Neural nets for speech signal processing

Ling 574 guest lecture

Matthew C. Kelley, Dept. of Linguistics, University of Washington

Learning outcomes

- Describe how speech data differs from textual data
- Describe the steps needed to convert speech data to a format neural nets can use, including some advantages and disadvantages
  - MFCCs/log mel spectrograms
  - Raw speech data
  - wav2vec
- Identify loss functions that are commonly used for speech recognition
- Describe how a neural network’s output is decoded and scored to yield the final sequence of recognized words in speech recognition

Some speech-related tasks

- There are many speech-related tasks performed with neural nets
- (Automatic) speech recognition (ASR): produce textual data from acoustic speech data
  - We will focus on this task today
- Speech synthesis (text-to-speech; TTS): produce acoustic speech data from text
- Speaker diarization: tag which speaker is speaking in a region of speech
- Forced alignment: automatically place boundaries between speech segments
  - This is my specialty
- Keyword spotting: detect the presence of certain important words in a recording
- Automated acoustic measurements: measure properties like formant values or pitch without needing to set speaker-specific parameters
- Wake word detection: detect words that signal the beginning of a user action (like “Alexa” or “Hey, Google”)

Basic speech concepts

- In phonetics, we talk about speech as an acoustic signal
- Signal has different frequency components that make it up
- Those frequency components are related to speech sounds and words
  - Though this relationship is *remarkably* complex
- Our general goal in ASR is to take this speech signal and get words out of it

Speech recognition pipeline

Acoustic representation

- How is speech represented?

Acoustic model

- Maps speech to recognition units

Decoding

- Convert model output to symbols

Language model

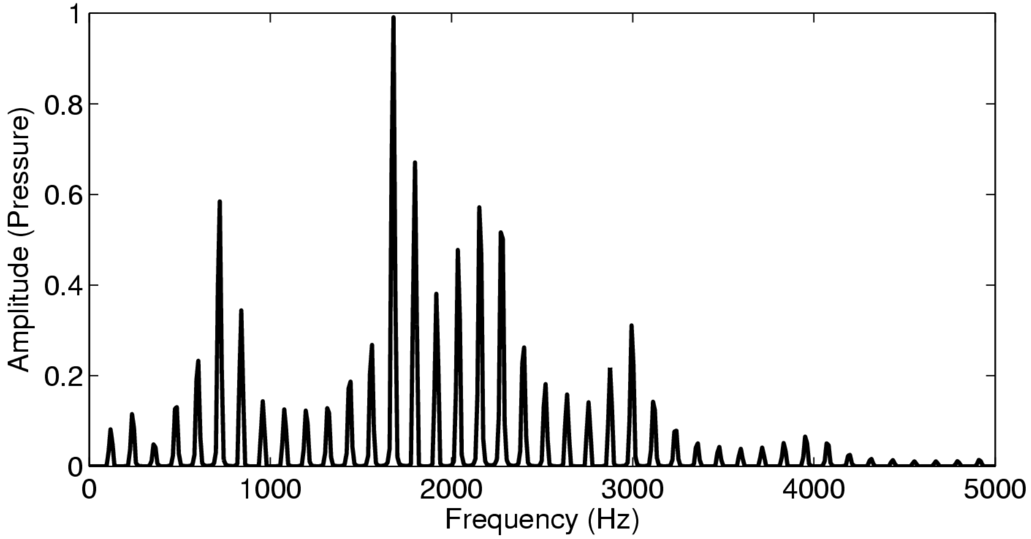
- Use probability to score possible outputs

