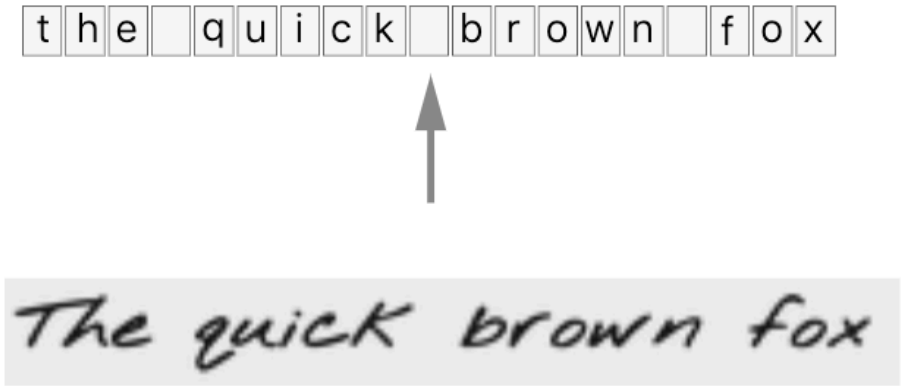
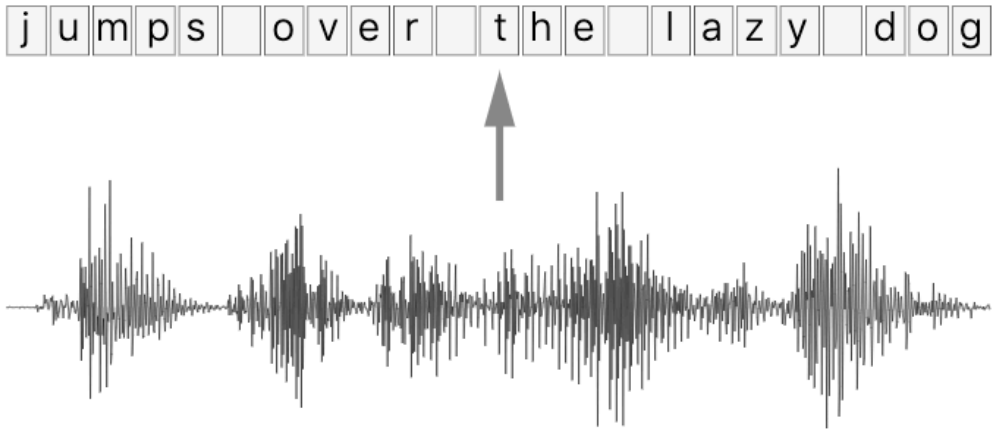
Sequence to Sequence

27.5.2019

Sequence-to-Sequence



Handwriting recognition: The input can be (x, y) coordinates of a pen stroke or pixels in an image.



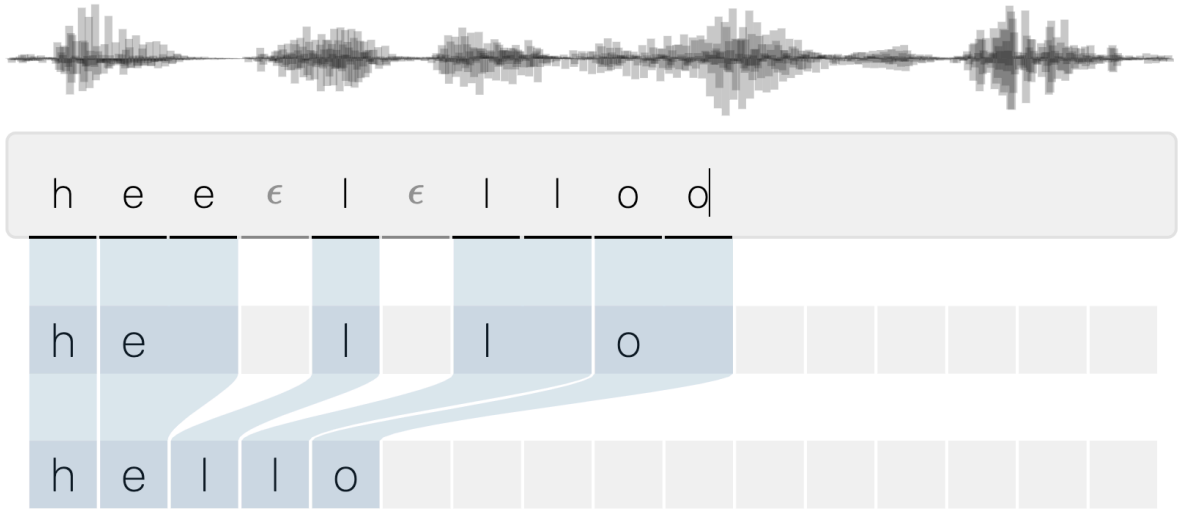
Speech recognition: The input can be a spectrogram or some other frequency based feature extractor.

