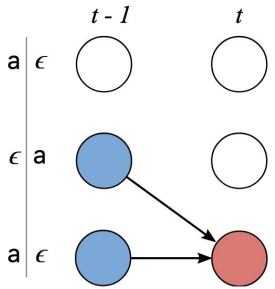
$$Z = [\epsilon, y_1, \epsilon, y_2, \ldots, \epsilon, y_U, \epsilon]$$

- Case 1: $\alpha_{s,t} = (\alpha_{s-1,t-1} + \alpha_{s,t-1}) \cdot p_t(z_s \mid X)$

Can't skip s-1:
$z_s = z_{s-2}$

The CTC probability of the two valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.

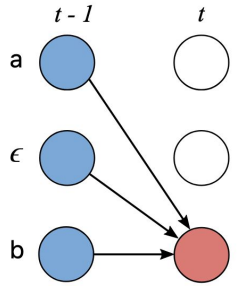


- Case 2: $\alpha_{s,t} = (\alpha_{s-2,t-1} + \alpha_{s-1,t-1} + \alpha_{s,t-1}) \; p_t(z_s \mid X)$

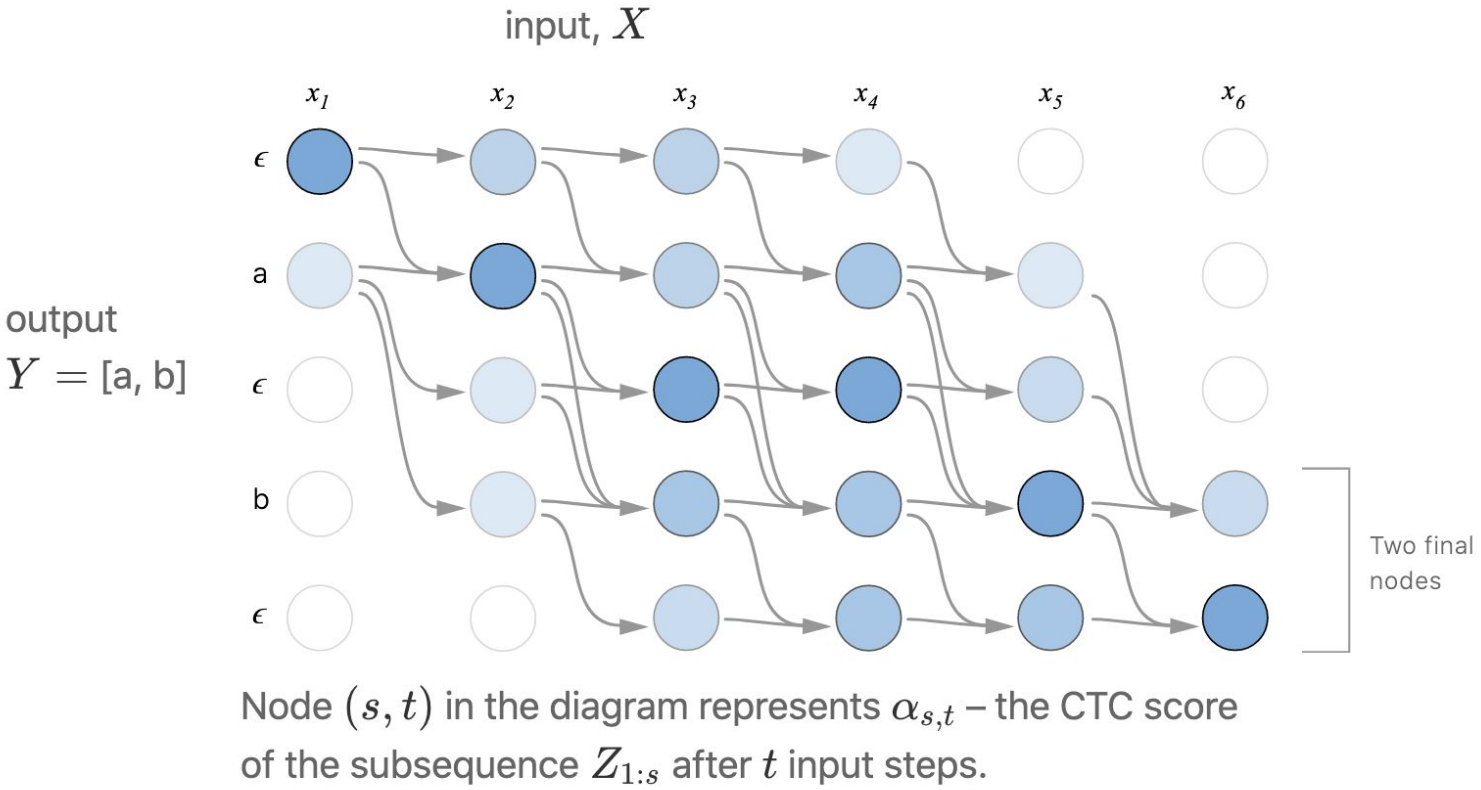
Can skip s-1:
z_{s-1} is an ϵ

The CTC probability of the three valid subsequences after $t-1$ input steps.

