Speech Processing with Neural Networks

Jan Chorowski

Neural Nets can work on any kind of data

- Tabular data:
  - Normalize numerical columns,
  - 1-Hot encode (embed) discrete-value columns
- Images/movies:
  - work with raw pixels using 2D or 3D convolutions
- Language:
  - Treat as sequence of tokens (characters/words/wordpieces)
  - Assign a real-valued vector to each token
  - Deal with different durations using recurrent nets and/or attention
- Speech (this lecture):
  - Treat as a very long time series of samples (single measurements)
  - Transform to a time x frequency representation.

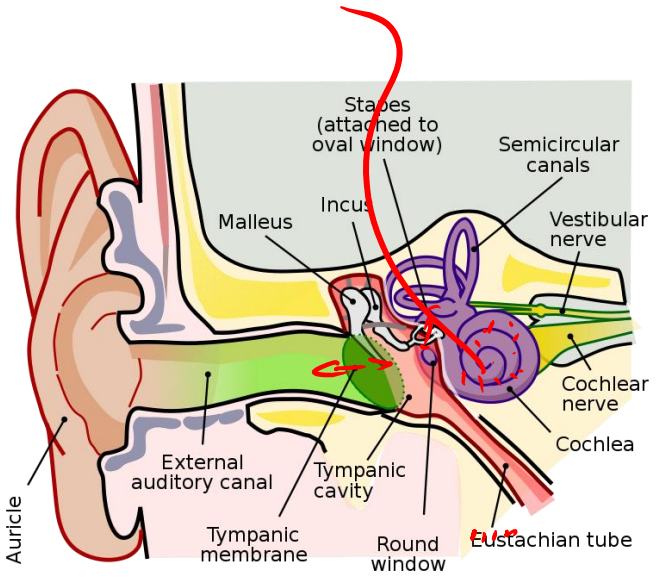
Neural Architecture choice match problems

We often match the neural net to problem domain

- Local vs global connectivity, match data layout
  1D (words, audio), 2D (images), 3D (movies, tomography)
- Weight sharing:
  - All image location use same set of filters
  - All words use same embedding matrix

SPEECH AND SOUND PROCESSING

Sounds



vibration

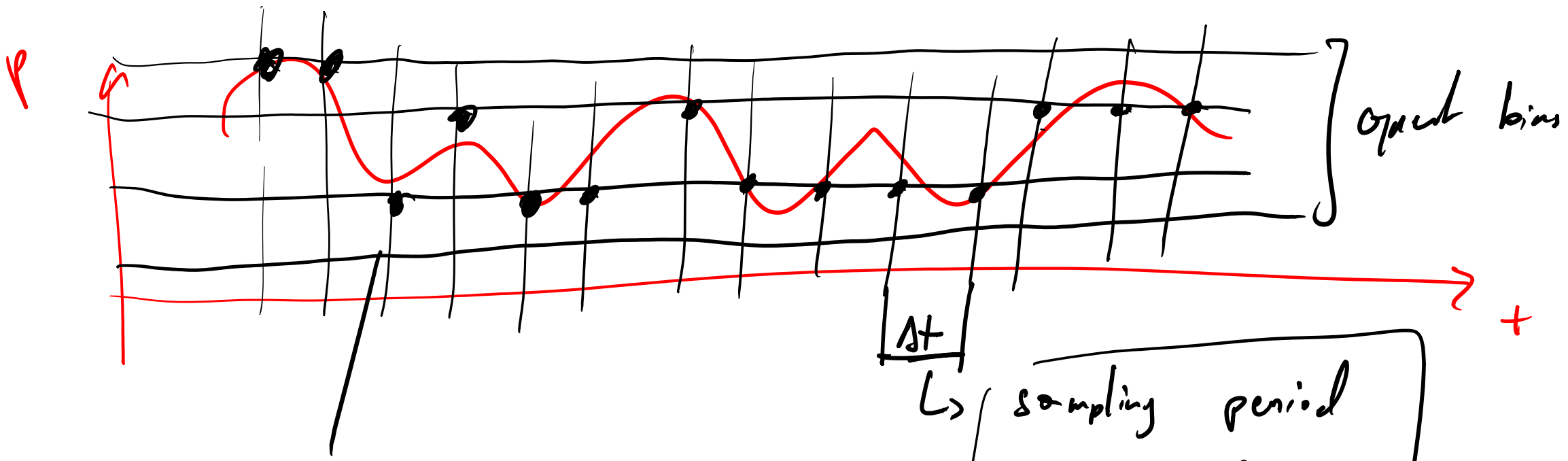
air starts to vibrate

pressure wave

we feel the movement of these tiny hairs

modulating an electrical property

Digital sound: discrete in time and magnitude



$\mathbb{R}^T$

seq of floating point values

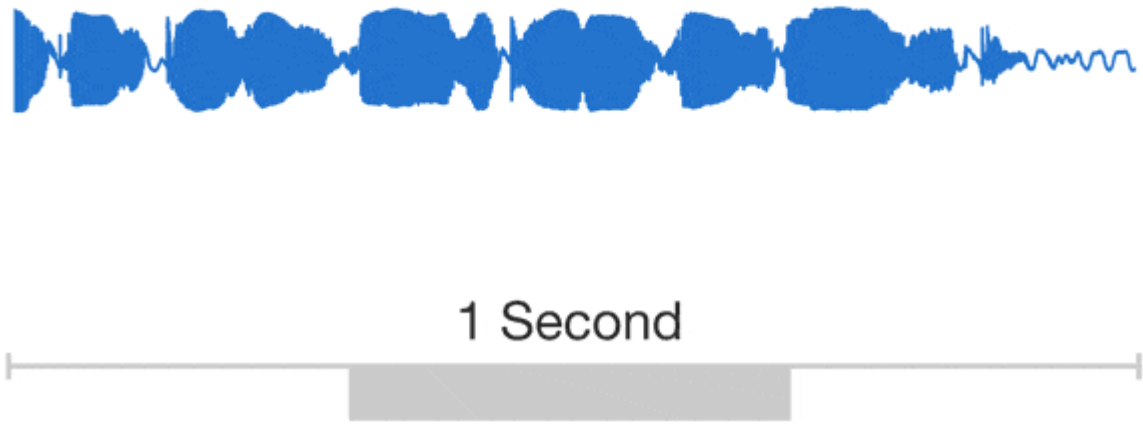
sampling period

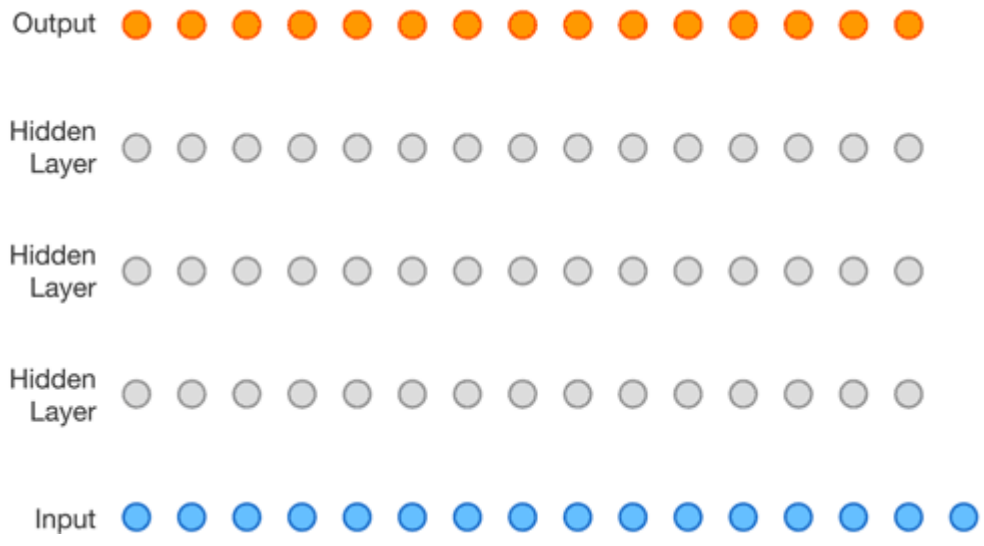
sampling frequency

depend on the signal

44kHz

Nnets for direct sound sample processing
WaveNet: a convnet for raw audio





Conditoned on text

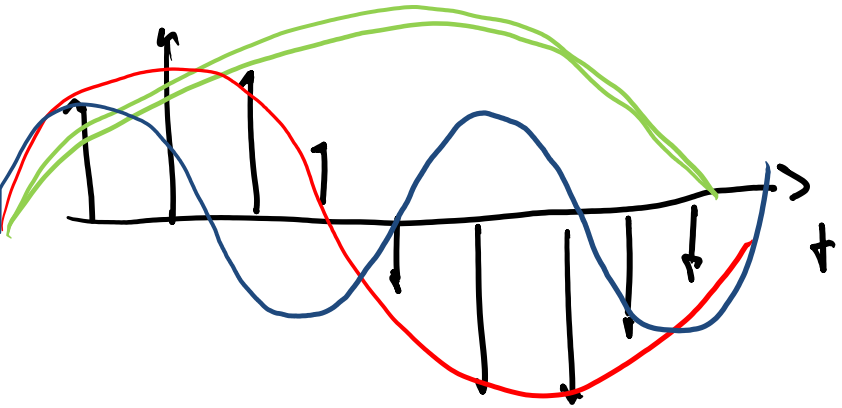
Unconditional

Piano

Wavenet: https://arxiv.org/pdf/1609.03499.pdf https://deepmind.com/blog/wavenet-generative-model-raw-audio/