https://distill.pub/2017/ctc/

Sequence-to-Sequence

- Speech recognition
- Handwriting recognition
- Machine translation
- Speech synthesis (text to speech)
- Summarization
- Text generation

Speech Recognition: Hybrid HMM-DNN

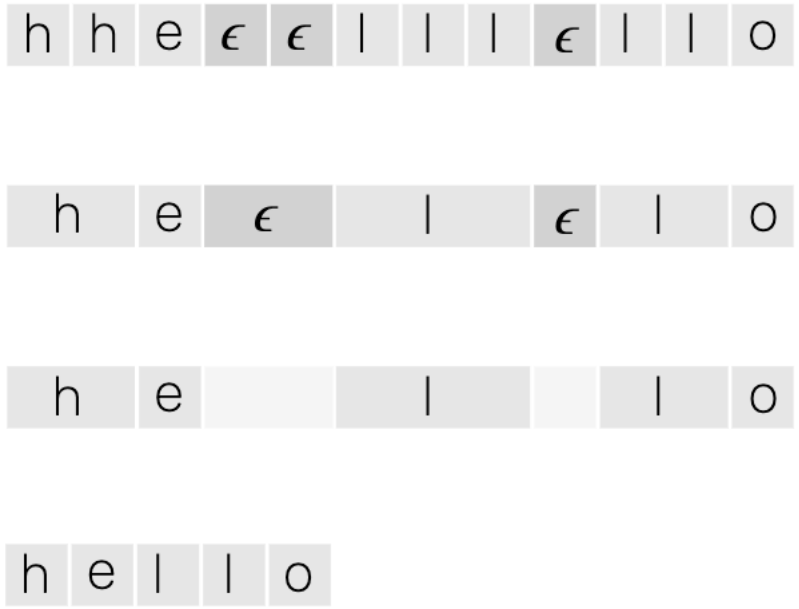


- Use feed-forward net to predict phones
- Use existing HMM/graph structure to form words and sentences
- **Problem: requires alignment for frame-wise training!**
- *(Did not work for translation and generation)*

Connectionist Temporal Classification (CTC)

- Map from $X = [x_1, x_2, \ldots, x_T]$ to $Y = [y_1, y_2, \ldots, y_U]$, where
  - Both X and Y can vary in length.
  - The ratio of the lengths of X and Y can vary.
  - We don’t have an accurate alignment of X and Y
- Key ideas:
  - Use epsilon padding to ensure equal lengths
  - Marginalize over all possible alignments