The probability of the current character at input step t.



CTC: Result of the dynamic prog.



Node (s, t) in the diagram represents $\alpha_{s,t}$ – the CTC score of the subsequence $Z_{1:s}$ after t input steps.

The first ASR CTC paper

Graves et al. 2006

Table 1. **Label Error Rate (LER) on TIMIT**. CTC and hybrid results are means over 5 runs, ± standard error. All differences were significant ($p < 0.01$), except between weighted error BLSTM/HMM and CTC (best path).

System	LER
Context-independent HMM	38.85 %
Context-dependent HMM	35.21 %
BLSTM/HMM	33.84 ± 0.06 %
Weighted error BLSTM/HMM	31.57 ± 0.06 %
CTC (best path)	31.47 ± 0.21 %
CTC (prefix search)	30.51 ± 0.19 %

There were more than 900,000 parameters in total.

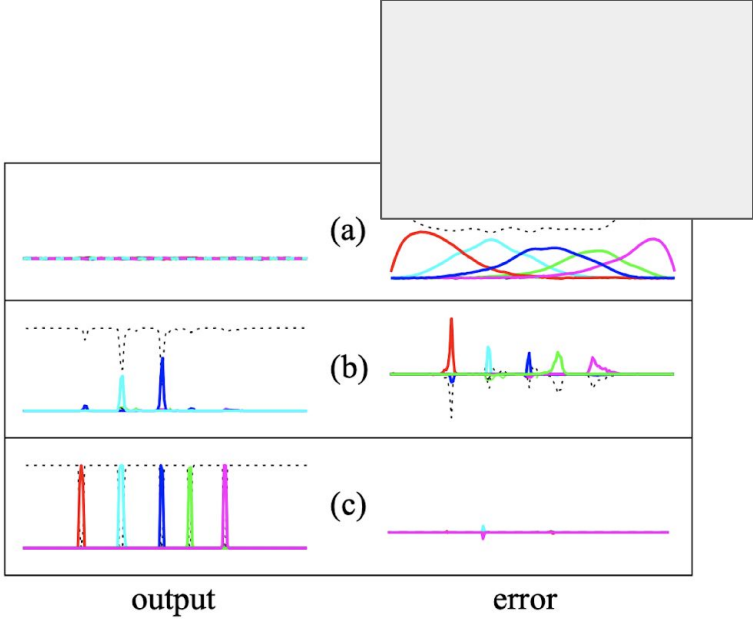


Figure 4. **Evolution of the CTC Error Signal During Training**. The left column shows the output activations for the same sequence at various stages of training (the dashed line is the 'blank' unit); the right column shows the corresponding error signals. Errors above the horizontal axis act to increase the corresponding output activation and those below act to decrease it. (a) Initially the network has small random weights, and the error is determined by the target sequence only. (b) The network begins to make predictions and the error localises around them. (c) The network strongly predicts the correct labelling and the error virtually disappears.

Some key CTC papers

- Graves & Jaitly 2014: first end-to-end ASR system with CTC.
- Hannun et al. 2014: Deep Speech, first pass decoding with an LM with CTC.
- Amodei et al. 2015: Deep Speech 2, even larger scale training (12k hours).
- Soltau et al. 2017: direct-to-word CTC (125k hours).

Limitations of CTC

- The blank label (ø or ϵ) makes the graph a bit complicated, models both “garbage” (silences) and repetitions.
- Y cannot be longer than X
- The alignment obtained is not great (spiky, see later example too)
- Conditional independence assumption (what makes it efficient): $P(a_t|X)$ is independent from $P(a_{t-1}|X)$

If we predict an ‘A’ as the first letter then the suffix ‘AA’ should get much more probability than ‘riple A’. If we predict ‘t’ first, the opposite should be true.