Fourier Transform: Time <-> Frequency

Sampled signal

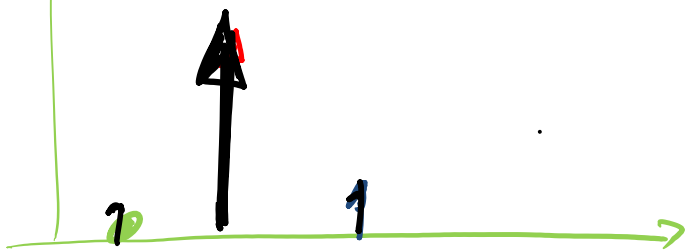


sequence of floats

correlate with sin waves with different frequencies

Fourier Transform

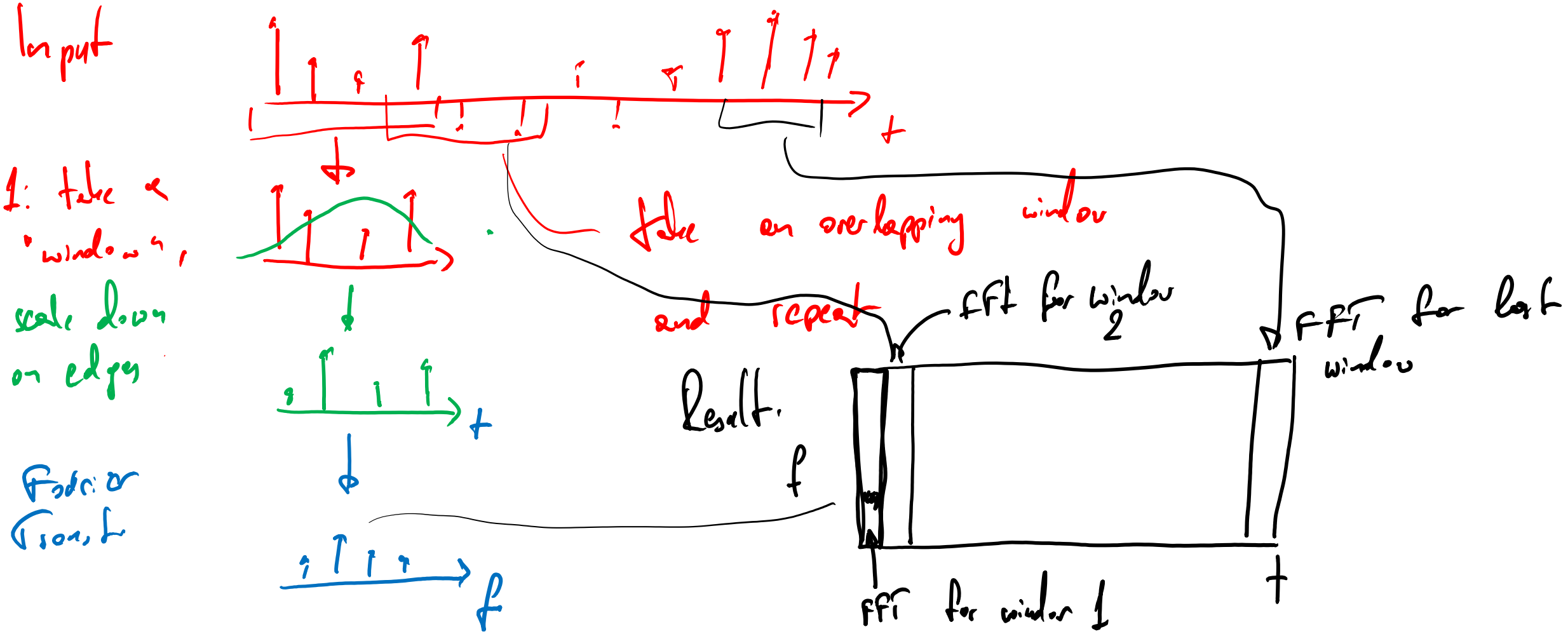
amplitude + phase - complex number



we will only look at amplitudes and drop the phase

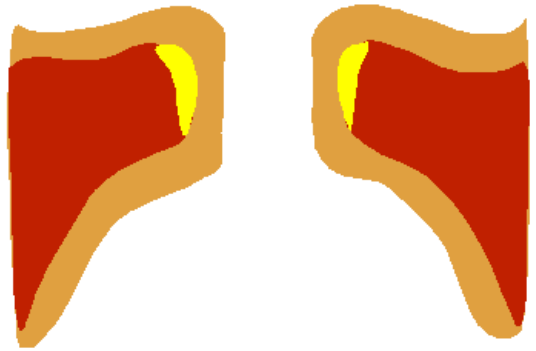
Short Time Fourier Transform STFT

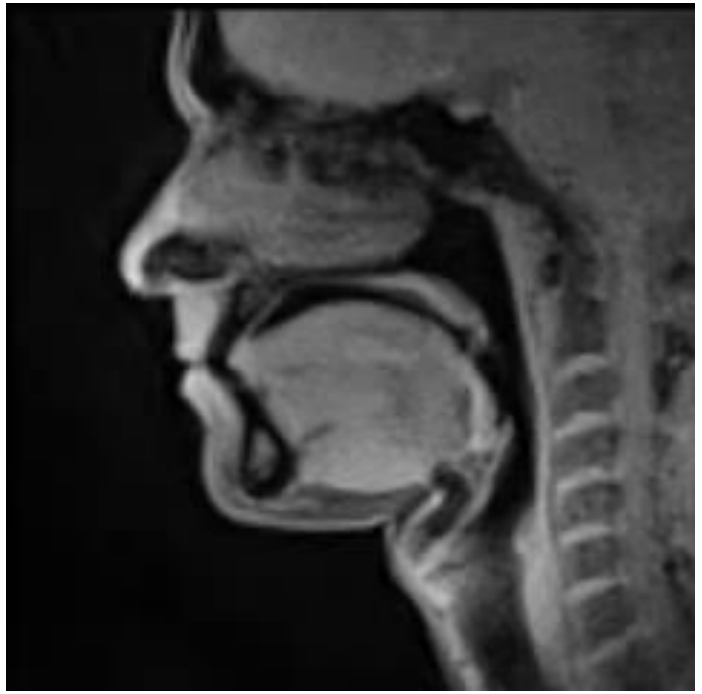
Intuition: time signal <-> time x frequency representation



What is Speech

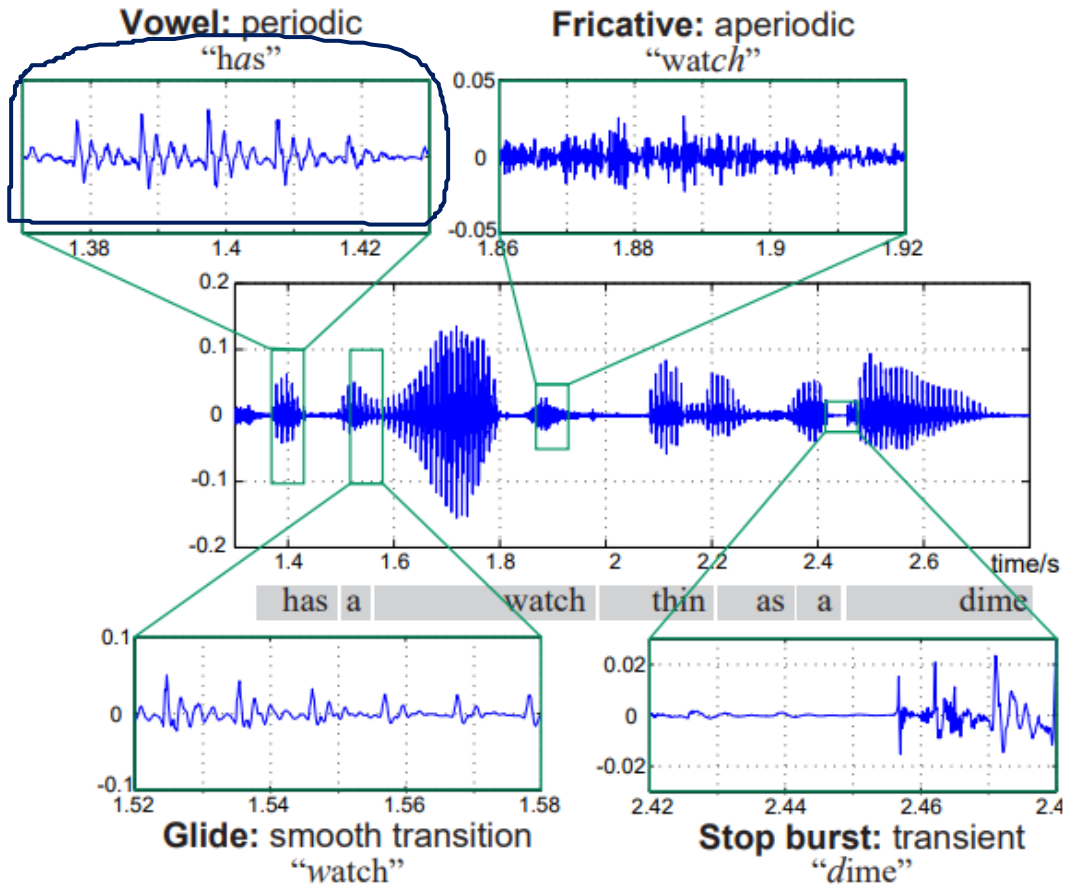
- Vocal cords vibrate (think of saying aaaa)
- The throat, mouth and tongue filter out some frequencies,