CTC – Blank (eps) Symbol

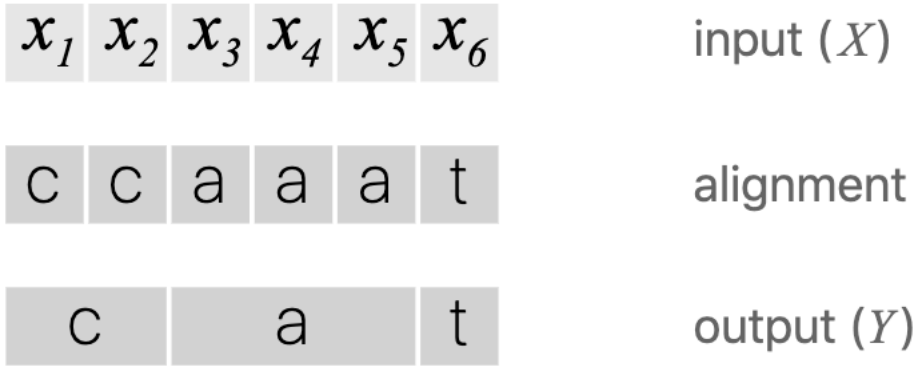


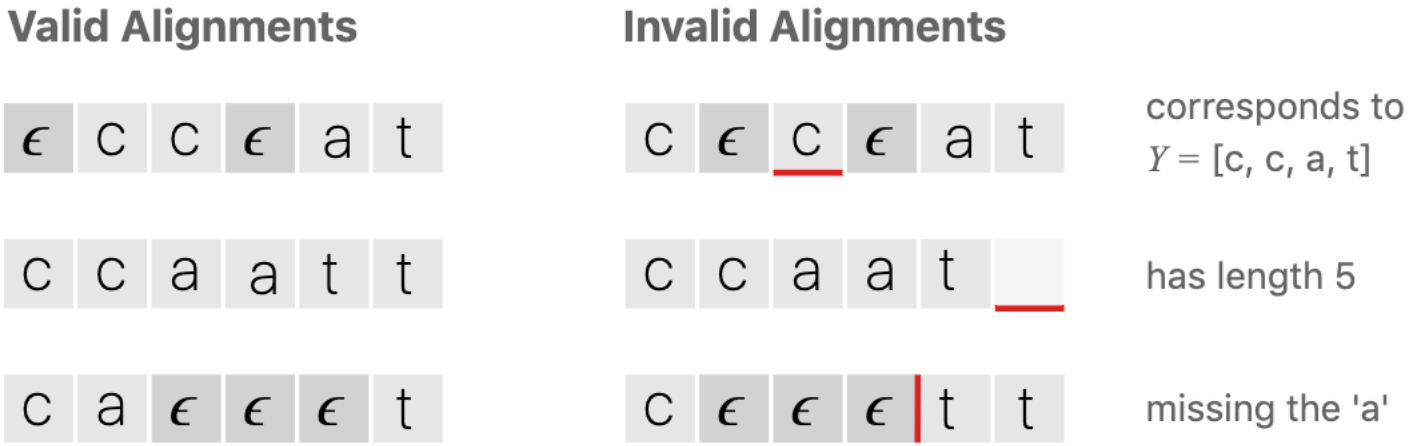
First, merge repeat characters.

Then, remove any ϵ tokens.

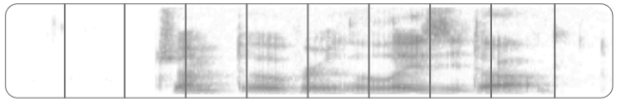
The remaining characters are the output.

Example

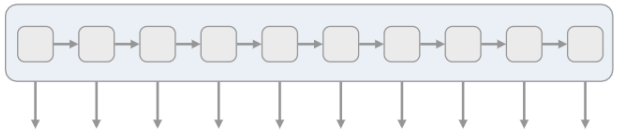

x_1 x_2 x_3 x_4 x_5 x_6
input (X)
c c a a a t
alignment
c a t
output (Y)


Valid Alignments
Invalid Alignments
ϵ c c ϵ a t
c ϵ c ϵ a t
corresponds to
Y = [c, c, a, t]
c c a a t t
c c a a t
has length 5
c a ϵ ϵ ϵ t
c ϵ ϵ ϵ t t
missing the 'a'

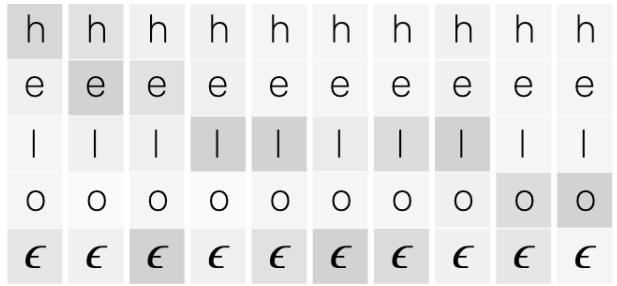
CTC Loss Function – illustrated



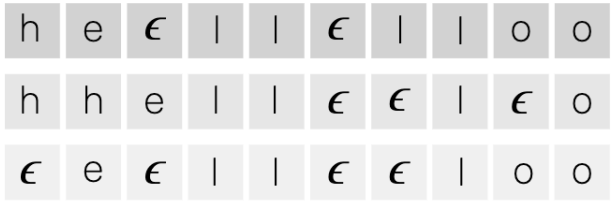
We start with an input sequence, like a spectrogram of audio.



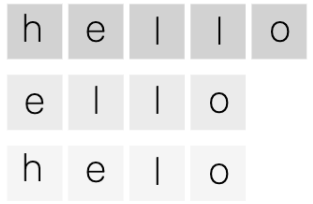
The input is fed into an RNN, for example.