Acoustic representation

CHOOSING A FORMAT FOR SPEECH

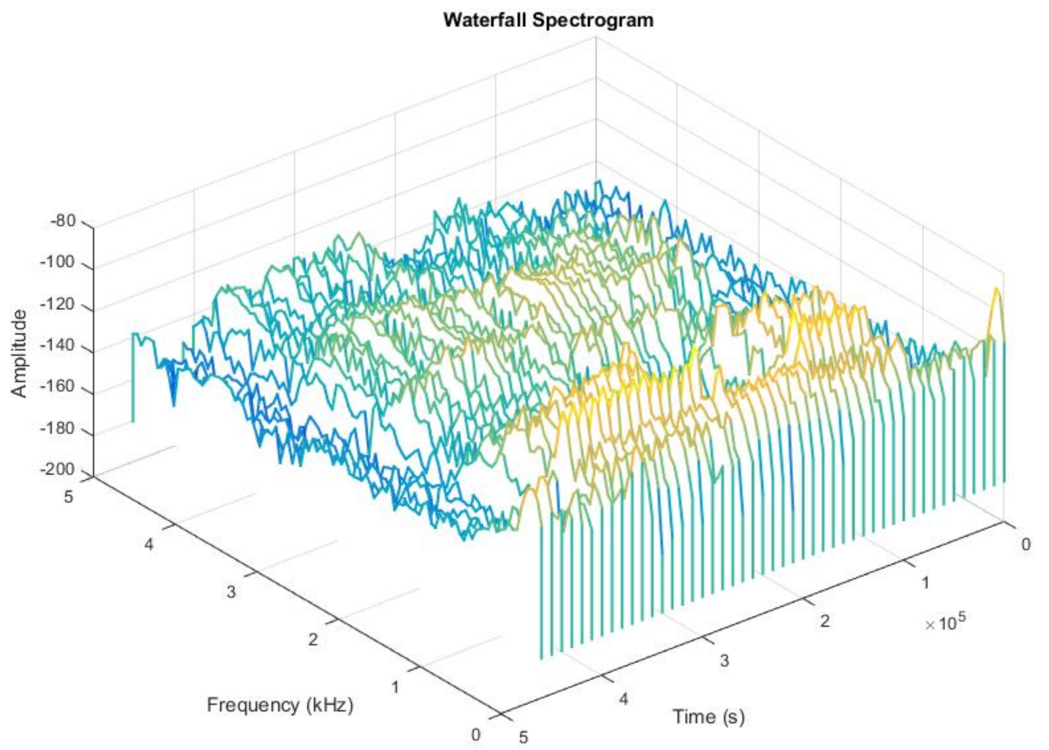
Time and frequency

- Recordings are stored as series of amplitude samples over time
  - This is **time-domain** representation
- We can convert to frequency-domain representation using Fourier transform and get a **power spectrum**
  - This is a **frequency-domain** representation
- Usually easier to analyze speech in the frequency domain than the time domain



Audio formats for neural nets

- Using just a single spectrum won't let us do anything interesting
- We need to use a time-frequency format
  - A spectrogram is a time-frequency format
- Most common format for audio is what is known as mel frequency cepstral coefficients (MFCCs)
  - At least historically…



MFCCs: Calculation

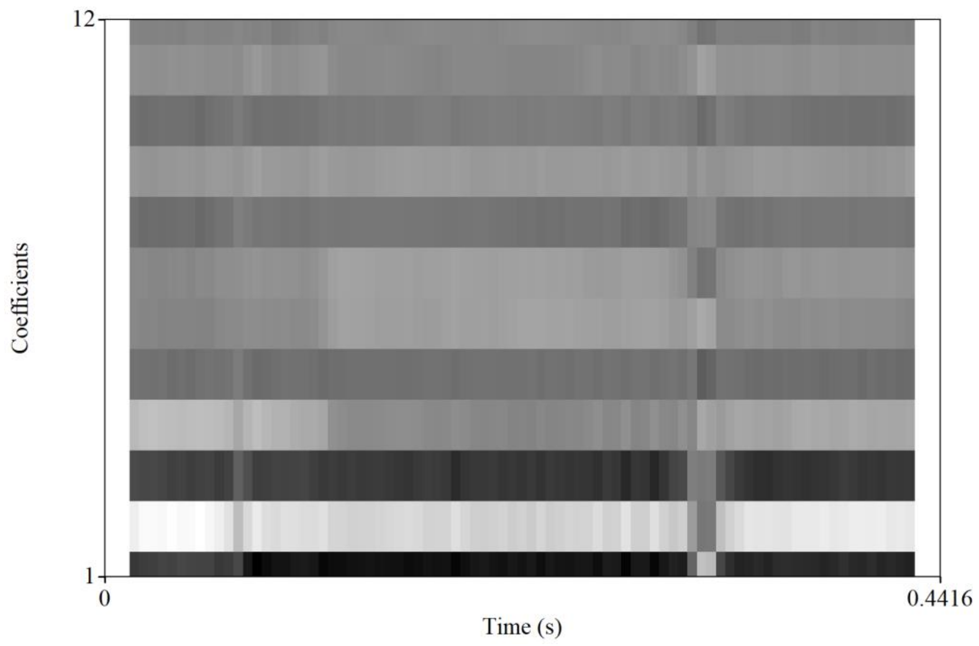
- To calculate
  1. Start with power spectrum
  2. Make frequency axis nonlinear along mel scale
  3. Apply a 40-channel filterbank across the nonlinear frequency axis
  4. Apply the discrete cosine transform to the filterbank
- MFCCs are useful because they are decorrelated from each other
  - This was important for some simpler HMM+GMM models for speech rec
- They are also a compact way of representing the speech spectrum