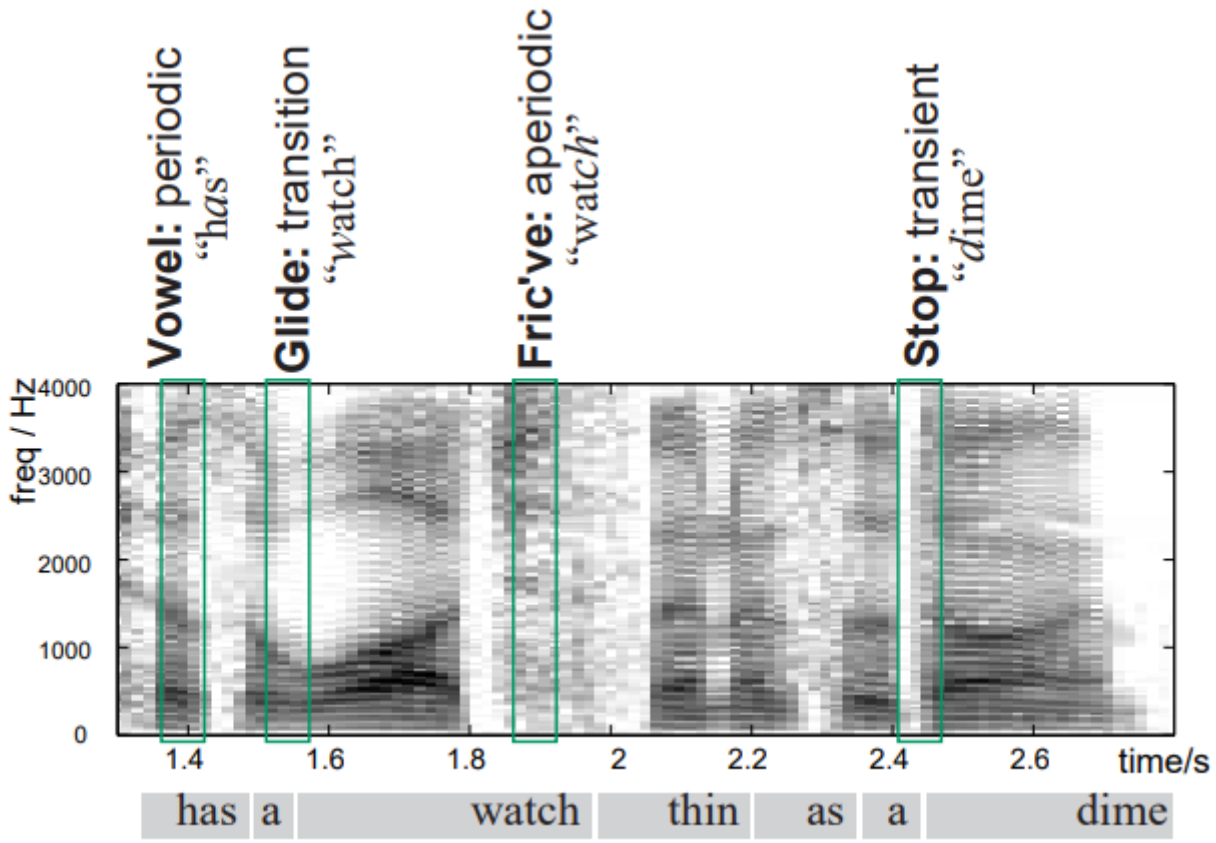
Media source: Wikipedia

Speech visualization: time domain signal



Slide from: http://www.ee.columbia.edu/~dpwe/e6820/lectures/L01-intro+dsp.pdf

Speech visualization: STFT



Slide from: http://www.ee.columbia.edu/~dpwe/e6820/lectures/L01-intro+dsp.pdf

STFT and neural networks

STFT can be implemented using “Neural” operations:

- Windowing and FT can be implemented using a specially crafted convolution
- Complex numbers can be simulated by computing the real and imaginary part separately

The payoff: we can adapt the filters, or backpropagate to the waveform, reconstructing from an STFT.

I did an audio texture synthesizer (similar to neural style transfer) using backprop through STFTs (https://arxiv.org/pdf/1712.08363.pdf).

Signal manipulation using STFT

STFT is readable as a time x frequency image of our signal

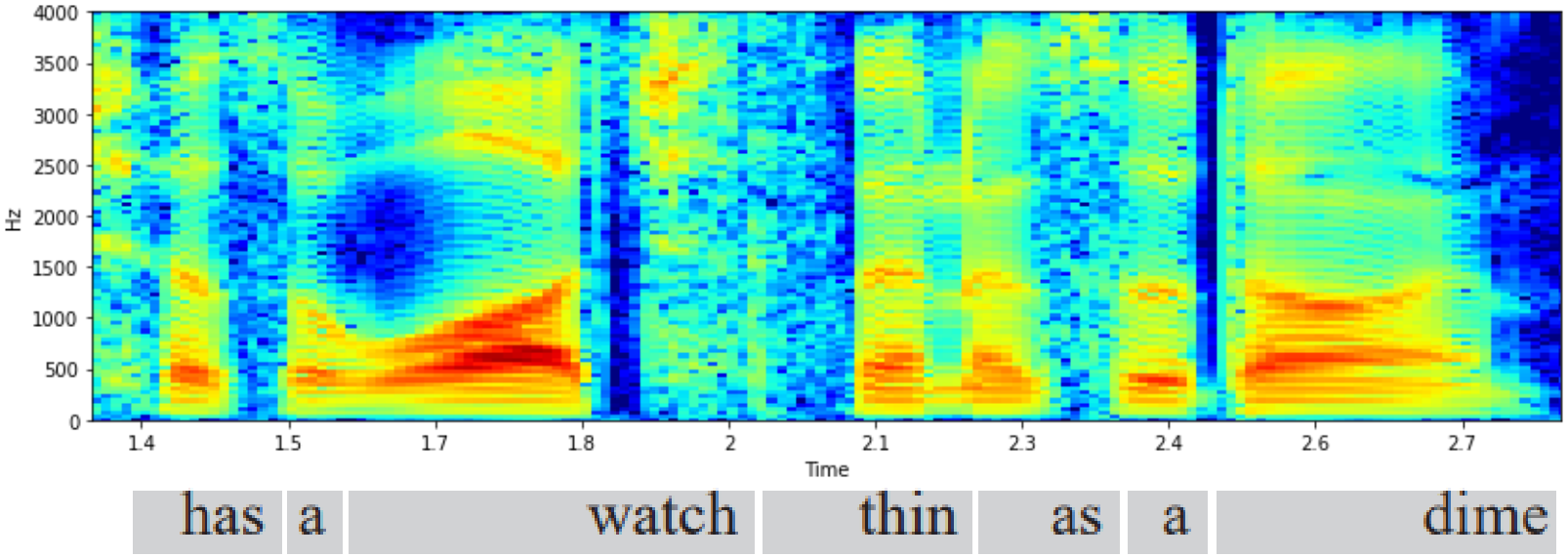
We can reconstruct the audio by searching for a waveform whose STFT matches a desired one.

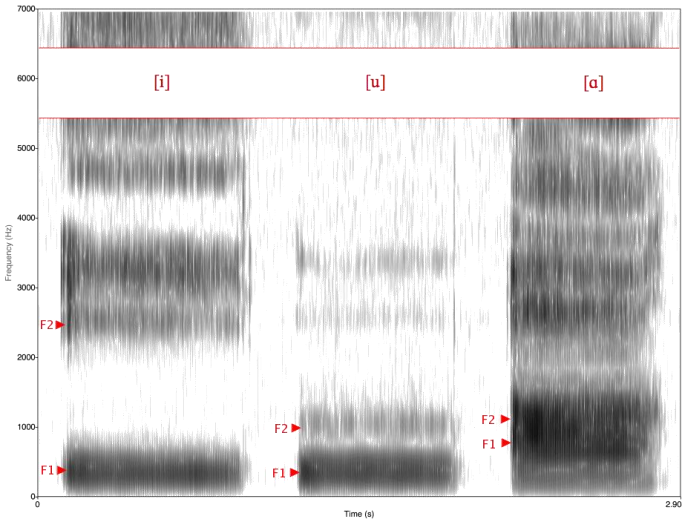
We can use manipulation on STFT to manipulate audio in meaningful ways, e.g. change duration without changing pitch (or change pitch without affecting duration, like in autotune)

FEATURES FOR SPEECH PROCESSING

What ASR features should discard/keep?