AutoSeg Criterion

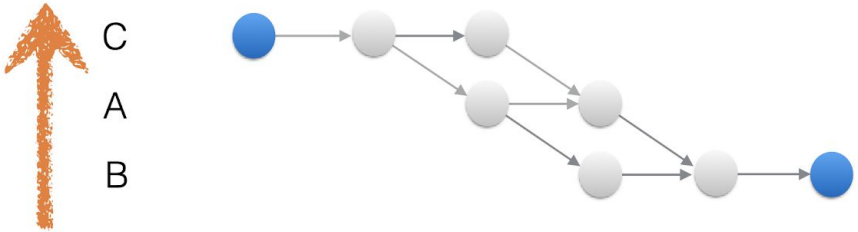
- Say "cab" is the target, token dictionary is {a, b, c}, over 4 frames, can be written caab, ccab, cabb, etc..
- AutoSeGmentation (ASG) criterion, globally (sequence) normalized!

$$s_{\oplus}(\pi^{\tau}|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t^{\tau}|\mathbf{X}) \qquad s_{\ominus}(\pi|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t|\mathbf{X})$$

$$L = -\exp(\sum_{\tau} \log s_{\oplus}(\pi^{\tau}|\mathbf{X})) + \exp(\sum_{\forall \pi} \log s_{\ominus}(\pi|\mathbf{X}))$$

AutoSeg Criterion

- Say "cab" is the target, token dictionary is {a, b, c}, over 4 frames, can be written caab, ccab, cabb, etc..
- AutoSeGmentation (ASG) criterion, globally (sequence) normalized!



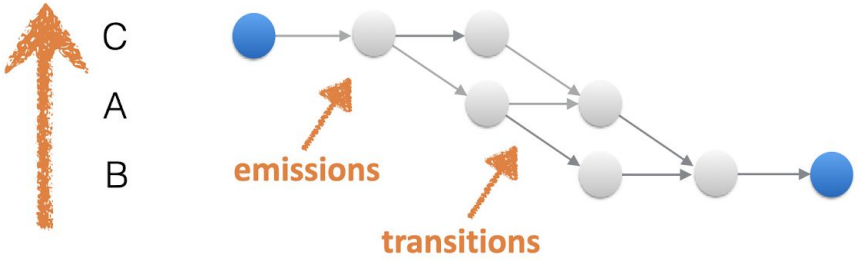
$$s_{\oplus}(\pi^{\tau}|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t^{\tau}|\mathbf{X})$$

$$s_{\ominus}(\pi|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t|\mathbf{X})$$

$$L = -\exp(\sum_{\tau} \log s_{\oplus}(\pi^{\tau}|\mathbf{X})) + \exp(\sum_{\forall\pi} \log s_{\ominus}(\pi|\mathbf{X}))$$

AutoSeg Criterion

- Say "cab" is the target, token dictionary is {a, b, c}, over 4 frames, can be written caab, ccab, cabb, etc..
- AutoSeGmentation (ASG) criterion, globally (sequence) normalized!



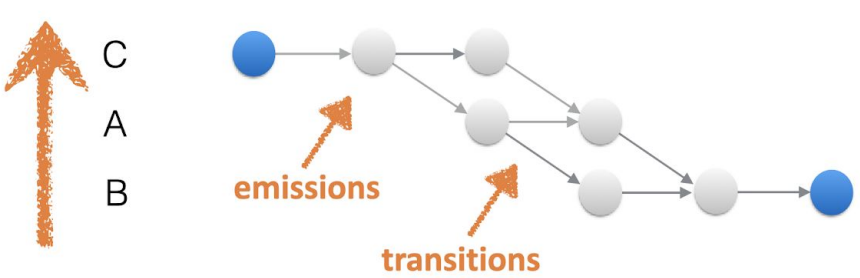
$$s_{\oplus}(\pi^{\tau}|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t^{\tau}|\mathbf{X})$$

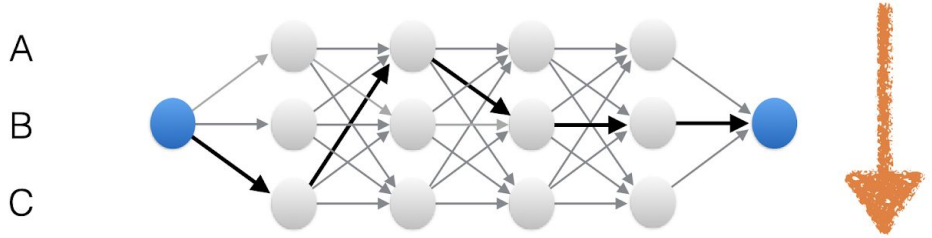
$$s_{\ominus}(\pi|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t|\mathbf{X})$$

$$L = -\exp(\sum_{\tau} \log s_{\oplus}(\pi^{\tau}|\mathbf{X})) + \exp(\sum_{\forall \pi} \log s_{\ominus}(\pi|\mathbf{X}))$$

AutoSeg Criterion

- Say "cab" is the target, token dictionary is {a, b, c}, over 4 frames, can be written caab, ccab, cabb, etc..
- AutoSeGmentation (ASG) criterion, globally (sequence) normalized!