MFCCs: Interpretation

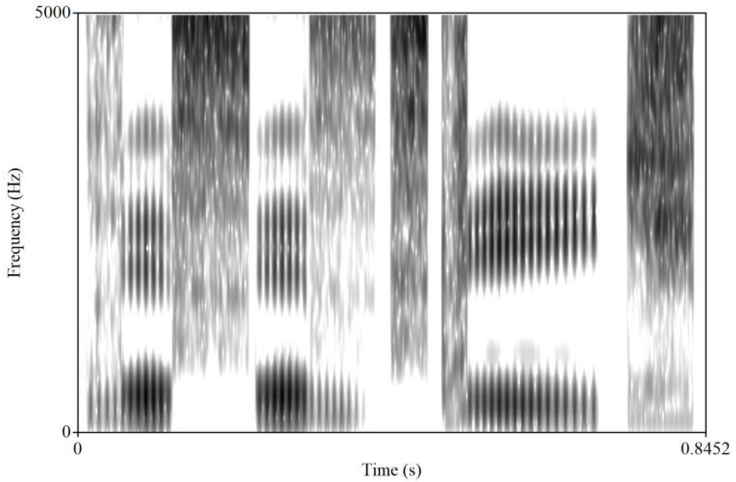
- “Cepstral” is used to refer to a spectrum of a spectrum
  - So, we have treated the power spectrum like a time-domain signal
- Each coefficient relates to a cosine wave of a different frequency being used to represent the power spectrum
  - Similar to the sine waves in the Fourier transform
- MFCCs are, effectively, a compression of the power spectrum

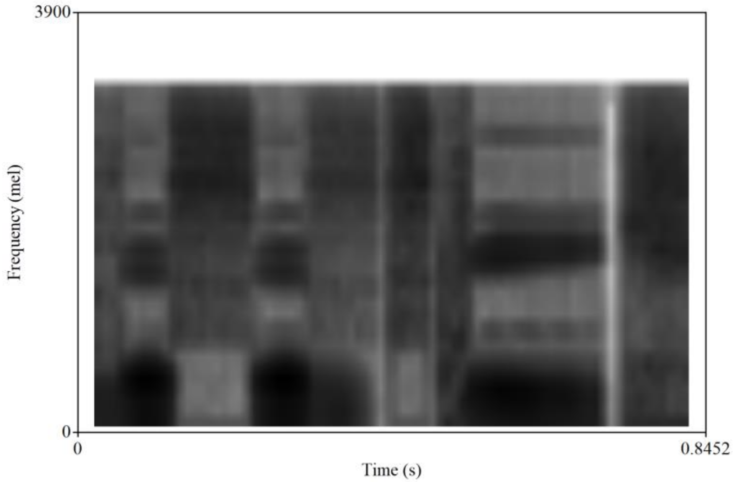
MFCCs as features

- As with spectrograms, we will need to calculate MFCCs at various time points of the speech signal
- Standard: calculate MFCCs over 25 ms windows of audio, spaced every 10 ms
- Also often use delta and delta-delta features
  - Discrete versions of 1st and 2nd derivative of MFCCs



Standard spectrogram





Mel spectrogram

(Log) mel spectrograms

- (Log) mel spectrograms became more popular with the popularity of neural nets
  - MFCCs had some convenient properties when using HMM+GMM models
- Have more information than MFCCs
- To calculate, do the same process as calculating MFCCs, but stop short of using the discrete cosine transform
  - May apply an elementwise log function on the energies too; this produces units of dB instead of power