- 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

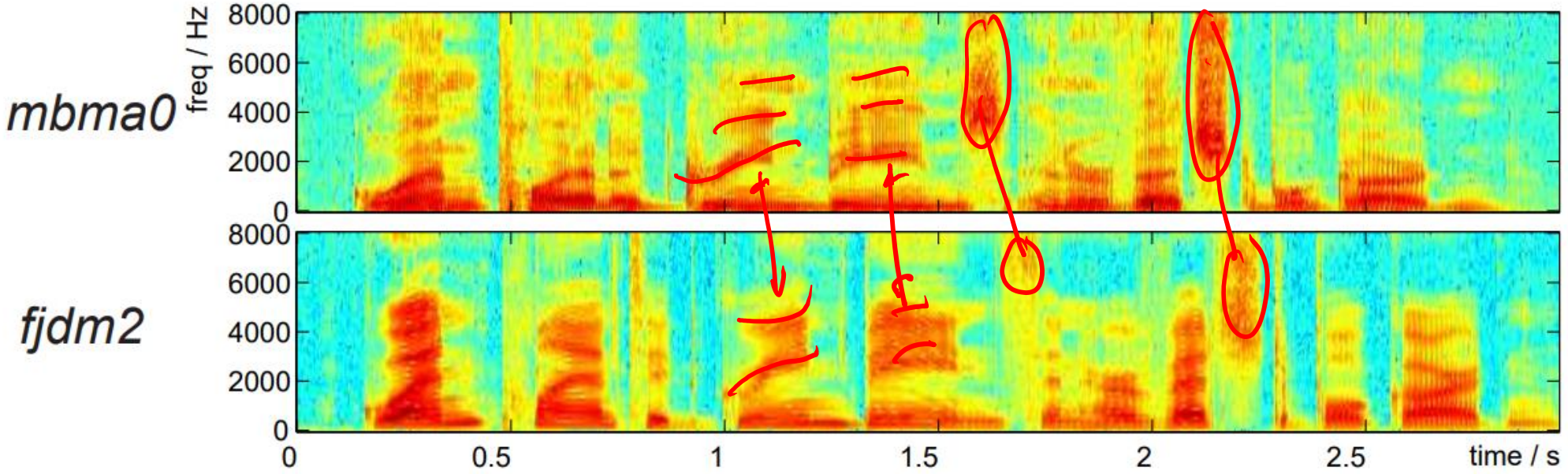


Image from: http://www.ee.columbia.edu/~dpwe/e6820/lectures/L09-asr.pdf

Speech feature intuitions

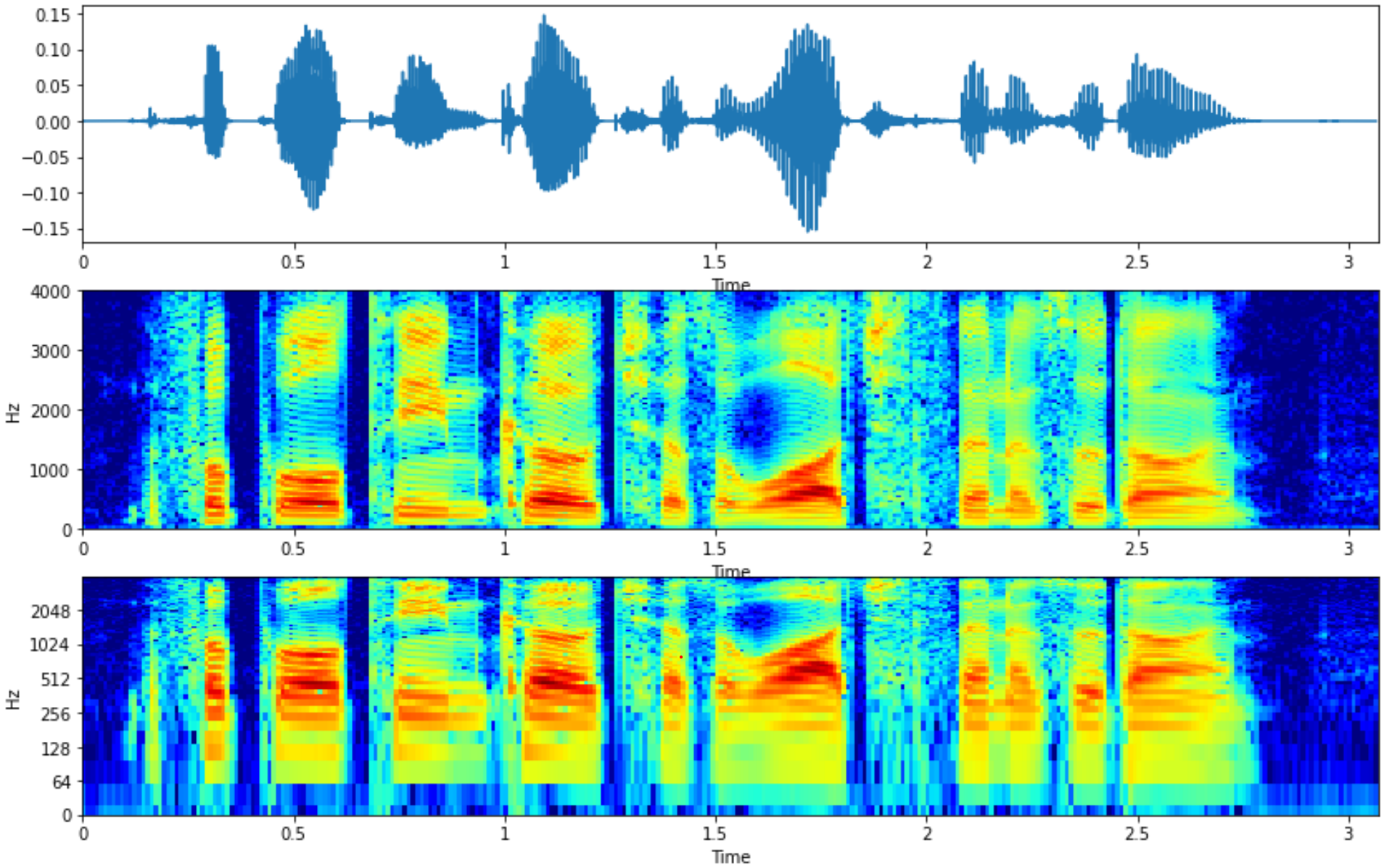
Features for speech recognition should:

- Maximize differences between different phonemes
- Hide differences between individuals (e.g., f0 change for male vs female speakers).

Classical speech recognition systems postprocessed STFTs with:

- Logarithmic compression of the frequency axis (Mel-fiterbank), this is less sensitive to different F0. This is stil useful for neural networks
- Compression and decorrelation of features using a Cosine transform (yielding MFCC, Mel-Frequency Cepstral Coefficients), which could be modeled even using mixtures of Gaussians! This is not so much useful anymoreg

Speech Features for Neural Networks



- Raw

STFT

Log compress Freq axis

MEL scale

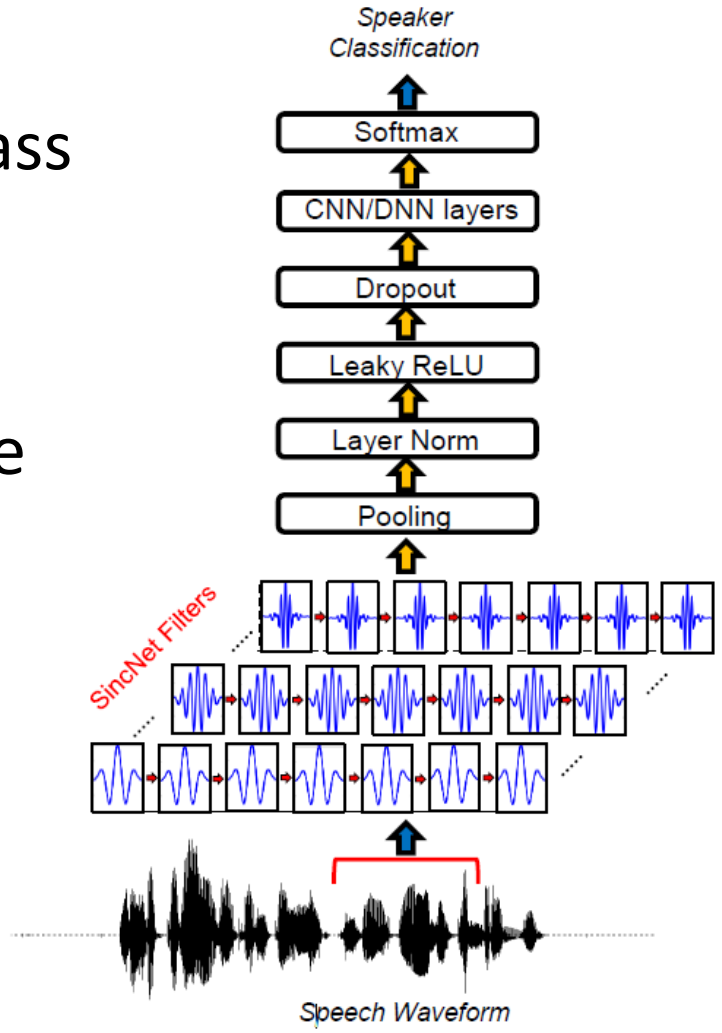
SincNet: Learnable Speech Features

- STFT can be computed using typical neural operations (convolutions, multiplications, additions) with specially hardcoded weights
  - Idea: we could tune allow the handcrafted weights to adapt!
  - But this amounts to learning a raw waveform frontend: slow and prone to overfitting because there are may weights to learn.
- Alternative idea: learn a set of filters with only a few parameters