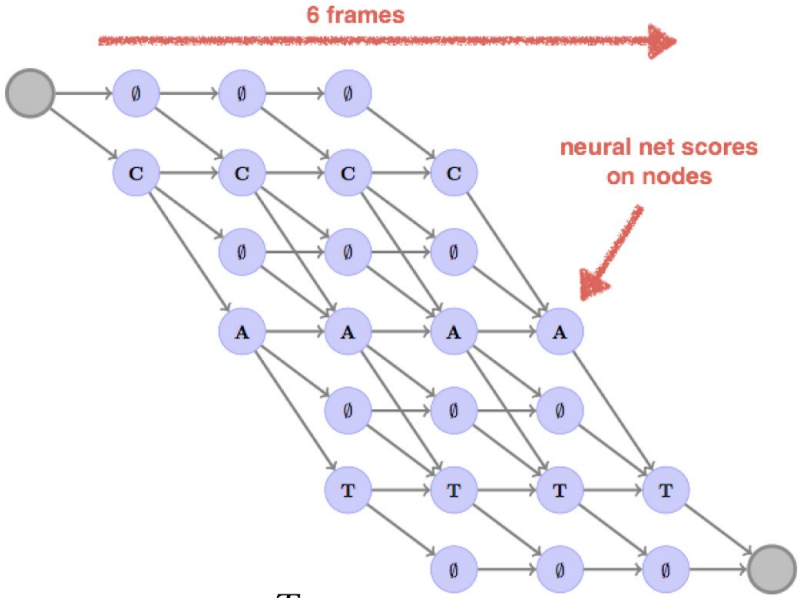


$$s_{\oplus}(\pi^{\tau}|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t^{\tau}|\mathbf{X})$$

$$s_{\ominus}(\pi|\mathbf{X}) = \sum_{t=1}^{T} f_{\theta}(\pi_t|\mathbf{X})$$

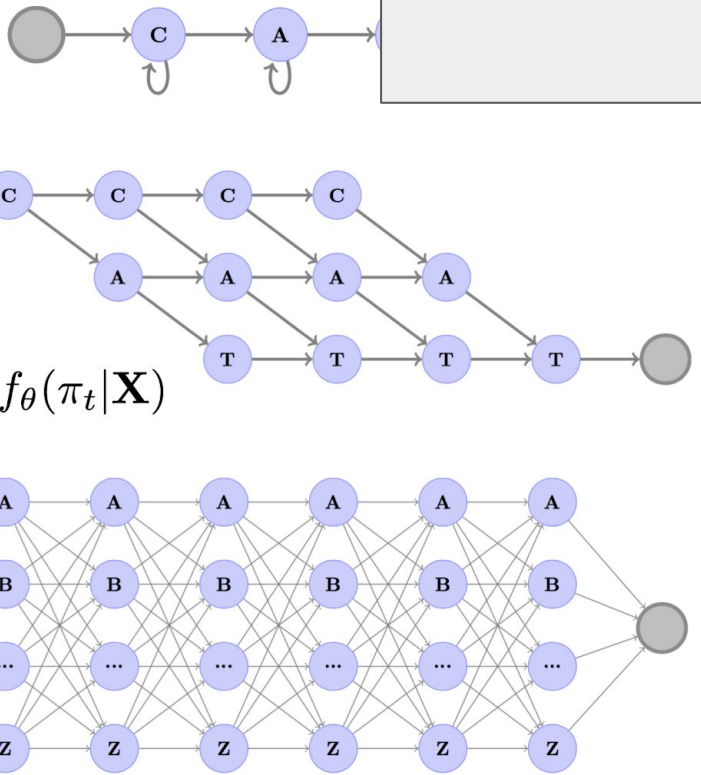
$$L = -\exp(\sum_{\tau} \log s_{\oplus}(\pi^{\tau}|\mathbf{X})) + \exp(\sum_{\forall \pi} \log s_{\ominus}(\pi|\mathbf{X}))$$