The network gives $p_t(a \mid X)$, a distribution over the outputs {h, e, l, o, ϵ} for each input step.



With the per time-step output distribution, we compute the probability of different sequences



By marginalizing over alignments, we get a distribution over outputs.

CTC Loss Function

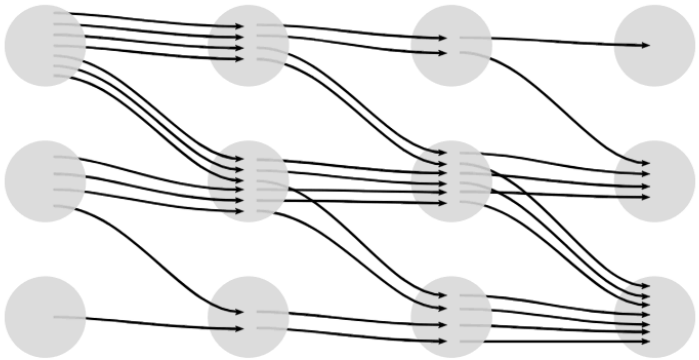
To be precise, the CTC objective for a single (X, Y) pair is:

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

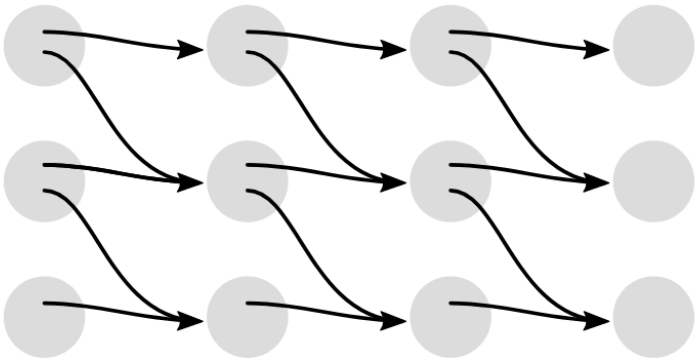
The CTC conditional **probability** **marginalizes** over the set of valid alignments computing the **probability** for a single alignment step-by-step.

Marginalize over all Alignments?

- Valid alignments
  - Retain the original sequence of tokens
  - Optionally insert epsilons between tokens
- Enumerating all alignments is way too expensive → DP!