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)

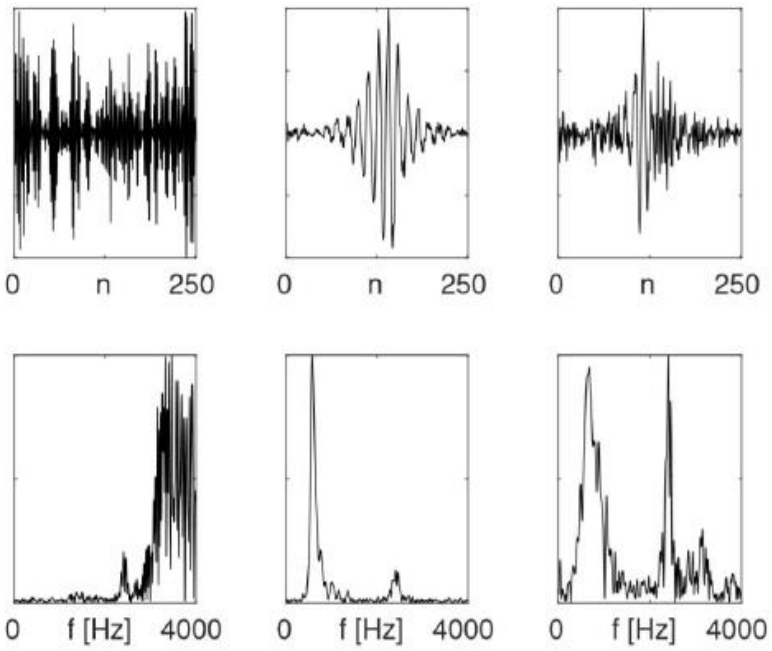
The number of parameters doesn't depend on L.

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

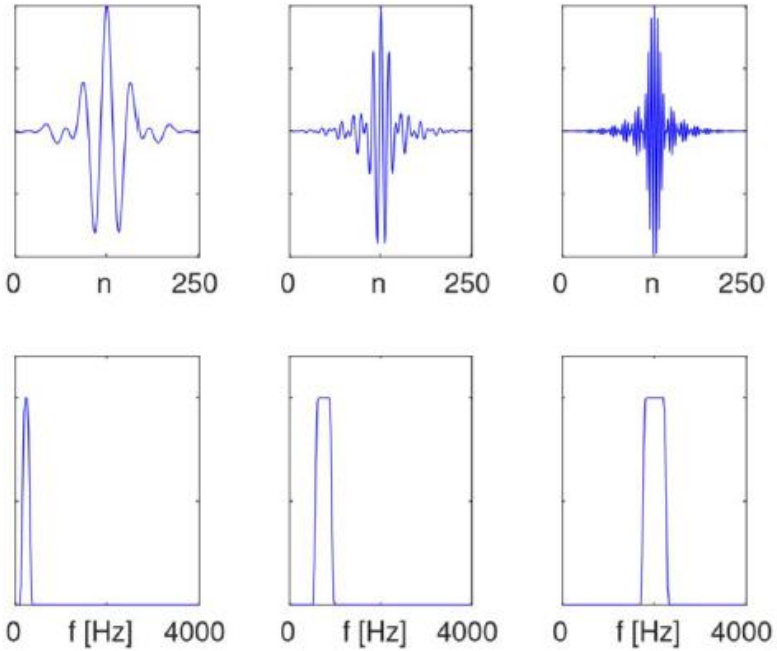
https://www.clsp.jhu.edu/wp-content/uploads/2019/06/Ravanelli_JSALT_Presentation.pdf

Raw waveform ConvNet vs SincNet filters

CNN Filters:



SincNet Filters:



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	Subs	Del	OK	Ins	OK

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

Speech Features for ASR

We typically use STFTs or Mel-Filterbanks.

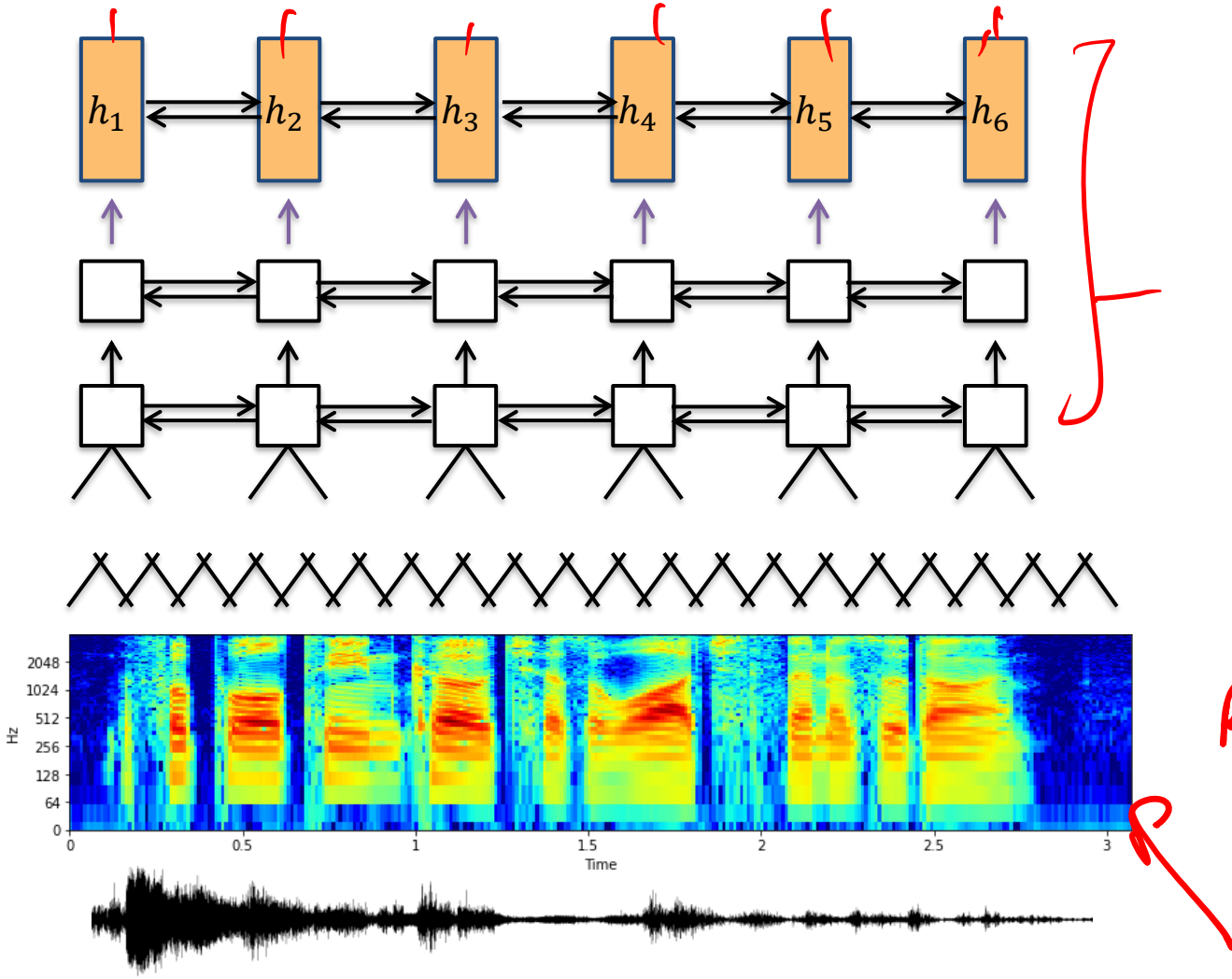
=> An utterance is represented as a $Time \times Frequency$ array of amplitudes.

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. Can allocate per frequency biases.