CTC compared to ASG



$$s(\pi^\tau|\mathbf{X}) = \sum_{t=1}^{T} \textsc{SoftMax}(f_\theta(\pi_t^\tau|\mathbf{X}))$$

$$\mathsf{L} = -\exp(\sum_{\tau} \log s(\pi^\tau|\mathbf{X}))$$



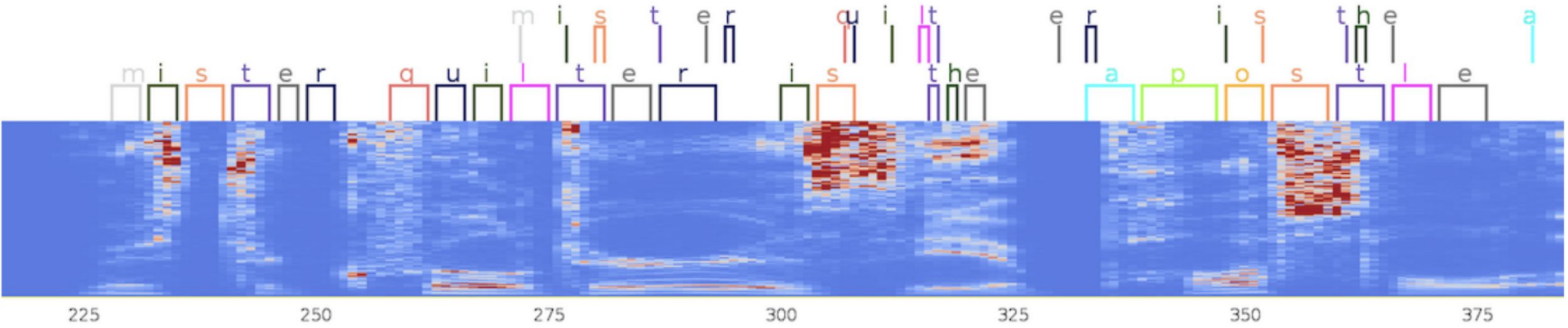
$$s(\pi|\mathbf{X}) = \sum_{t=1}^{T} f_\theta(\pi_t|\mathbf{X})$$

$$\mathsf{L} = -\exp(\sum_{\tau} \log s(\pi^\tau|\mathbf{X})) + \exp(\sum_{\forall\pi} \log s(\pi|\mathbf{X}))$$

ASG vs. CTC

Liptchinsky et al. 2017

	WSJ eval92	LibriSpeech test-clean	LibriSpeech test-other
(Povey et al., 2016)	4.3	-	-
(Panayotov et al., 2015)	3.9	5.5	14.0
(Peddinti et al., 2015b)	-	4.8	-
(Chan and Lane, 2015b)	3.5	-	-
(Ko et al., 2015)†	-	-	12.5
(Zhou et al., 2018)†	5.5	5.7	15.2
(Amodei et al., 2016)⋆	3.6	5.3	13.3
(Zeyer et al., 2018)	-	4.8	14.7
(Zeyer et al., 2018) (*LSTM LM*)	-	3.8	12.8
this paper (CTC)	-	5.1	16.0
this paper (ASG)	5.6	4.8	14.5

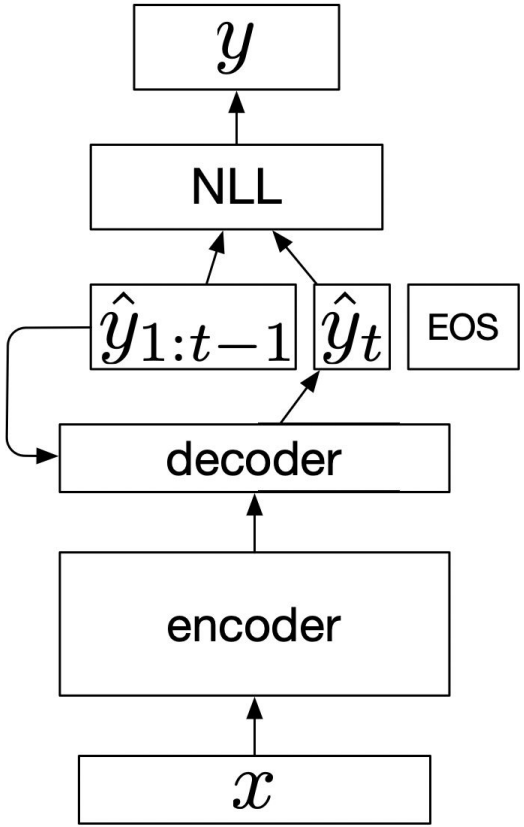


Limitations of ASG

- Doesn’t scale as well as CTC with the number of classes (the denominator is expensive)