Raw audio as input

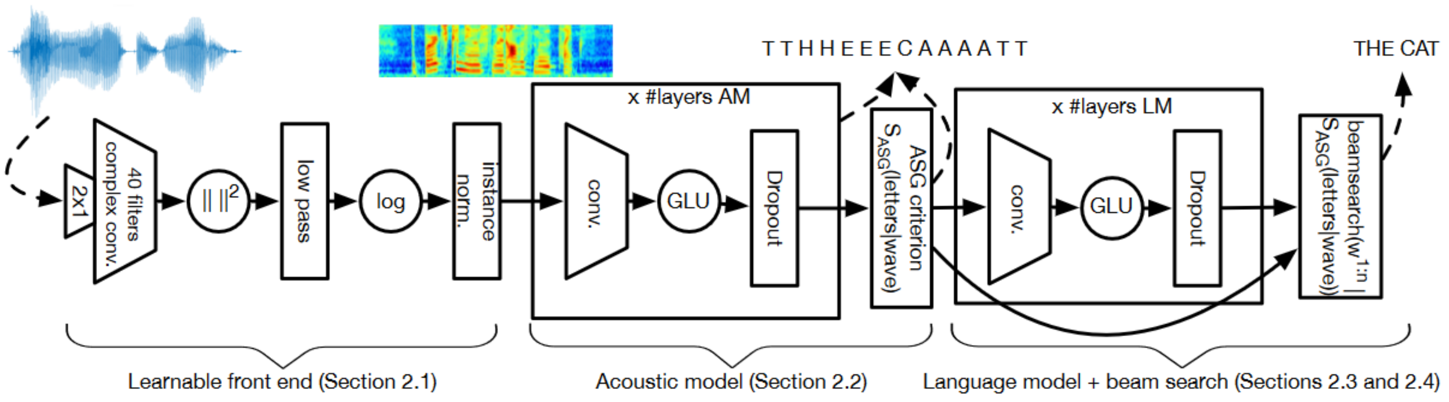


Figure 1: *Overview of the fully convolutional architecture.*

- If you configure some convolutional layers correctly, you can use raw audio as input
- See Zeghidour et al. (2018a, 2018b)

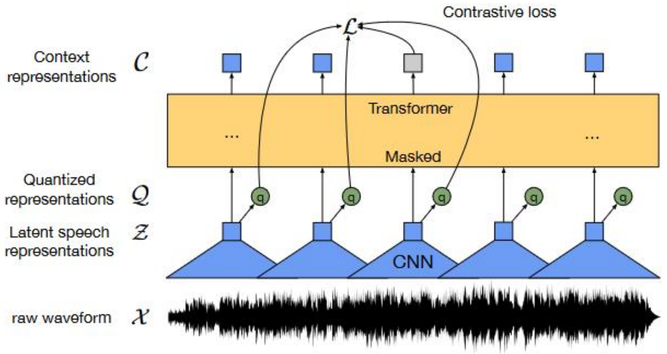


Figure 1: Illustration of our framework which jointly learns contextualized speech representations and an inventory of discretized speech units.

wav2vec

- Recent research has focused on self-supervised models to generate speech features relevant to many tasks at once (Baveski et al., 2020; Schneider et al., 2019)
- It is increasingly common to use wav2vec features or to fine tune large models for specific tasks/languages

Acoustic model

MAPPING FROM SPEECH TO RECOGNITION UNITS

What kind of output?

- We need to find a way to map from speech input to linguistic output
- It would be very convenient to map onto words themselves!
  - Unfortunately, this is difficult and has needed a lot of compute
- Instead, we map from acoustics to phones (traditionally) or letters (more modern)
- Then, we search to find an optimal match between words and acoustics

"Please call Stella. Ask her to bring these things with her from the store."

Common loss functions for ASR

- Categorical cross entropy (CCE, sometimes)
  - $CCE(\hat{y}, y) = -\sum_i \log\left(\frac{e^{\widehat{y_i}}}{\sum_j e^{\widehat{y}_j}}\right)$
  - Used for multiclass classification
  - Enforces separation of categories
- Connectionist temporal classification (CTC, Graves, 2006)
  - Too complicated to write on a single line (see next slide for a snippet)
  - Used for labeling problems where you have more time steps than labels
  - Collapses repeated characters, so [bbiiitttt] = [bbit] = [bit]
  - Uses a “blank” symbol to separate symbols and permit label collapsing

CTC code snippet

```julia
function ctc_alpha(ŷ::AbstractArray, y)
  typed_zero = zero(ŷ[1])
  ŷ = logsoftmax(ŷ)
  blank = size(ŷ, 1)
  z′ = add_blanks(y, blank)
  T = size(ŷ, 2)
  U′ = length(z′)

  α = fill(log(typed_zero), U′, T)
  α[1,1] = ŷ[blank, 1]
  α[2,1] = ŷ[z′[2], 1]
  for t=2:T
    bound = max(1, U′ - 2(T - t) - 1)
    for u=bound:U′
      if u == 1
        α[u,t] = α[u, t-1]
      else
        α[u,t] = logaddexp(α[u, t-1], α[u-1, t-1])

        # array bounds check and f(u) function from Eq. 7.9
        if u > 2 && !(z′[u] == blank || z′[u-2] == z′[u])
          α[u,t] = logaddexp(α[u,t], α[u-2,t-1])
        end
      end
      α[u,t] += ŷ[z′[u], t]
    end
  end
  return (loss=-1 * logaddexp(α[end,T], α[end-1, T]), alpha=α, zprime=z′, logsoftyhat=ŷ)
end
```

From the NNlib.jl deep learning function package in Julia: https://github.com/FluxML/NNlib.jl

(This code was actually contributed to the package by me [initially in Flux.jl]!)