A typical speech processing architecture



(Bi) LSTMs

2D convs, per-freq bias

Nets: MBL

Speech – Time Alignment

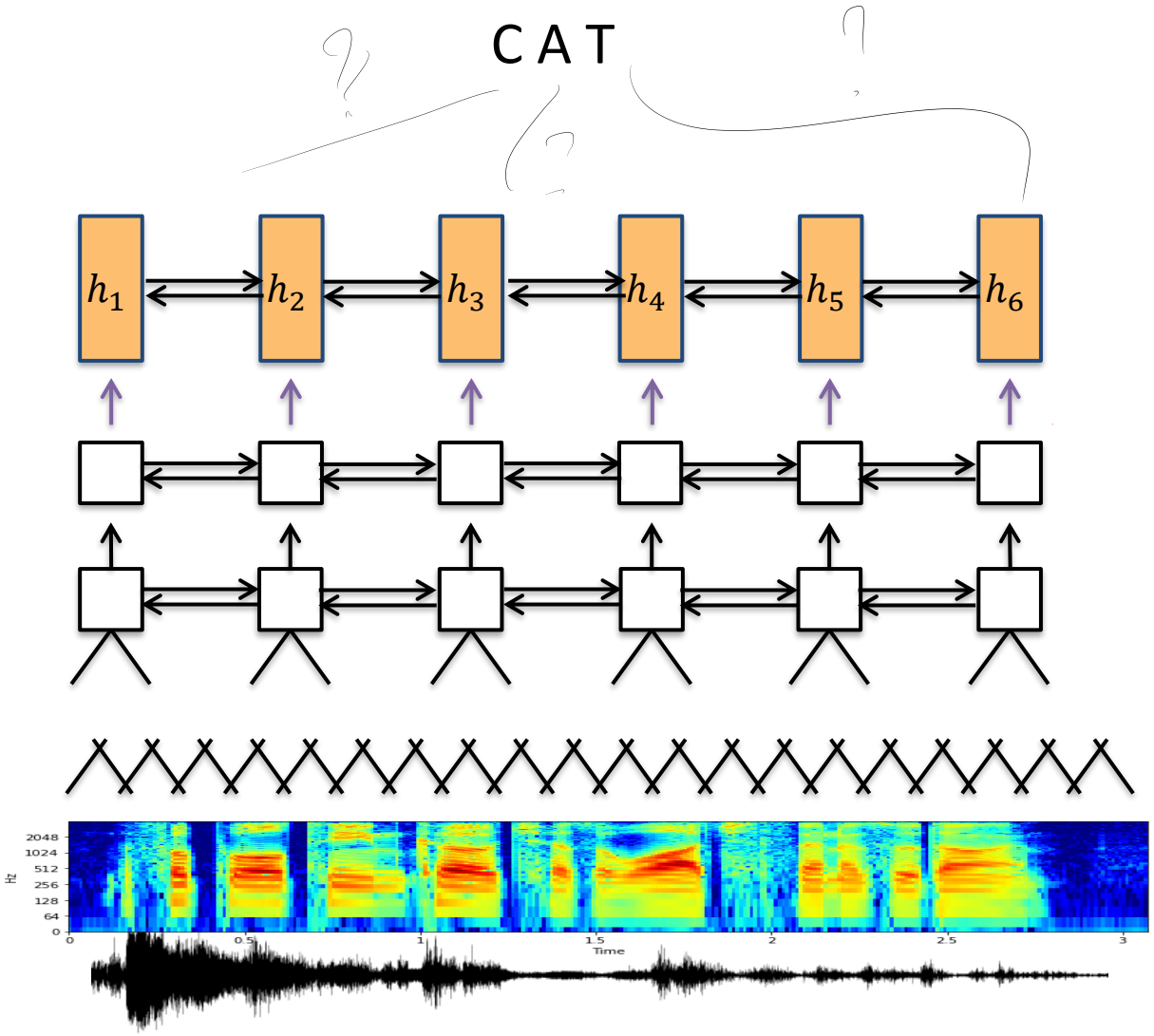
Need alignments between:

- 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)

Goal: find per-utterance alignment, then reinforce it!

Let:

Y – transcript (sequence of characters or phonemes)

X – speech features, processed with a neural net

Crucial assumption:

$$|Y| < |X|$$

CTC – intuitions

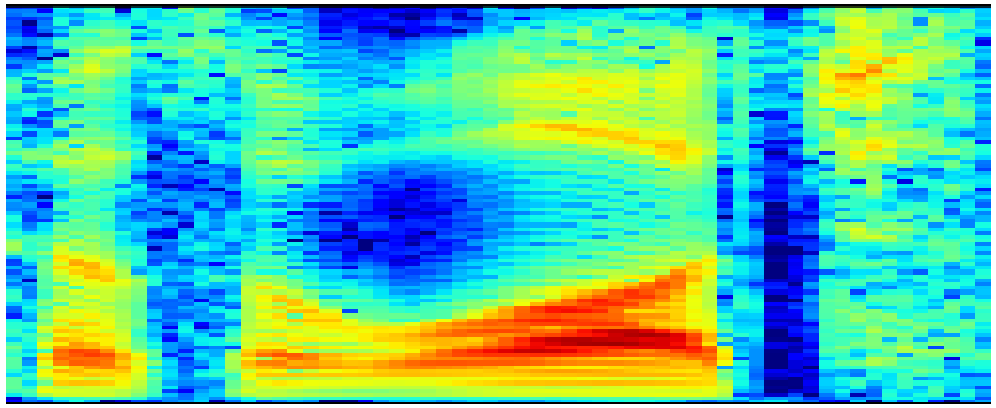
Y: transcript, “a watch”

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 extended sequences

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

: this one is easy, score independently all frame wise predictions

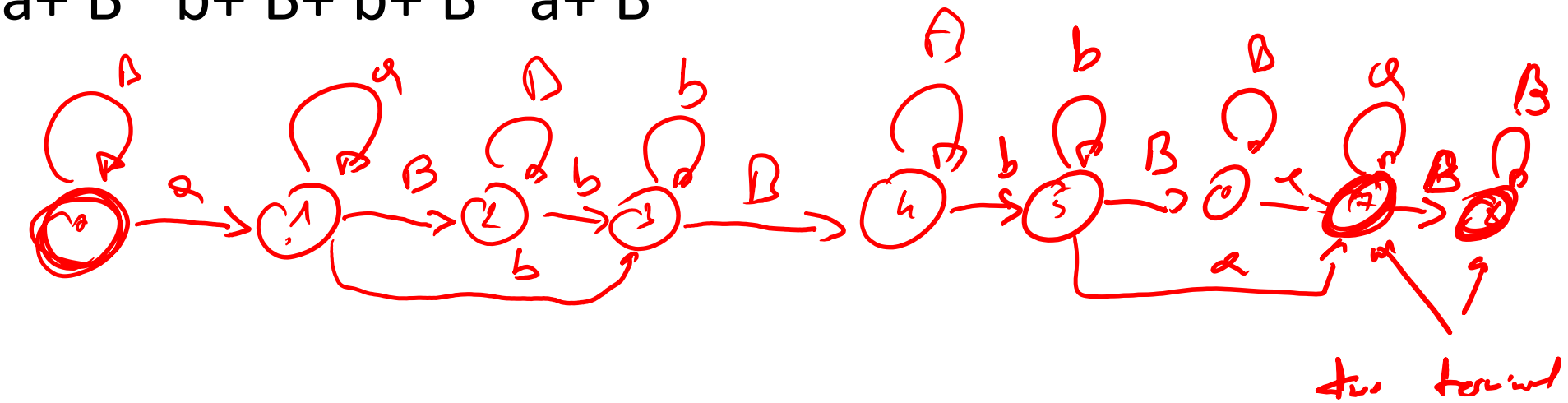
CTC: intuitions

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

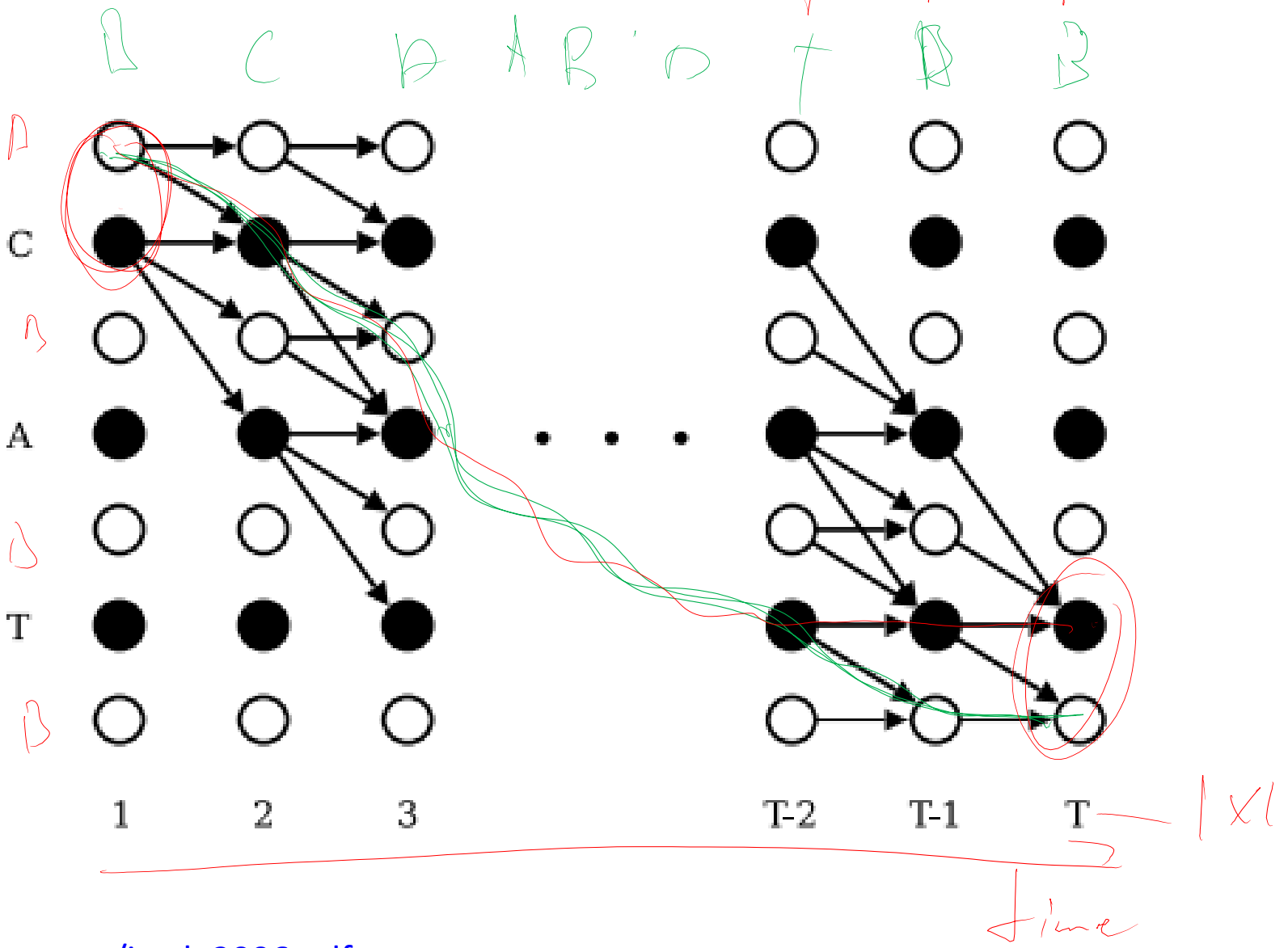
We can build a regular expression (automaton) that accepts all Y*:

E.g. Y = “abba”

Y* = B* a+ B* b+ B+ b+ B* a+ B*

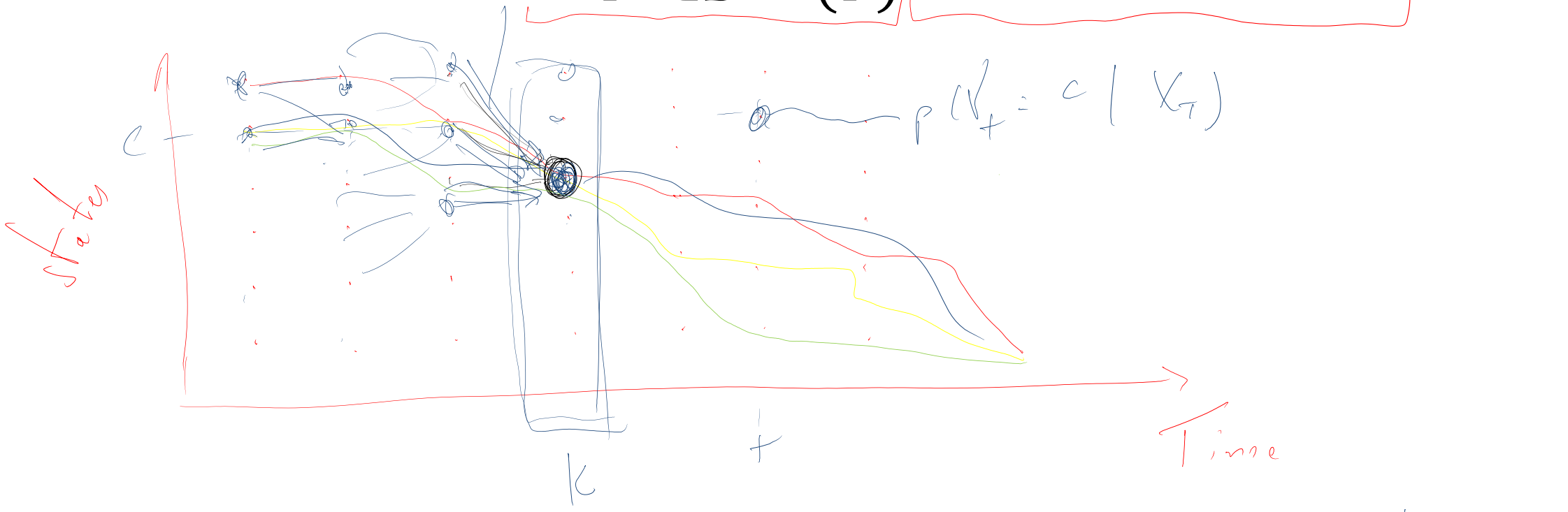


Computing $\sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)$



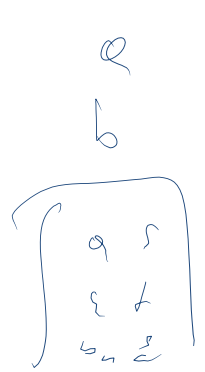
https://www.cs.toronto.edu/~graves/icml_2006.pdf

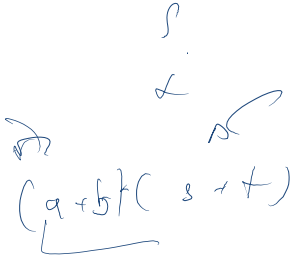
Computing $\sum_{Y^* \in \mathcal{B}^{-1}(Y)} \prod_{t=1}^{T} p(Y_t^* | X_t)$



$\sum_c$ (paths that go through state c at time k)

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

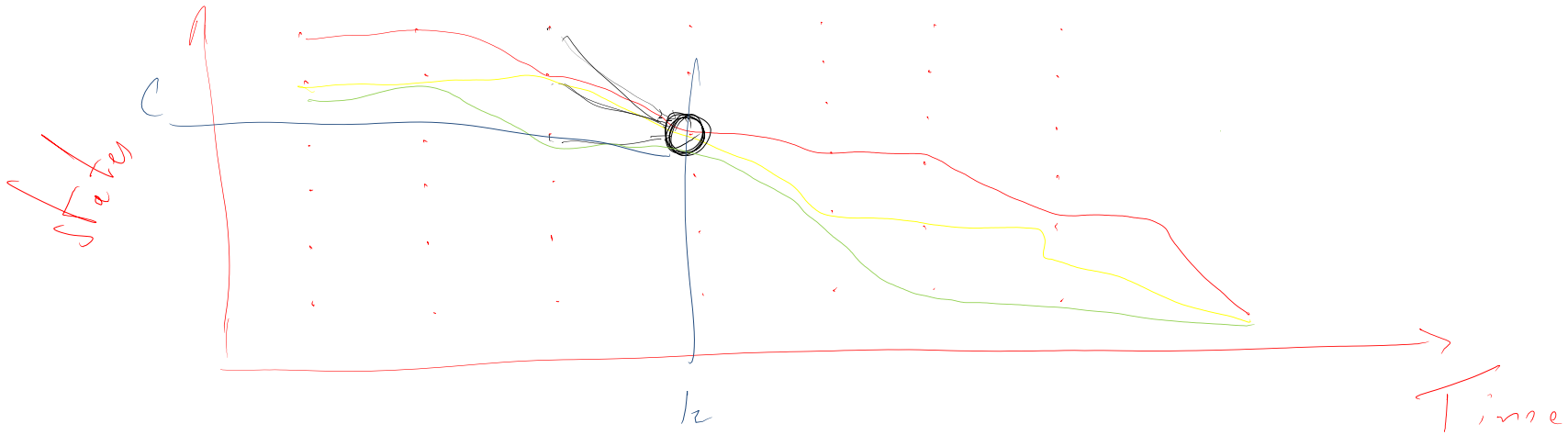




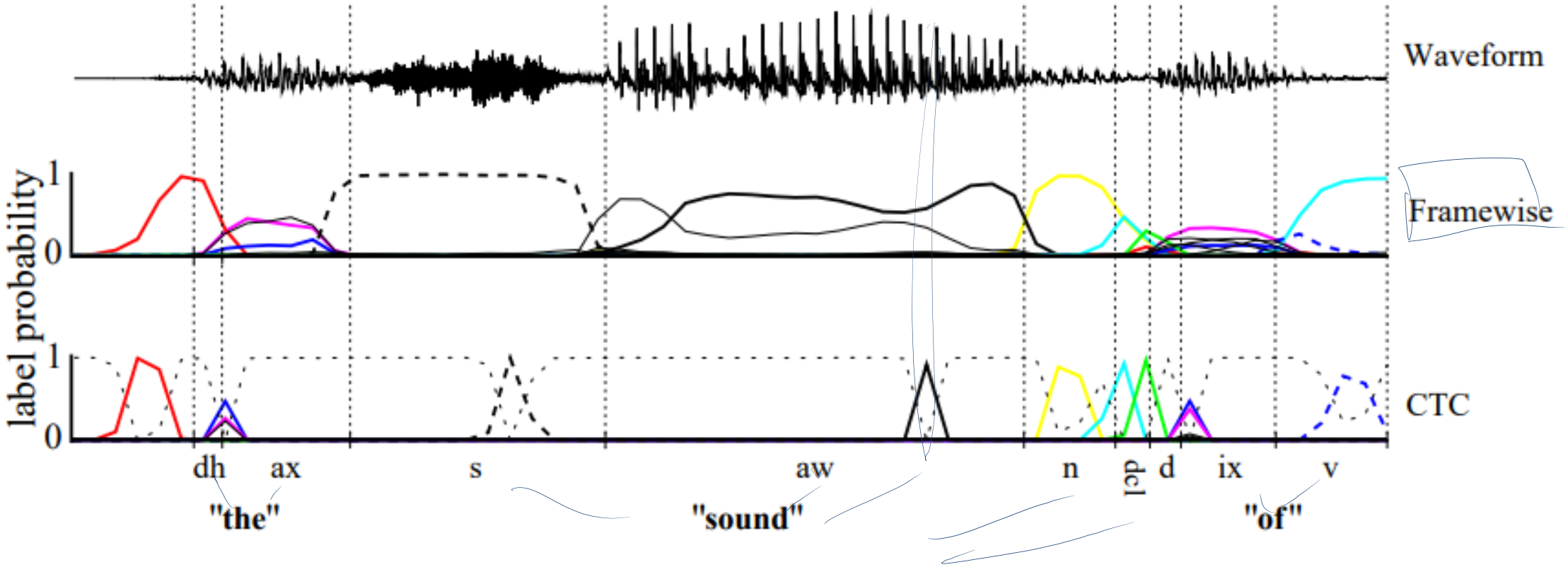
Dynamic programming to the rescue

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

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



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??

