Speech Processing with Neural Networks: CTC

Jan Chorowski

SPEECH RECOGNITION

Speech recognition: prepare a transcription

“Speech to text”:
find a word sequence that matches the recorded utterance

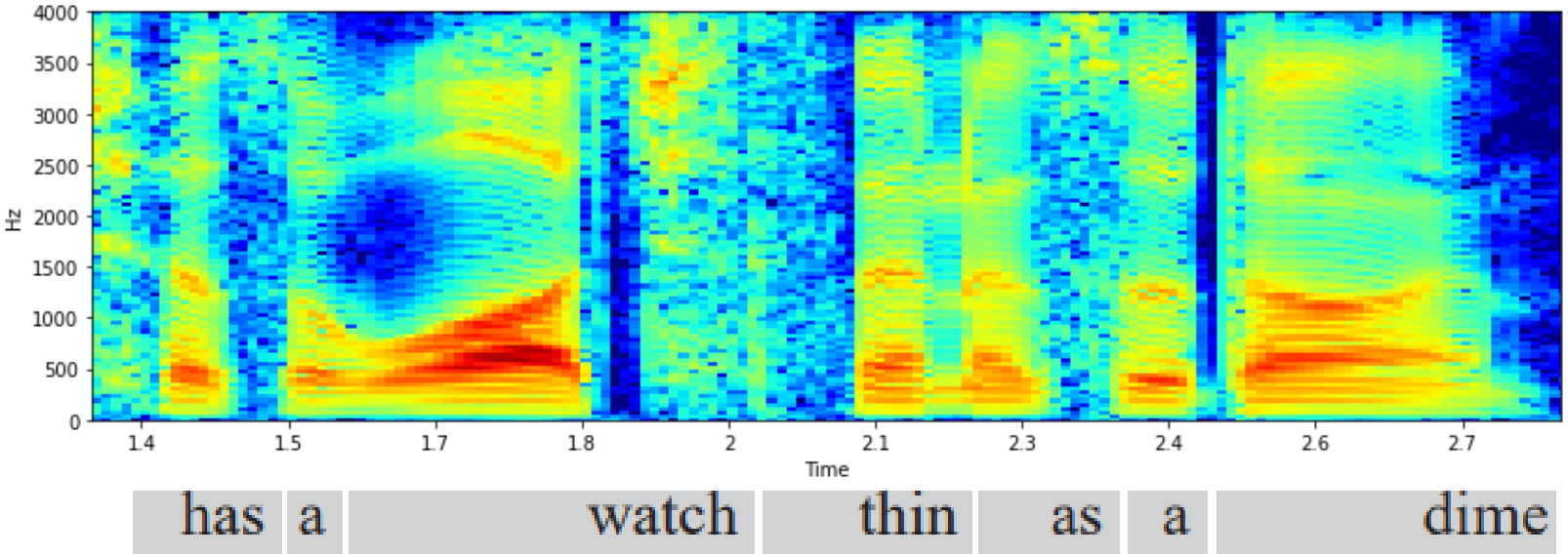
Error measure: Word Error Rate (WER)

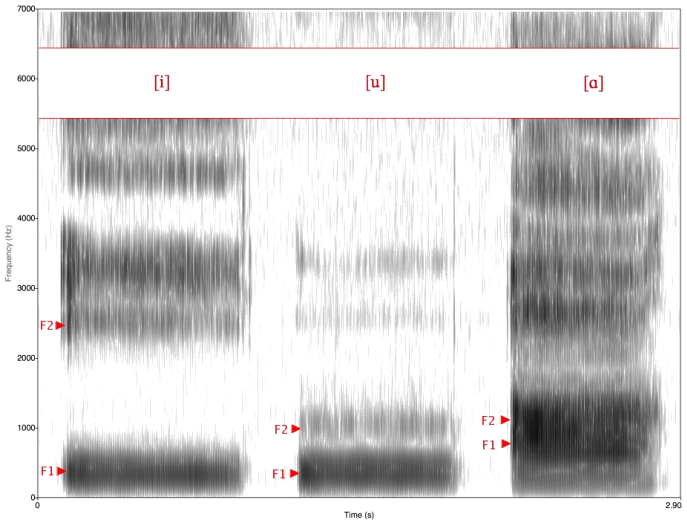
Reference:	It	is	a	sunny		day	
Recognized:	It	was		sunny	all	day	
Word status:	OK	S		D	OK	I	OK

WER = (S + D + I) / len(ref)

What the net should discard/recognize?

- Discard: vocal cord frequency (f0)
- Keep: formant frequency (set position of the mouth, tongue…)





Speech Differences Between Individuals

- Differences between individuals:
  - f0 (base vocal cord frequency)
  - Speed, prosody
  - accents