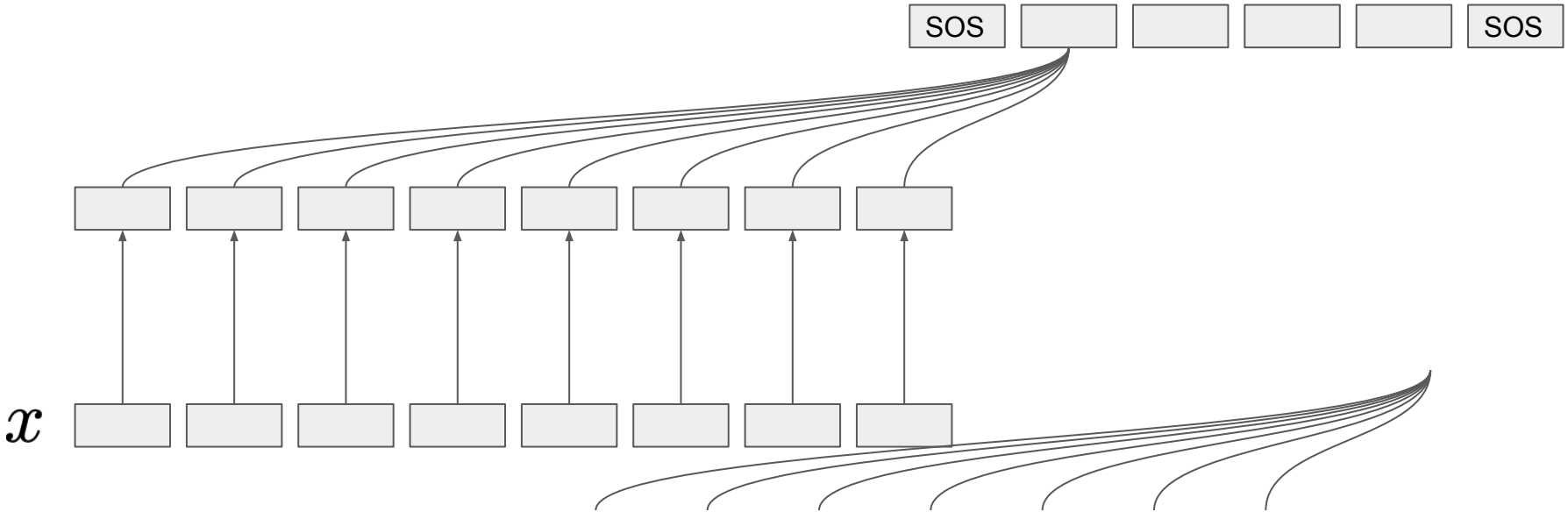
Other deep learning losses: seq2seq

- x, y, yhat are sequences
- EOS is the end of sequence token, also SOS: start of sequence
- In machine translation:
  - Bahdanau et al. 2014
  - Sutskever et al. 2014
- In ASR:
  - Chan et al. 2015
  - Chorowski et al. 2015



Seq2seq training

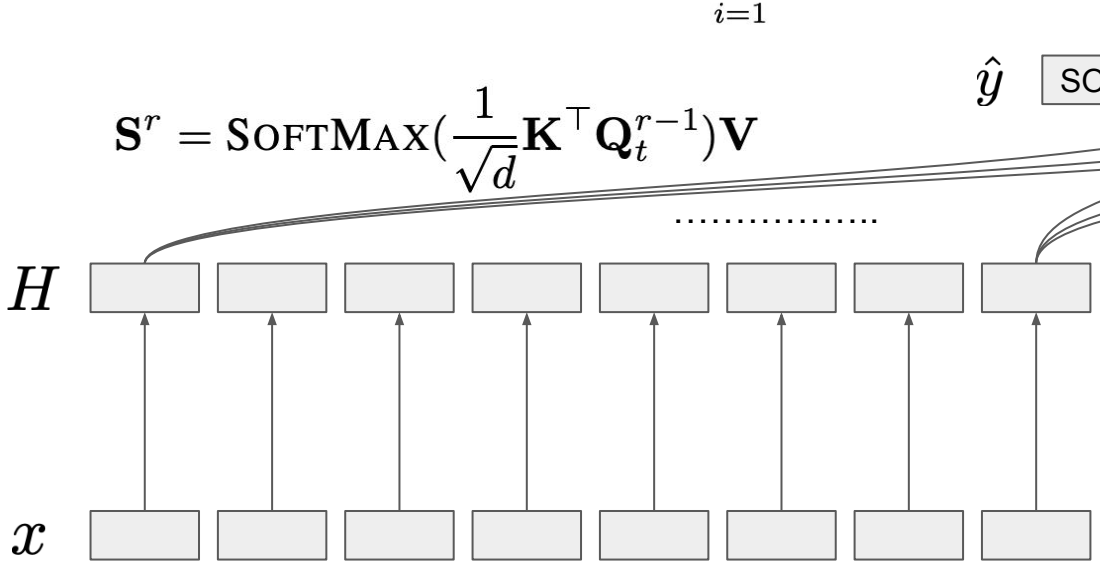
$$\mathbf{S}^r = \textsc{SoftMax}(\frac{1}{\sqrt{d}}\mathbf{K}^\top \mathbf{Q}_t^{r-1})\mathbf{V}$$

