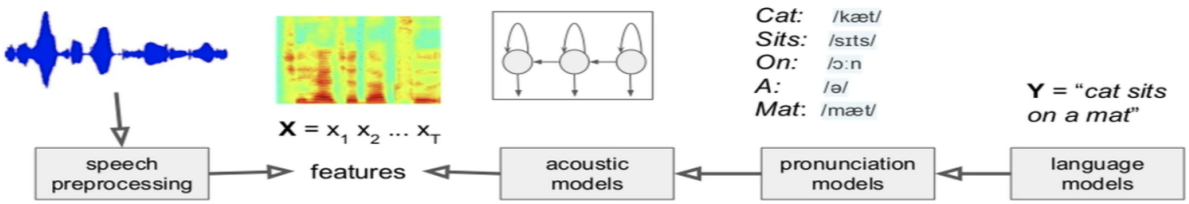
end to end models for speech recognition

This invited lecture wasn't very clear in walking thru the concepts so these notes are hand wavy. The high level ideas of traditional speech recognition, CTC are mentioned and seq2seq is super briefly described.

- The slides for this lecture are very good so I'm going to extensively copy pasta.

Speech Recognition -- the classical way

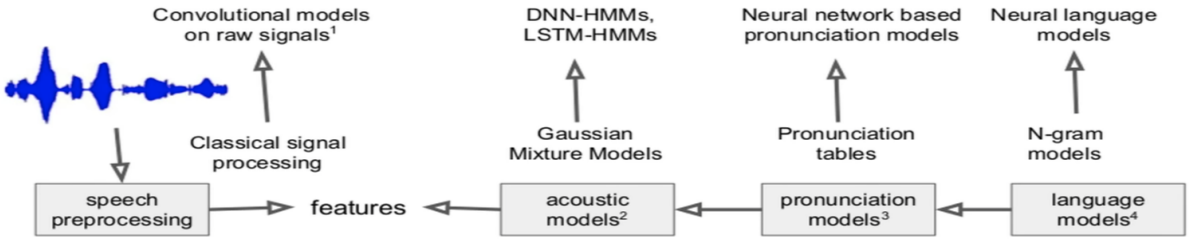
- Inference: Given audio features $\mathbf{X} = x_1 x_2 \ldots x_T$ infer most likely text sequence $\mathbf{Y}^* = y_1 y_2 \ldots y_L$ that caused the audio features



$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} p(\mathbf{X}|\mathbf{Y})\, p(\mathbf{Y})$$

Speech Recognition -- the neural network invasion

- Each of the components seems to be better off with a neural network



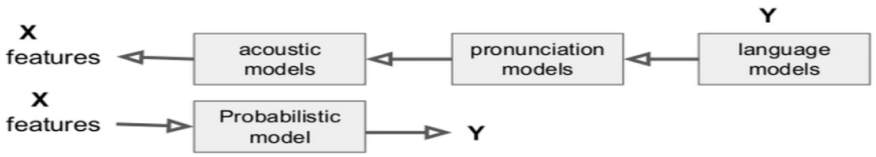
And yet ...

- Each component is trained independently, with a different objective!
- Errors in one component may not behave well with errors in another component
- Instead, lets train models that encompass all of these components together (end-to-end models)
  - Connectionist Temporal Classification (CTC)
  - Sequence to sequence (Listen Attend and Spell)

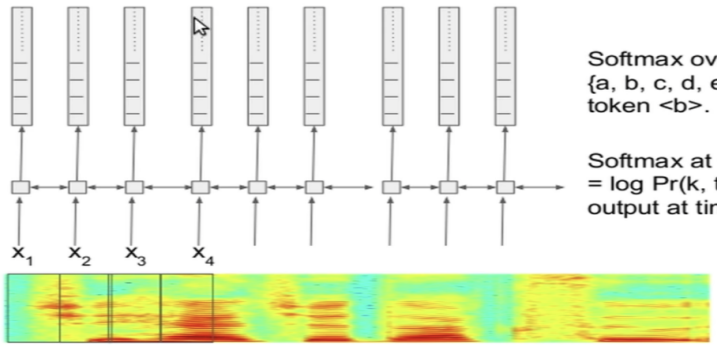
Treat end-to-end speech recognition as a modeling task

- Given audio **X** = $x_1 x_2 \ldots x_T$ and corresponding output text **Y**= $y_1 y_2 \ldots y_L$ where y $\in$ {a, b, c, d, … z, ?, !, …}
- **Y** is just a text sequence (transcript), **X** is the audio / processed spectrogram
- Perform speech recognition, by learning a probabilistic model p(**Y**|**X**)

$$\mathbf{Y}^* = \arg\max_{\mathbf{Y}} p(\mathbf{Y}|\mathbf{X})$$



Connectionist Temporal Classification



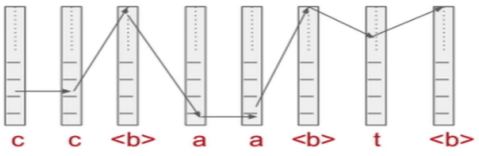
Softmax over vocabulary {a, b, c, d, e, f, .. z, ?, ., !,..} and extra token <b>.

Softmax at step, t, gives a score s(k,t) = log Pr(k, t|**X**) to category k in the output at time t.

- CTC can only transition from a label to itself at the next step, or from a label to empty token ϵ.

CTC - How frame predictions map to output sequences

- Repeated tokens are deduplicated
  - cc <b> aa <b> t <b>
- Any original transcript, maps to all possible paths in the duplicated space:
  - cc<b>aa<b>t<b> maps to cat
  - cc<b><b>a<b>t<b> maps to cat
  - ccccc<b>aaaaa<b>ttttt<b> maps to cat
  - ccccc<b>aaaaa<b>ttttt<b> maps to cat
- The score (log probability) of any path is the sum of the scores of individual categories at the different time steps
- The probability of any transcript is the sum of probabilities of all paths that correspond to that transcript



*Because of dynamic programming, it is possible to compute both the log probability p(**Y**|**X**) and its gradient exactly! This gradient can be propagated to neural network whose parameters can then be adjusted by your favorite optimizer!*

- the audio waveform is produced by taking the raw audio and breaking them up into frequencies using FFT → he doesn't go into much detail on this.
- the problem is the above is error prone since it does not take advantage of a language model. if you can take advantage of a language model, this improves performance quite a bit → most production systems do this
- he does not go into exactly how to use a language model here 😞
- You could also try CTC with word targets instead of character targets.
- instead we could use a seq2seq model. the problem with a naive seq2seq model that we saw in machine translation is that these audio sequences are broken up into 1000s of frames even for a short audio

signal, so the problem of long term dependencies is a lot higher. thus we will need to use attention.

- Word error rate metric for speech recognition is the fraction of words that are correct.