Handling context

- Speech has allophonic relations and coarticulatory effects that need to be handled
  - Learning so called “context-free” phones ends up not working so well
  - That is, speech is sequential and contextual, just like text
- LSTMs were a go-to standard choice
- The advent of transformers gives another viable option for sequence modeling
- Convolutional layers are evergreen since they can extract abstract features from the raw audio
  - With enough depth, they can model sequences too

What relationship should we learn?

- Mapping acoustics directly to words is difficult
- More manageable to map to smaller units
- Phones are a more manageable choice, though letters/graphs are increasingly becoming a common output format
- Some researchers say they are classifying “phonemes,” though in practice they are classifying phones
  - Different disciplines are, of course, allowed to have different terminology
- We can use output sequence of phones to map to words since words can be represented as phones

More on relationships

- Often, goal is to minimize word error rate or phone error rate
- This is rather hard to directly optimize in the neural net training
  - Which is where the CTC loss comes in
- CTC loss considers all possible paths through phones/letters that will lead to the desired output

$$\text{Word Error Rate} \;=\; 100 \times \frac{\text{Insertions} + \text{Substitutions} + \text{Deletions}}{\text{Total Words in Correct Transcript}}$$

Decoding and language models

FROM MODEL OUTPUT TO WORDS

What is decoding?

- Our neural network has given us a series of probabilities of each phone label
  - This doesn't actually give us words yet!
- We need a way to algorithmically convert these probabilities to words
  - That is, we need to **decode** the network output
- If we know in advance what the words are, like for forced alignment, we can use a simpler search to determine an optimal sequence of phones that gives that word sequence
- Otherwise, we need to use more sophisticated algorithms
  - These algorithms often model language structure as well

Easiest decoding

- Choose the most probable phone at every time point
- Graves et al. (2006) call this **best path decoding**
- Works reasonably well considering how easy it is to implement
- Can result in poor labeling if the acoustic model is poor
  - And, the acoustic model is always poor… That is, never excellent

Decoding in practice

- How can we choose between possible outputs like “the stuff he knows will get him in trouble” or “the stuffy nose will get him in trouble”?
- Acoustic models aren’t perfect, so we help with language knowledge
  - Historically, *n*-gram probabilities like trigrams
  - Now, transformer-based LMs are becoming a common choice as well
- Language model paired with **beam search** to score possible outputs
- “stuff he knows” is more probable than “stuffy nose”
  - Though, trigrams may not actually capture this

Summary

- ASR with deep neural networks has four main parts

1. Acoustic input
   - Often represented as mel frequency cepstral coefficients
2. Acoustic model
   - Context-aware neural net that predicts phone/letter categories
3. Decoding algorithm
   - Usually beam search over the language’s vocabulary
4. Language model
   - Needs to yield probabilities over different word sequences

References

Amodei, D., Ananthanarayanan, S., Anubhai, R., Bai, J., Battenberg, E., Case, C., ... & Zhu, Z. (2016, June). Deep speech 2: End-to-end speech recognition in english and mandarin. In *International conference on machine learning* (pp. 173-182). PMLR.

Baevski, A., Zhou, Y., Mohamed, A., & Auli, M. (2020). wav2vec 2.0: A framework for self-supervised learning of speech representations. *Advances in neural information processing systems*, *33*, 12449-12460.

Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006). Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks. *Proceedings of the 23rd International Conference on Machine Learning*, 369–376. https://doi.org/10.1145/1143844.1143891

Graves, A. (2012). *Supervised sequence labelling with recurrent neural networks*. Springer.

Graves, A., & Jaitly, N. (2014). Towards end-to-end speech recognition with recurrent neural networks. *PMLR*, *32*, 1764–1772. http://proceedings.mlr.press/v32/graves14.html

Schneider, S., Baevski, A., Collobert, R., & Auli, M. (2019). wav2vec: Unsupervised pre-training for speech recognition. *arXiv preprint arXiv:1904.05862*.

Zeghidour, N., Usunier, N., Kokkinos, I., Schaiz, T., Synnaeve, G., & Dupoux, E. (2018, April). Learning filterbanks from raw speech for phone recognition. In *2018 IEEE international conference on acoustics, speech and signal Processing (ICASSP)* (pp. 5509-5513). IEEE.

Zeghidour, N., Xu, Q., Liptchinsky, V., Usunier, N., Synnaeve, G., & Collobert, R. (2018). Fully convolutional speech recognition. *arXiv preprint arXiv:1812.06864*.