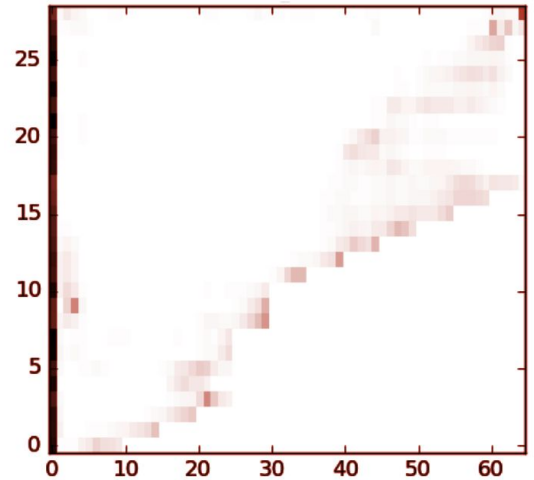


Seq2seq training

$$p(y_1, ..., y_n) = \prod_{i=1}^{n} p(y_i \mid y_0, ..., y_{i-1}, \mathbf{H}^{L_e})$$

$$\mathbf{S}^r = \textsc{SoftMax}(\frac{1}{\sqrt{d}}\mathbf{K}^\top \mathbf{Q}_t^{r-1})\mathbf{V}$$





Word Pieces / Sentence Pieces

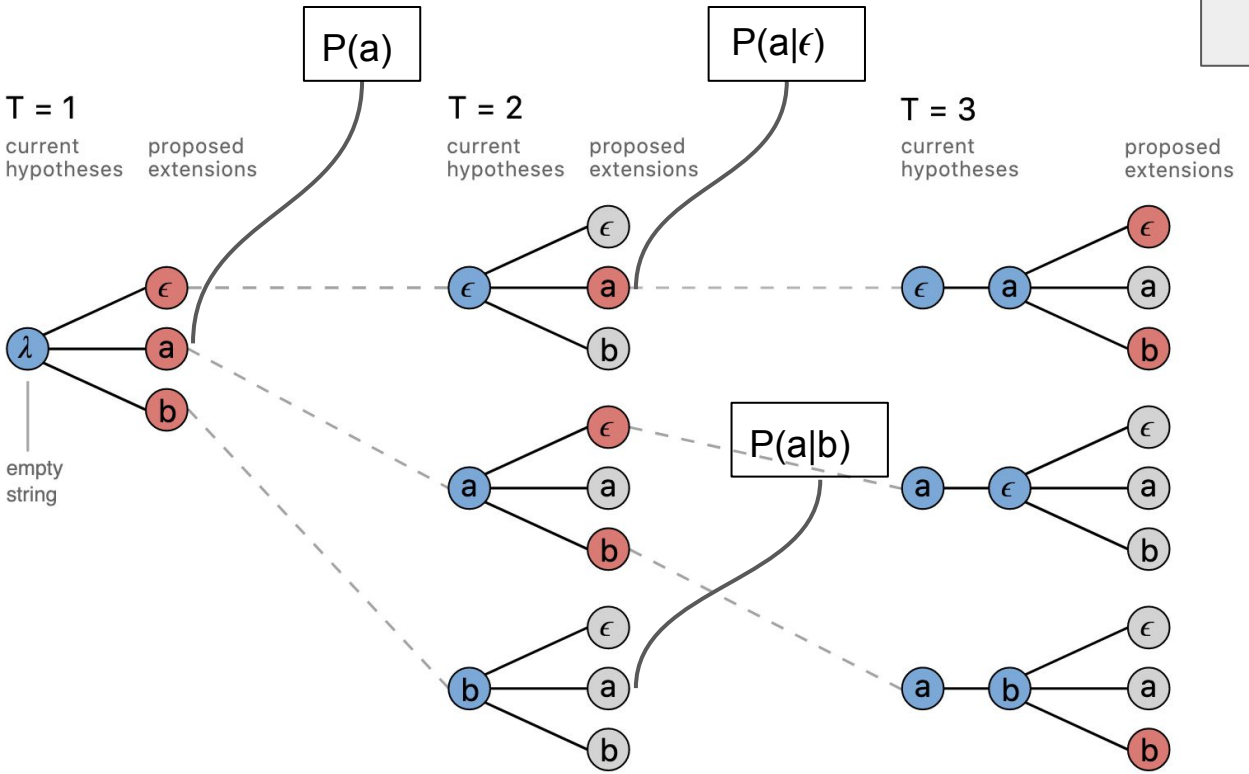
- The longer the token unit we can use, the more we can stride in the AM, and the faster the decoding is.
- Backs-off to letters, so there is no OOV! And most frequent words are 1 word piece.
- Inconvenient: multiple ways to write the same word (like phonemes).