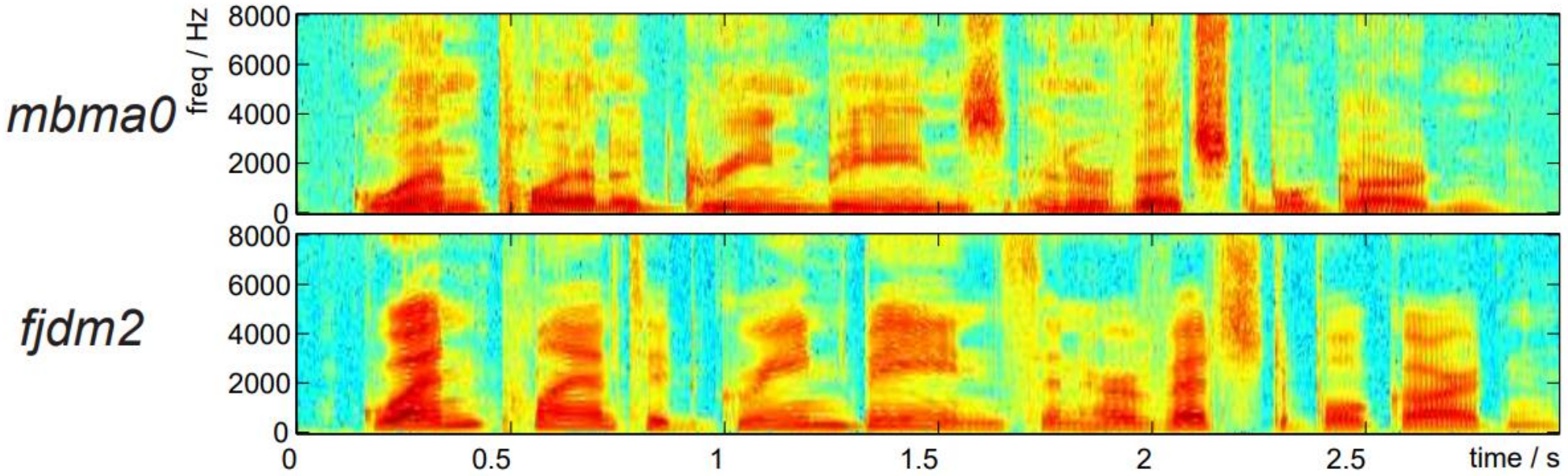
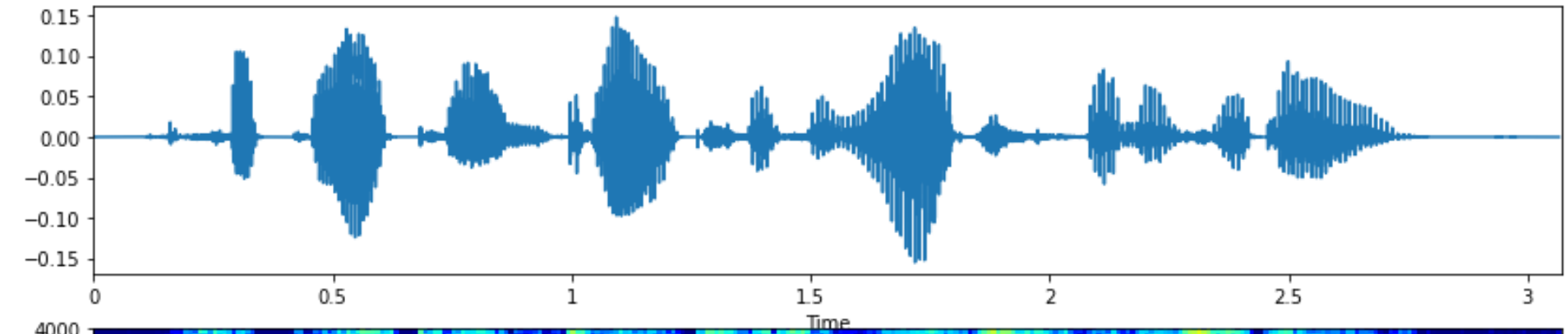
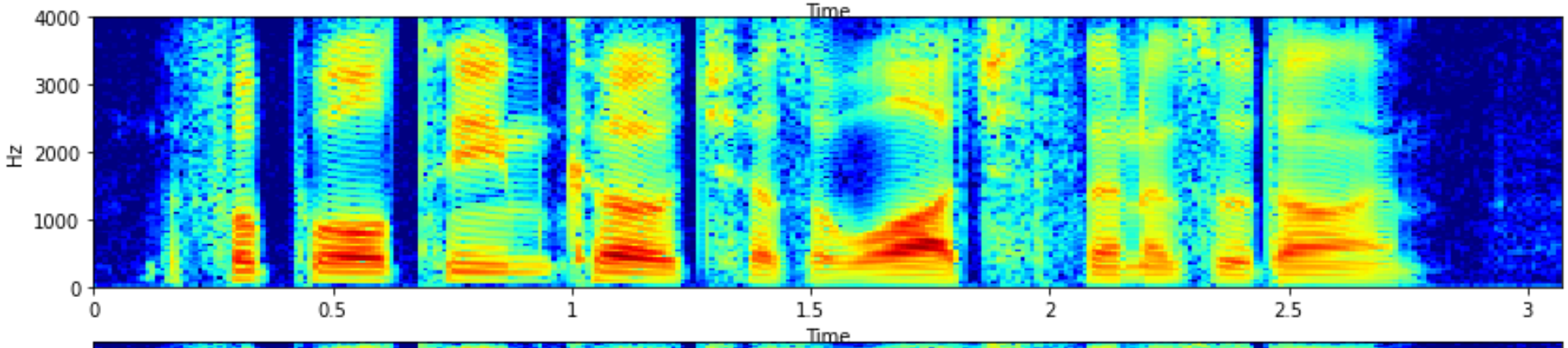


Image from: http://www.ee.columbia.edu/~dpwe/e6820/lectures/L09-asr.pdf

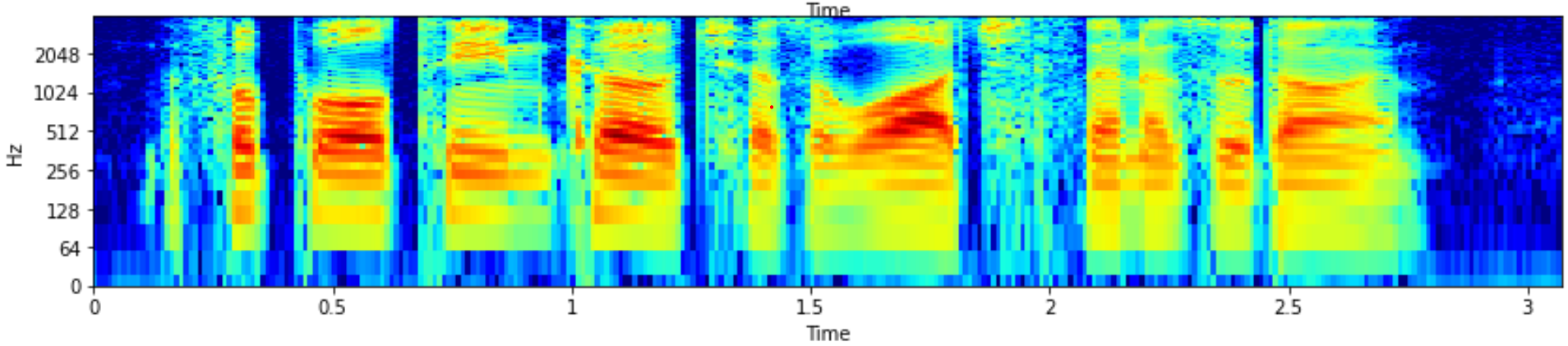
Speech Features for Neural Networks



Raw Audio



STFT



Compress Freq (Y) axis logarithmically (Mel scale)