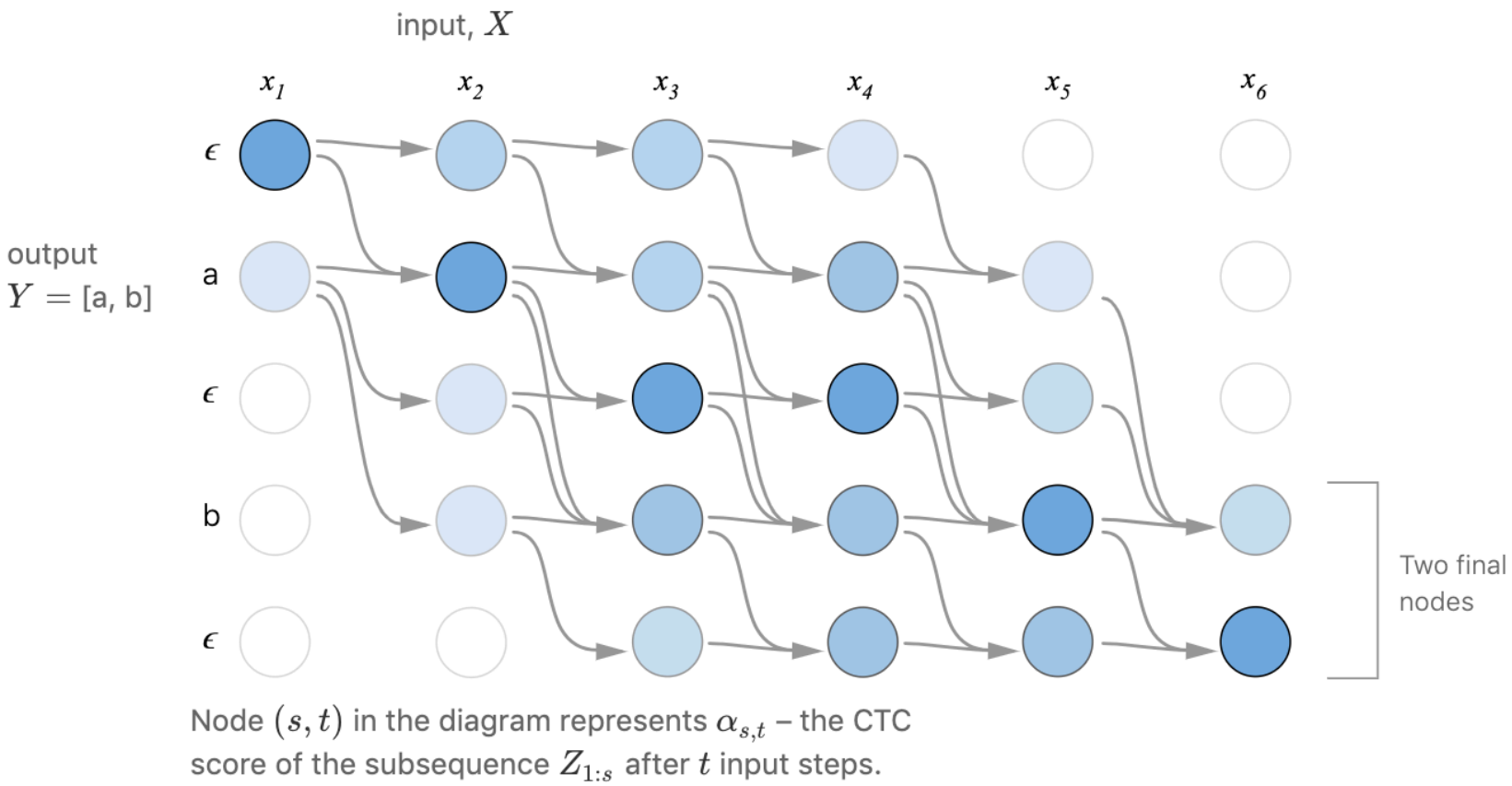


Summing over all alignments can be very expensive.