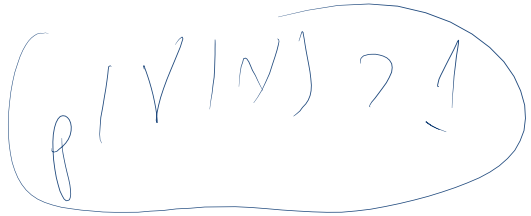
Then: $\sum_Y p(Y|X) \neq 1$

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:

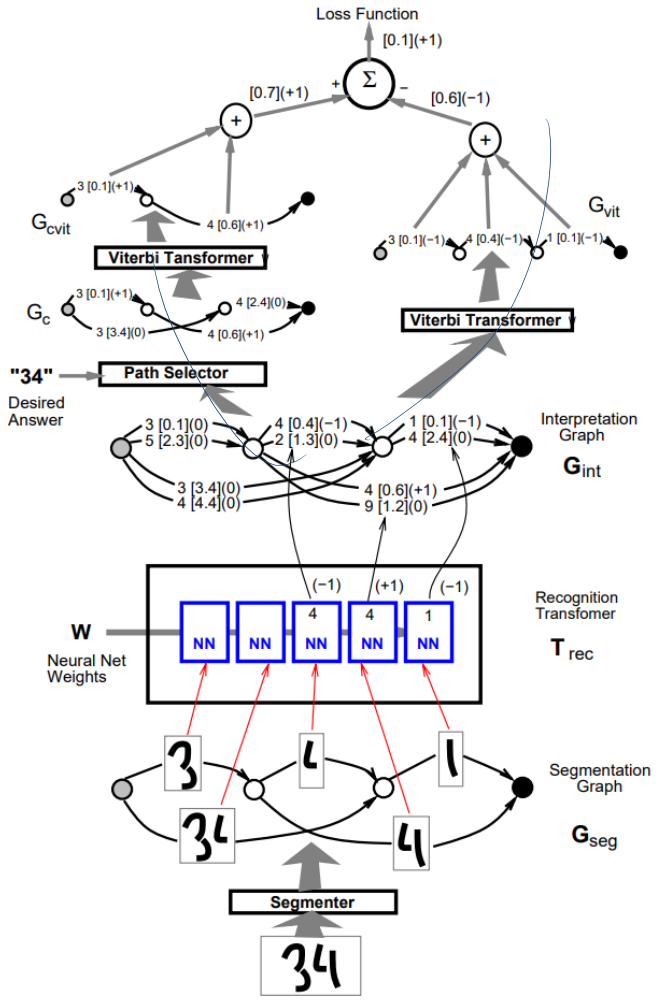
$$p(Y|X) = \frac{\sum_{Y^* \in \mathcal{B}^{-1}(Y)} p(Y^*|X)}{\sum_{Y^*|\exists_Y \mathcal{B}(Y^*)=Y} p(Y^*|X)}$$

Num is the same as in CTC

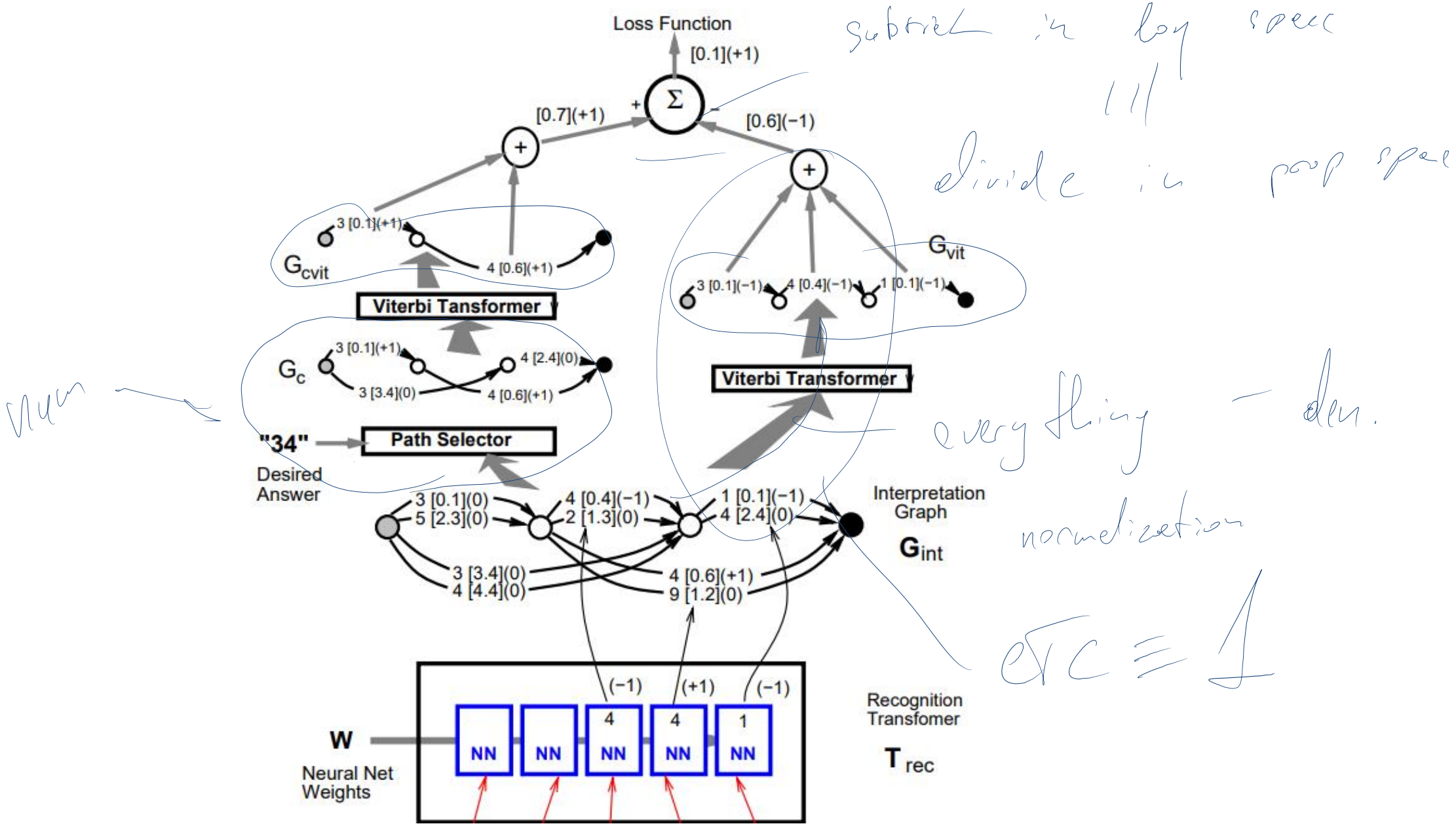
Den is another sum over a set of paths.:

- 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.

Sidenote: structured losses

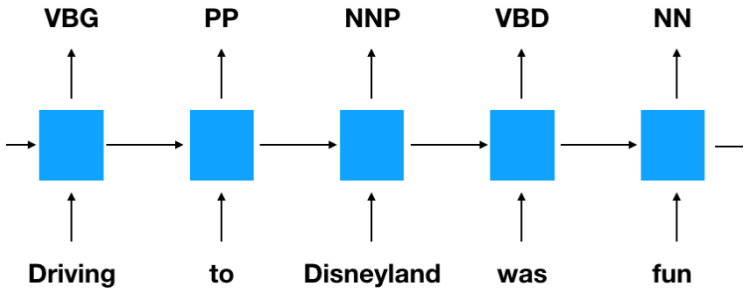
Consider **Image segmentation** and **Part of Speech Tagging:** map inputs (pixels/words) to some classes.

Classes of neighboring inputs interact:

- Two neighboring pixels will have same label
- A pronoun likes to be followed by a verb (e.g. I do)

CRF captures these interactions (think weighted grammar of valid labelings).

We balance fit of the predictions and the CRF grammar. Computation is more complex than generalized CTC, but often doable (e.g. when only pairwise interactions present inference uses max-flow algos).



Summary: CTC

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.

I did some extension of CTC in https://arxiv.org/abs/1901.04379

Summary: Speech and Neural Nets

- Neural nets can work on raw waveforms, but this is costly
  - Special architectures such as SincNet can help
- A time-frequency (STFT) representation is often sufficient, and much faster to process
- NNets are currently the tool of choice for voice processing
  - Speech recognition
  - Speech synthesis (often going through an intermediate STFT target)
  - More niche tasks: voice conversion, speaker classification