Speech Features: Intuitions

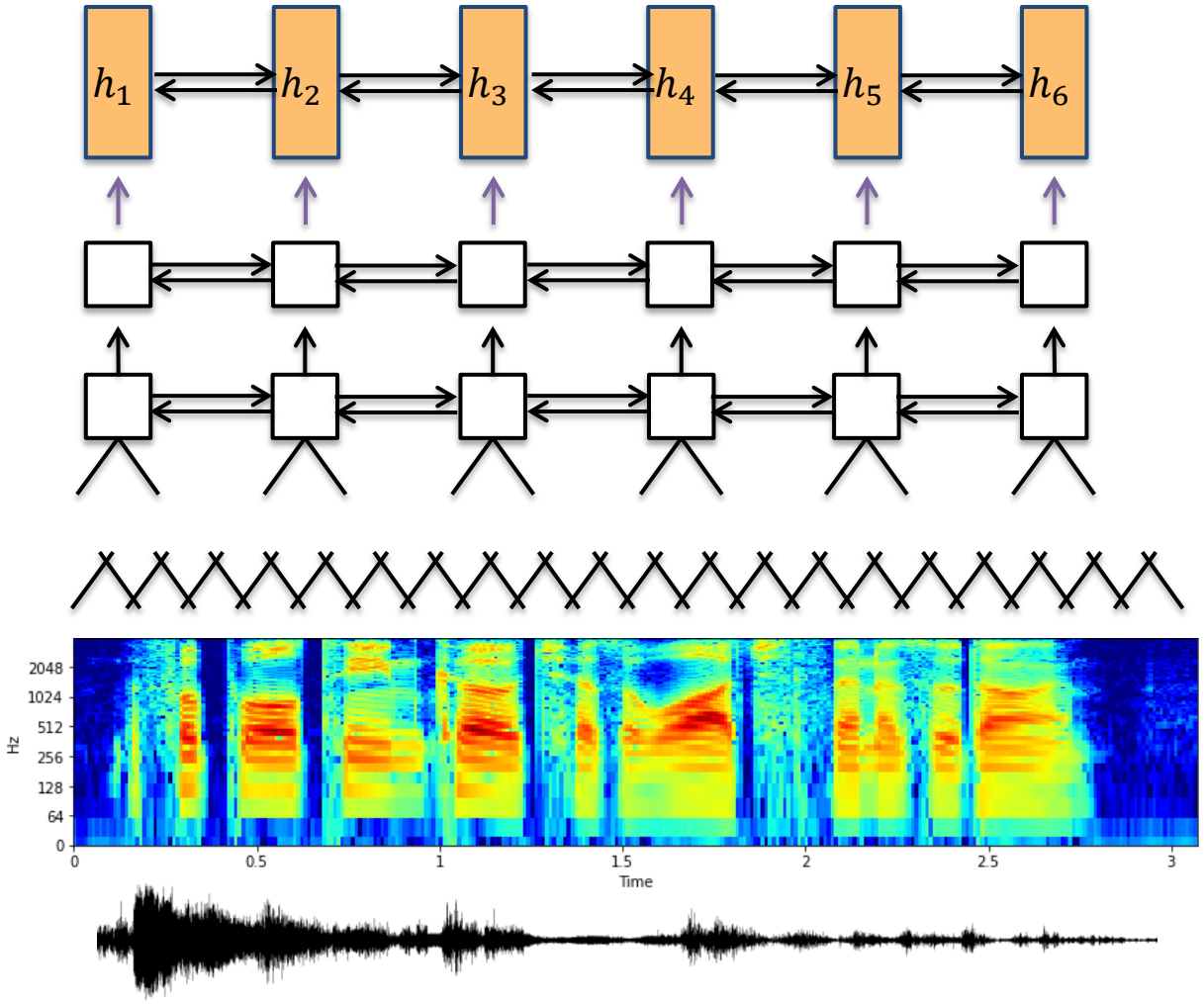
An utterance is represented as a $Time \times Frequency$ array of amplitudes.

Length along time axis varies between utterances.

The number of frequencies is fixed:

- Treat as a 1D signal with $|F|$ channels, apply 1D convolutions.
- Treat as a 2D signal (image) and apply 2D convolutions.

A typical speech processing architecture



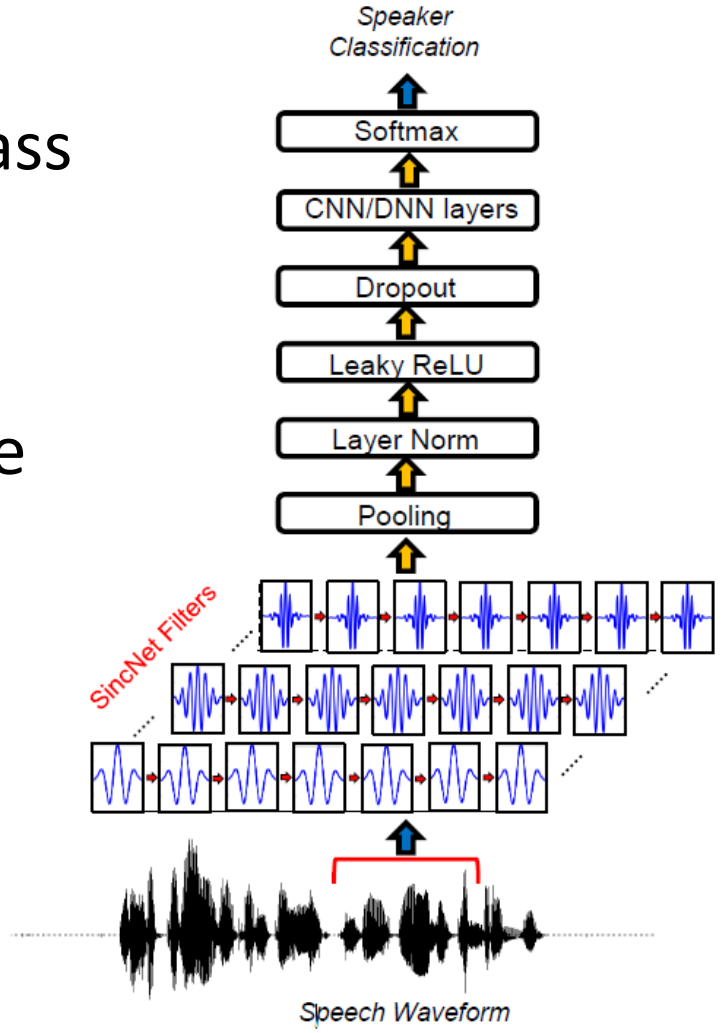
SincNet: Learnable Speech Features

- MFCCs can be computed using typical neural operations (convolutions, multiplications, additions) with specially hardcoded weights
- Learning a raw waveform frontend is however slow, and prone to overfitting because there are may weights to learn.
- Alternative idea: learn a set of filters

SincNet

Idea:
Train a set (bank) of bandpass filters.