Subwords (_ means spaces)	Vocabulary id sequence
_Hell/o/_world	13586 137 255
_H/ello/_world	320 7363 255
_He/llo/_world	579 10115 255
_/He/l/l/o/_world	7 18085 356 356 137 255
_H/el/l/o/_/world	320 585 356 137 7 12295

Table 1: Multiple subword sequences encoding the same sentence "Hello World"

Kudo 2018 (see also Sennrich et al. 2016, Shuster 2012)

Beam Search



A standard beam search algorithm with an alphabet of $\{\epsilon, a, b\}$ and a beam size of three.

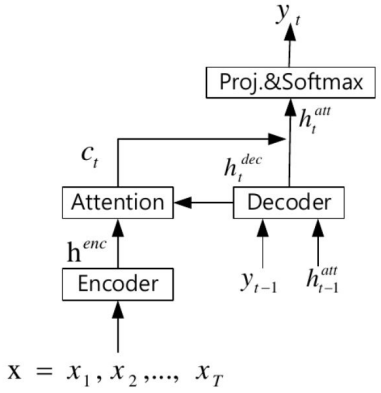
Leveraging LMs

$$\log P_{AM}(\hat{\mathbf{y}}|\mathbf{x}) + \alpha \log P_{LM}(\hat{\mathbf{y}}) + \beta|\hat{\mathbf{y}}|$$

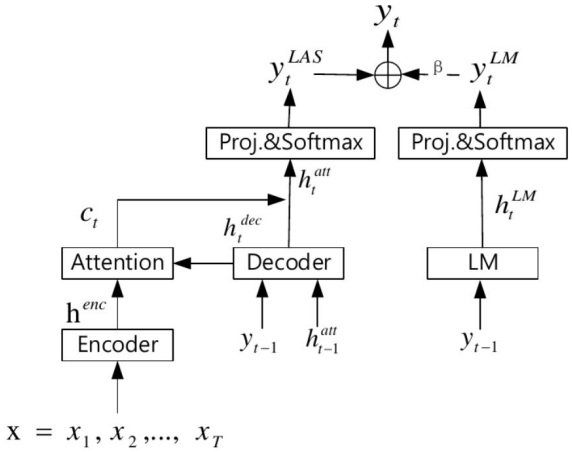
- P(w_t | w_{1..t-1}): doesn't matter if it comes from an ngram, RNN, LSTM, ConvLM, Transformer
  As long as it's autoregressive
- During the decoding process, can only be used when full words (for a word-level LM) gets formed in the beam (cannot score partial words)
- Another technique is rescoring: dump the whole beam (or a lattice) at the end of the utterance and "rescore" with a language model.

$$\log P_{AM}(\hat{\mathbf{y}}|\mathbf{x}) + \alpha_1 \log P_1(\hat{\mathbf{y}}) + \alpha_2 \log P_2(\hat{\mathbf{y}}) + \beta|\hat{\mathbf{y}}|$$

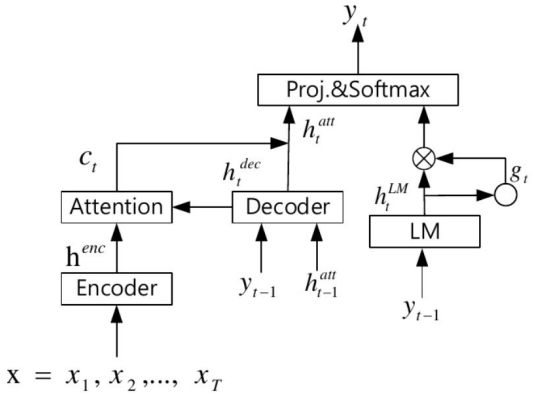
LM fusion for end-to-end models



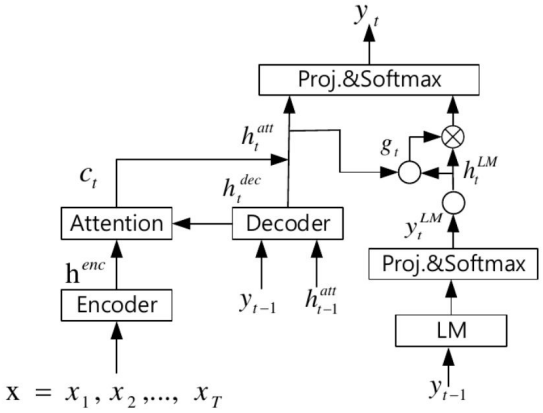
(a.) baseline



(b.) Shallow Fusion



(c.) Deep Fusion



(d.) Cold Fusion