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

CTC Alignment Graph



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

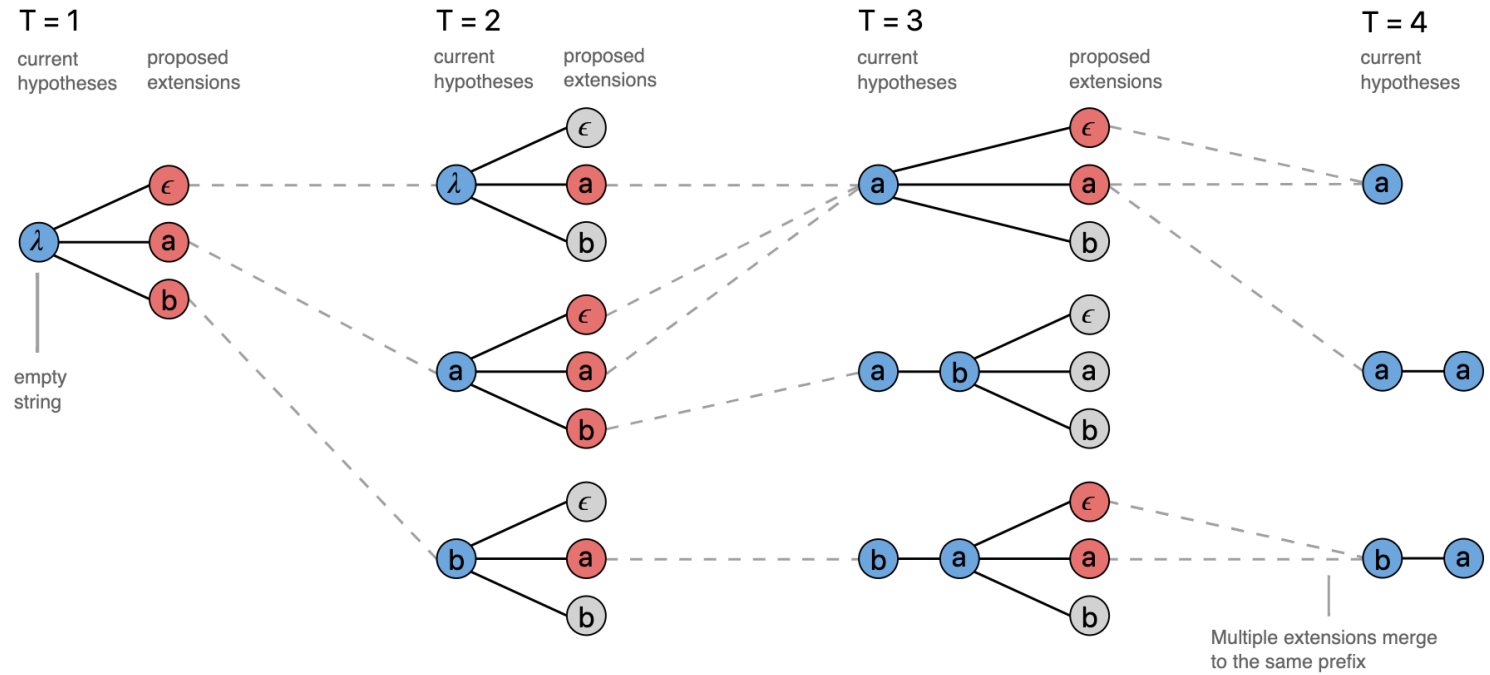
CTC Training

- Loss can be analytically computed → Gradient!
- Maximum Likelihood estimate, including sequence info

$$\sum_{(X,Y)\in\mathcal{D}} -\log\ p(Y \mid X) \quad \rightarrow \text{min.}$$

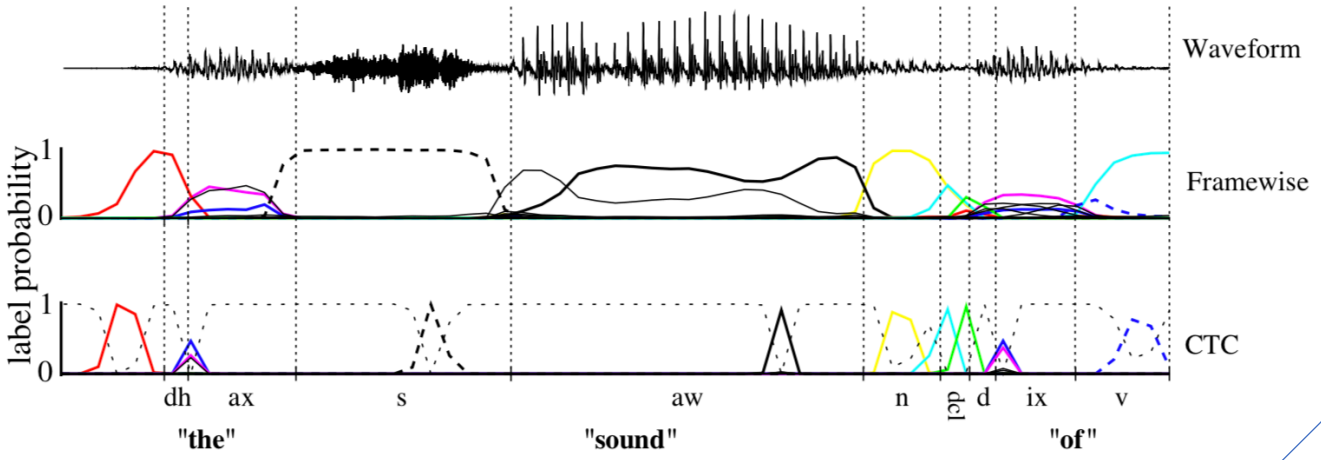
CTC Inference

- How to handle epsilons and repeats?
  - “naive“ way (regexp)
  - Beam search



The CTC beam search algorithm with an output alphabet $\{\epsilon, a, b\}$ and a beam size of three.

CTC for Speech Recognition



„spikey“ activations for non-blanks

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 %

„traditional“ HMM-GMM

„hybrid“ HMM-DNN

First „end-to-end“ in 2006!

CTC References

- „Connectionist Temporal Classification: Labelling Unsegmented Sequence Data with Recurrent Neural Networks.“ A. Graves et al., ICML2006
- https://distill.pub/2017/ctc/
- TensorFlow
  - https://www.tensorflow.org/api_docs/python/tf/nn/ctc_loss
  - https://www.tensorflow.org/api_docs/python/tf/nn/ctc_beam_search_decoder
- cuDNN
  - https://docs.nvidia.com/deeplearning/sdk/cudnn-developer-guide/index.html#cudnnCTCLossAlgo_t