The only parameters are the low- and high- cutoff frequencies!



Model Properties

- **Few Parameters**:

F= Number of filters (e.g. 80)
L= Length of each filter (e.g. 100)

Standard CNN
F · L parameters (8k)

SincNet
2F parameters (160)

The number of parameters doesn't depend on L.

We can achieve **high frequency selectivity** without wasting parameters!

https://www.clsp.jhu.edu/wp-content/uploads/2019/06/Ravanelli_JSALT_Presentation.pdf

Speech – Time Alignment

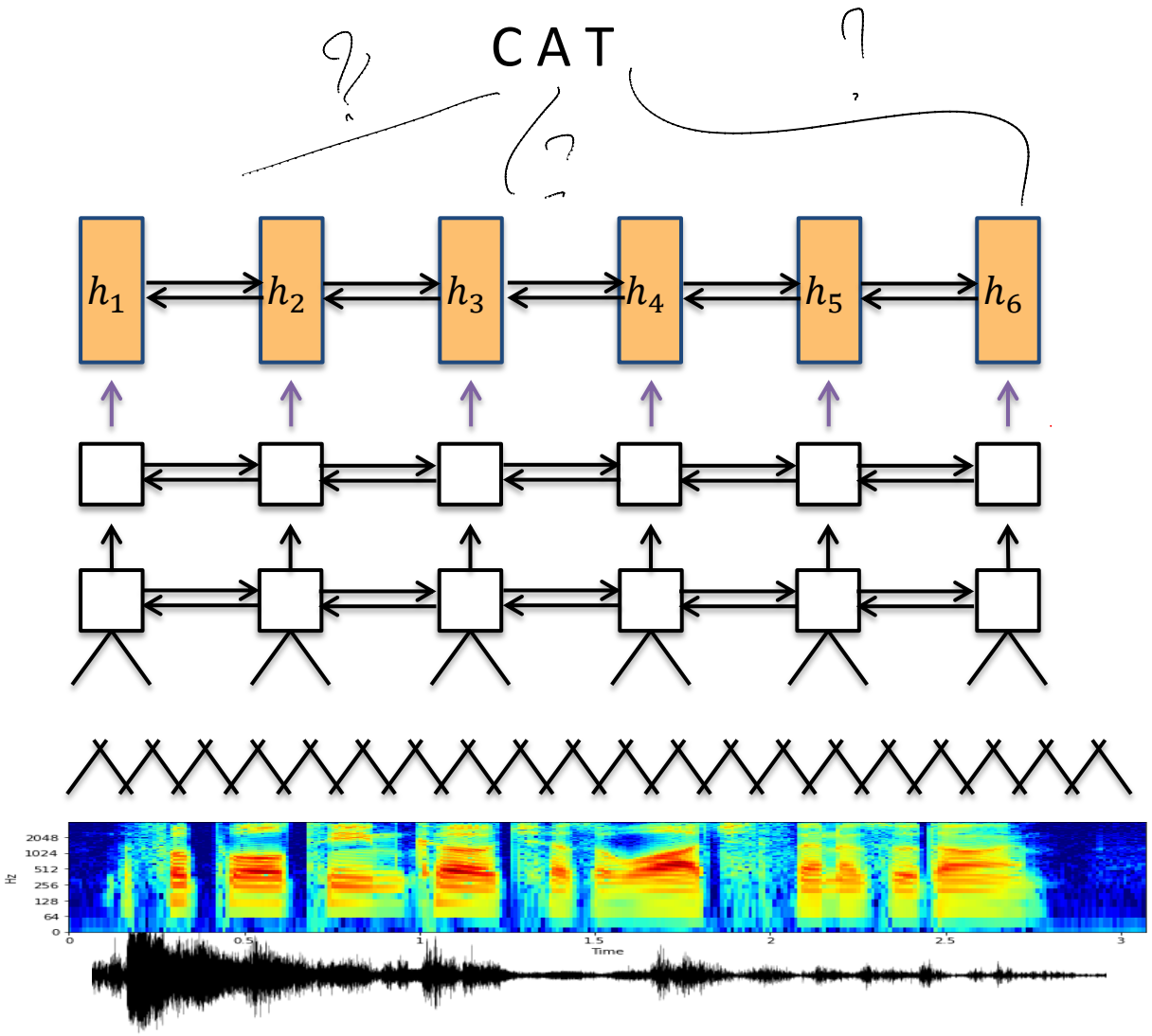
Need alignments between:

- transcript (sequence of words, or phonemes, or graphemes)
- audio (sequence of frames, sampled uniformly in time)

Some possible solutions:

- Attention, treat as sequence-to-sequence task

- Design a loss function that handles the alignment (CTC)



CTC Loss – monotonic sequence alignment

A. Graves, 2006 (https://www.cs.toronto.edu/~graves/icml_2006.pdf)
Let:

Y – transcript (sequence of characters or phonemes)

X – speech features, processed with a neural net

Crucial assumption:

$$|Y| < |X|$$

CTC – intuitions

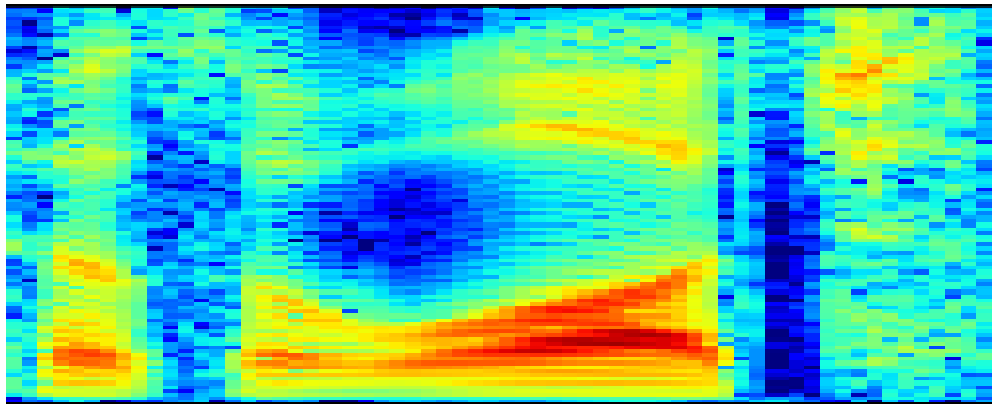
Y: transcript, “a watch”

Y*: **extended transcript**, formed by:

- repeating characters of Y
- allowing a special Blank symbol B.

Y*: `BaaBBwaaattttccchhh`

X:



Note that

1. $|Y| < |Y^*| = |X|$
2. Y can be recovered from Y*:
   1. remove repetitions
   2. remove blanks

CTC – intuitions

Y* maps **many-to-one** to Y

Let $Y = \mathcal{B}(Y^*)$

Define

$$p(Y|X) = \sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)$$

Sum over all ext transcripts / alignments

$$p(Y^*|X) = \prod_{t=1}^{T} p(Y_t^*|X_t)$$

prob of a single ext transcript

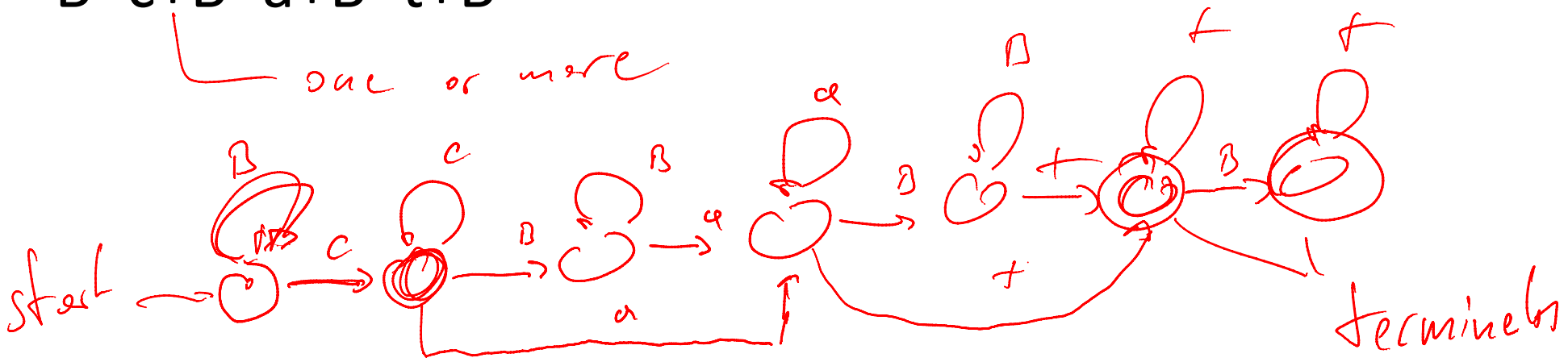
CTC: intuitions

For a given Y, the set of all Y* is regular

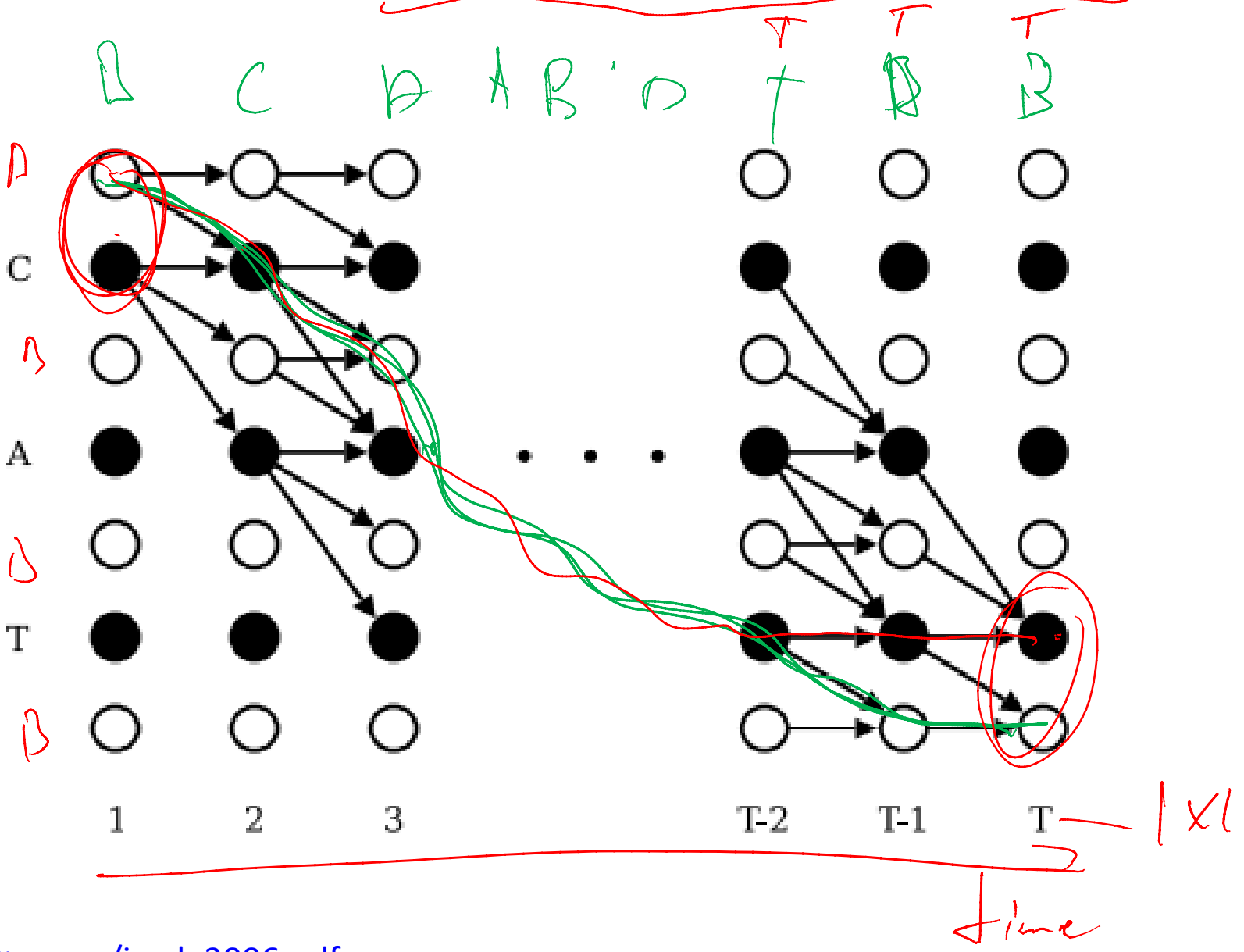
We can build a regular expression (automaton) that accepts all Y*:

E.g. Y = “cat”

Y* = B*c+B*a+B*t+B*

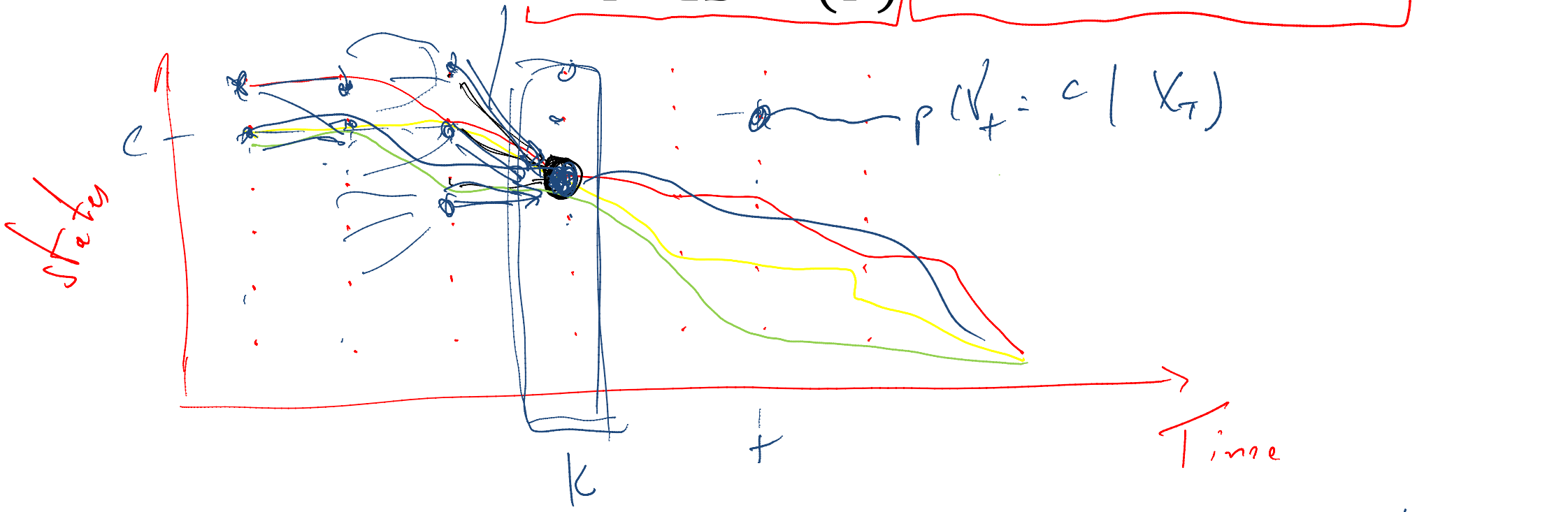


Computing $\sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)$



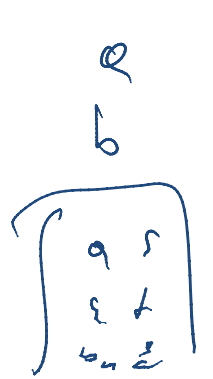
https://www.cs.toronto.edu/~graves/icml_2006.pdf

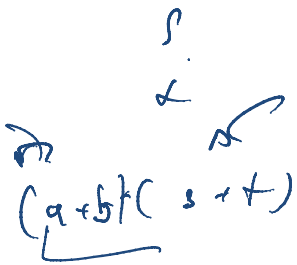
Computing $\sum_{Y^* \in \mathcal{B}^{-1}(Y)} \prod_{t=1}^{T} p(Y_t^* | X_t)$



$\sum$ paths that go through state c at time k

$\left(\sum \text{Prefixes}\right) \cdot \left(\sum \text{Suffixes}\right)$



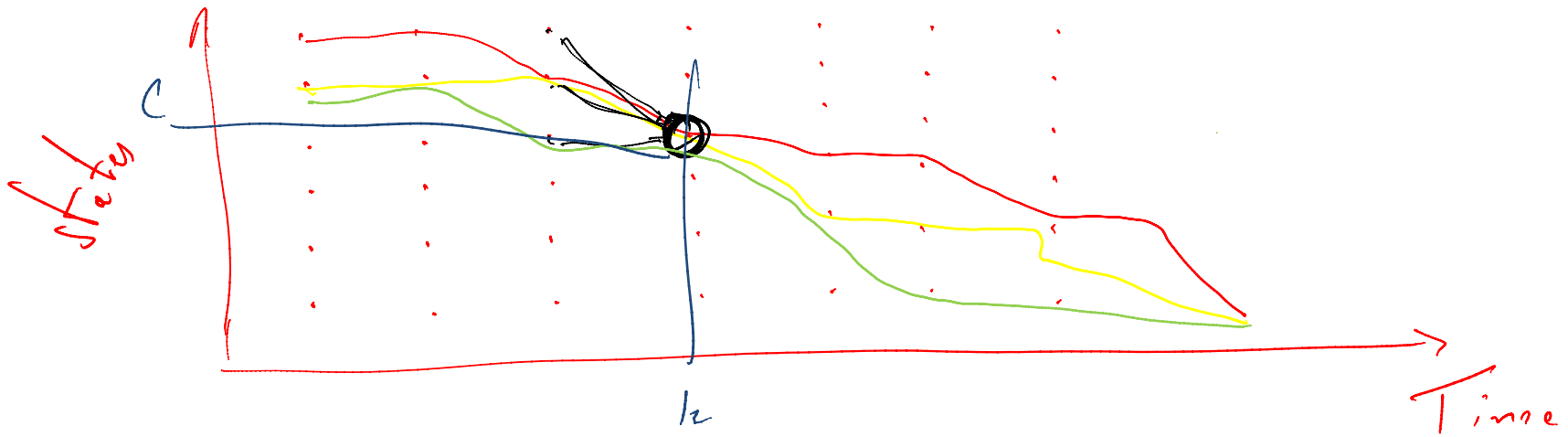


Dynamic programming to the rescue

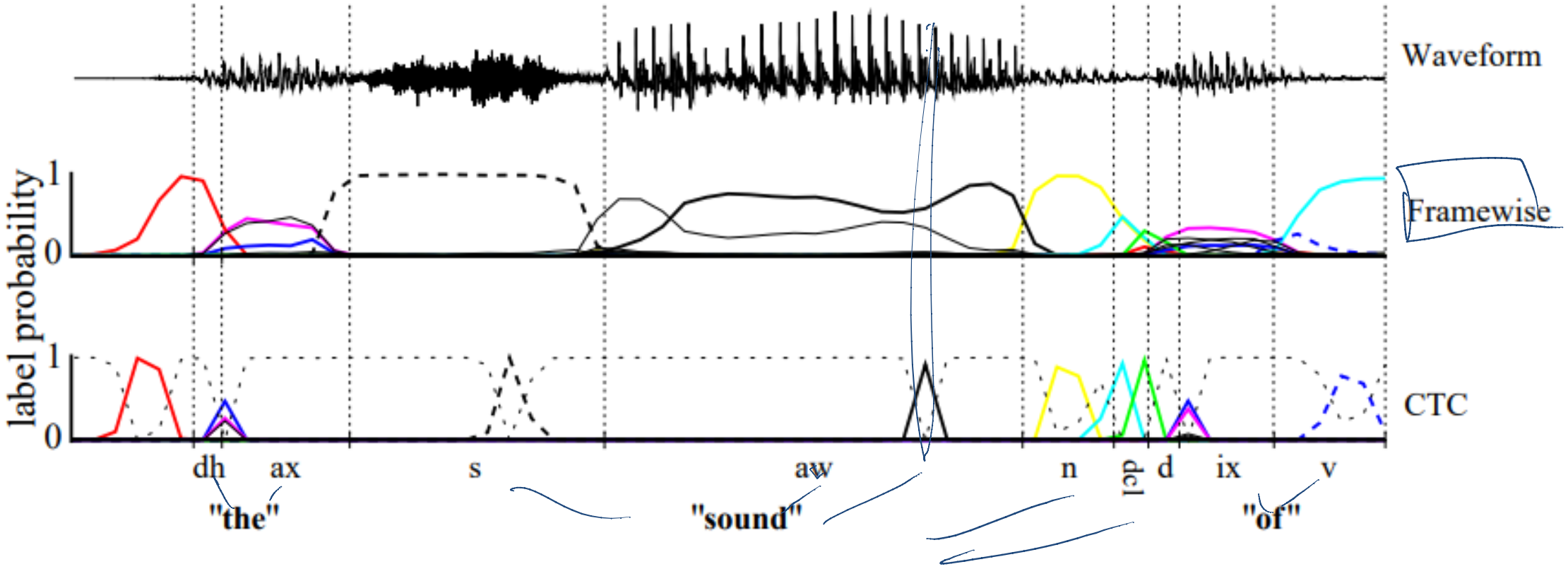
$$\sum_{Y^* \in \mathcal{B}^{-1}(Y)} \prod_{t=1}^{T} p(Y_t^* | X_t) =$$

compute using DP

$$\forall k \sum_{c} \left(\sum_{\substack{Y^*_{1:k} | Y^*_k = c \\ Y^*_{1:k} \in \mathcal{B}^{-1}(Y)_{1:k}}} \prod_{t=1}^{k} p(Y_t^* | X_t) \right) \left(\sum_{\substack{Y^*_{k+1:End} | Y^*_k = c \\ Y^*_{1:k} \in \mathcal{B}^{-1}(Y)_{1:k}}} \prod_{t=k+1}^{T} p(Y_t^* | X_t) \right)$$



CTC in Action



https://www.cs.toronto.edu/~graves/icml_2006.pdf

Generalizing CTC

In CTC:

$$p(Y|X) = \sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)$$

In CTC $\mathcal{B}$ is maps each Y^* to **only one** Y. It requires e.g. that there must be a blank between repeated characters.

It also normalizes probabilities, such that $\sum_{Y^*} p(Y^*|X) = 1$. This ensures $\sum_Y p(Y|X) = 1$

What if there are Y^* that don't map to any Y??
What if $\mathcal{B}$ is many to many??

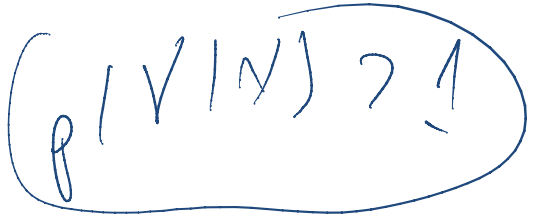
Then: $\sum_Y p(Y|X) \neq 1$

Generalizing CTC

Generalize the relation $\mathcal{B}$:

- Some Y^* don't map to any valid Y (e.g. we only recognize words from a wordlist)

- Some Y^* map to many Y (e.g. don't force a blank between repetitions)



Generalizing CTC

For a general $\mathcal{B}$ simply normalize:

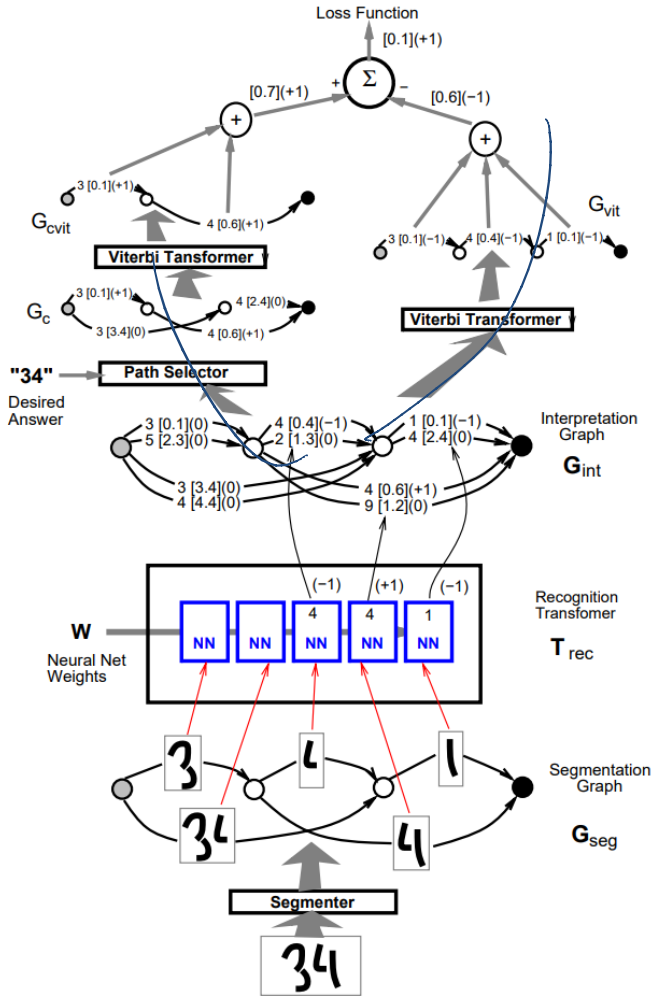
$$p(Y|X) = \frac{\sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)}{\sum_{Y^*|\exists Y \mathcal{B}(Y^*)=Y} p(Y^*|X)}$$

Num is the same as in CTC

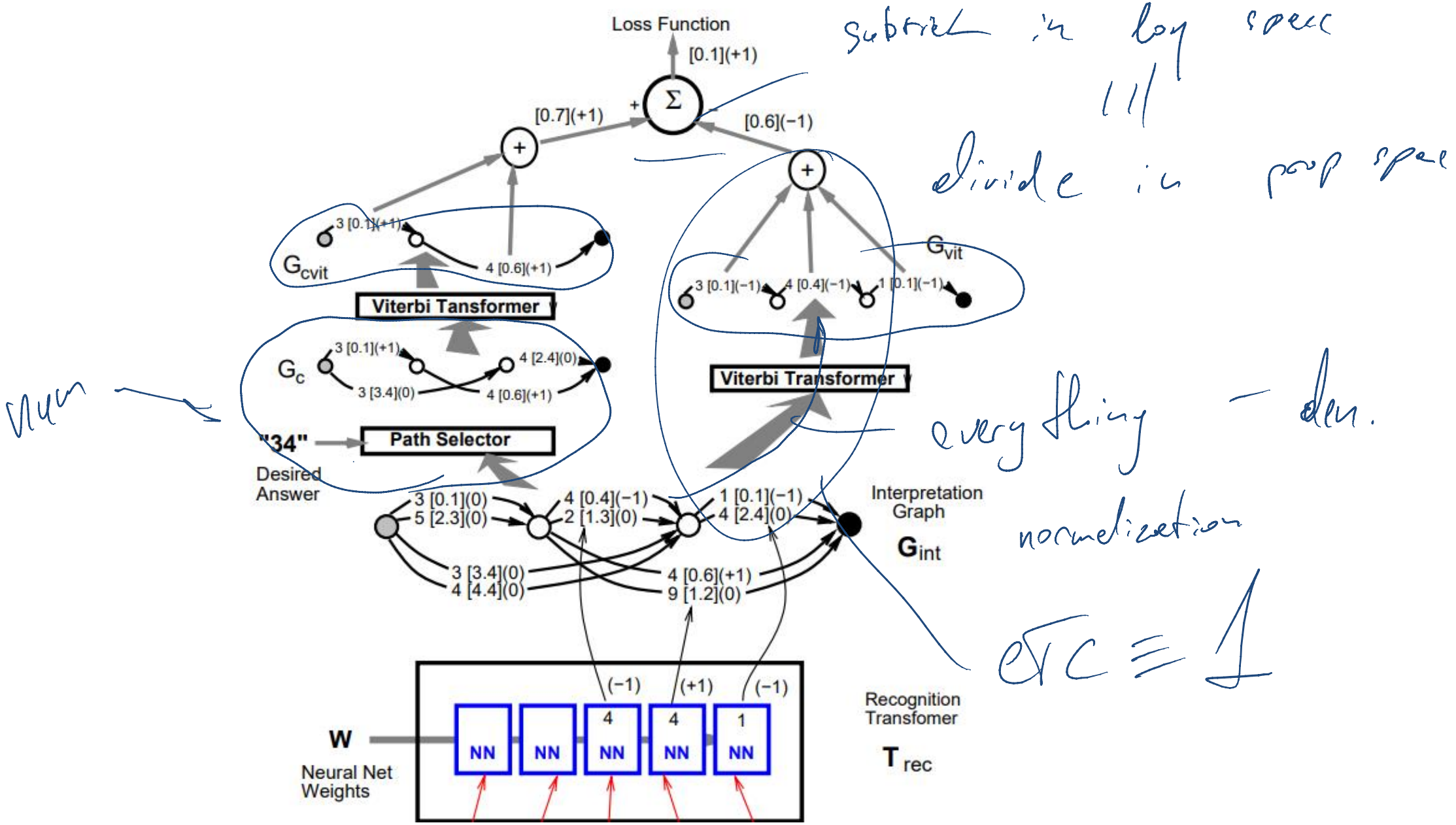
Den is another sum over a set of paths.:

- A similar dynamic program works.
- Caveat: the FST is larger and computation is slower!

Blast from the past:
LeCun's Graph Transformer Nets (1998) are the generalized CTC!



Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.



Y. Lecun, L. Bottou, Y. Bengio and P. Haffner, "Gradient-based learning applied to document recognition," in *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324, Nov. 1998, doi: 10.1109/5.726791.

Summary

CTC is a sequence-level loss that directly computes $p(Y|X)$.

It assumes that:

- $|Y| < |X|$, i.e. targets are shorter
- The alignment between Y and X is *monotonic*

- Many generalized CTC variants, known under different names (GTN, LF-MMI) exist, but due to its special structure, CTC is fast.

BTW, I did some work on CTC in https://arxiv.org/abs/1901.04379