Lecture 27: Exam 3 Review

Mark Hasegawa-Johnson
All content CC-BY 4.0 unless otherwise specified.

ECE 537, Fall 2022
1. Exam 3 Overview

2. Voice Conversion

3. CTC: Connectionist Temporal Classification

4. Transformer

5. Self-Supervised Learning

6. Summary
Exam 3 will be 12/16, 13:30-16:30, here.
If you need a conflict exam or an on-line exam, tell me in advance.
2/3 of the exam (about 6 questions) will cover voice conversion, CTC, Transformers, and self-supervised learning.
1/3 of the exam (about 3 questions) will be review of topics from exam 1 and exam 2.
You can bring two pages of handwritten notes, front and back.
No calculators.
Topics to be covered

- Material from exam 1: loudness, voder, pitch, speech production
- Material from exam 2: DTW, LPC, HMM
- New material:
  - Voice conversion: formant synthesis, neural nets
  - CTC: RNNs, CTC labeling, CTC forward-backward, loss gradient
  - Transformer: Dot-product similarity, Attention, Masking, Positional encoding
  - Self-supervised learning: CPC, HuBERT, VQ-VAE, Transposed Convolution
Outline

1. Exam 3 Overview
2. Voice Conversion
3. CTC: Connectionist Temporal Classification
4. Transformer
5. Self-Supervised Learning
6. Summary
Formant-based speech synthesis filters the excitation with

\[ R_k(z) = \frac{a_k}{1 - b_k z^{-1} - c_k z^{-2}}, \]

The coefficients are related to the formant frequency, \( F_k \), formant bandwidth, \( B_k \), and sampling frequency \( 1/T \) by

\[
\begin{align*}
    c_k &= -e^{-2\pi B_k T} \\
    b_k &= 2e^{-\pi B_k T} \cos(2\pi F_k T) \\
    a_k &= 1 - b_k - c_k
\end{align*}
\]

Klatt, 1980. (c) Acoustical Society of America
An example voice source is the LF model, which is determined by $T_0$ (the pitch period) plus four other parameters:

- $E_e$, amplitude of excitation
- $t_e$, time of the excitation
- time from upward-going zero-crossing, $t_c$, to downward-going zero-crossing, $t_p$
- slope of the return part, $\frac{E_e}{t_a}$

(c) http://www.speech.kth.se/gpsr
Formants can be converted from one voice to another using a neural network. Narendranath et al. (1995), Figure 4: (a) original voice, (c) conversion target, (b) conversion result.
Two-Layer Feedforward Neural Network

\[\vec{z} = h(\vec{x}, U, V)\]

\[z_\ell = g(b_\ell) \quad \vec{z} = g(\vec{b})\]

\[b_\ell = v_{k0} + \sum_{k=1}^{q} v_{\ell k} y_k \quad \vec{b} = V\vec{y}\]

\[y_k = f(a_k) \quad \vec{y} = f(\vec{a})\]

\[a_k = u_{k0} + \sum_{j=1}^{p} u_{kj} x_j \quad \vec{a} = U\vec{x}\]

\[\vec{x}\] is the input vector
Gradient Descent = Local Optimization

Given an initial $U, V$, find $\hat{U}, \hat{V}$ with lower error.

\[
\hat{u}_{kj} = u_{kj} - \eta \frac{\partial L}{\partial u_{kj}} \\
\hat{v}_{\ell k} = v_{\ell k} - \eta \frac{\partial L}{\partial v_{\ell k}}
\]

$\eta$ = Learning Rate

- If $\eta$ too large, gradient descent won’t converge. If too small, convergence is slow. Usually we pick $\eta \approx 0.001$ and cross our fingers.
- Second-order methods like L-BFGS and Adam choose an optimal $\eta$ at each step, so they’re MUCH faster.
Back-Propagation = Use Chain Rule to Find the Derivatives

\[ \mathcal{L} = \frac{1}{2} \sum_i \sum_\ell (z_\ell i - \zeta_\ell i)^2 \quad \Rightarrow \quad \frac{\partial \mathcal{L}}{\partial b_\ell i} = \frac{2}{n} (z_\ell i - \zeta_\ell i) g'(b_\ell i) \]

\[ \frac{\partial \mathcal{L}}{\partial u_{kj}} = \sum_{i=1}^n \left( \frac{\partial \mathcal{L}}{\partial a_{ki}} \right) \left( \frac{\partial a_{ki}}{\partial u_{kj}} \right) = \sum_{i=1}^n \left( \frac{\partial \mathcal{L}}{\partial a_{ki}} \right) x_{ji} \]

where...

\[ \left( \frac{\partial \mathcal{L}}{\partial a_{ki}} \right) = \sum_{\ell=1}^r \left( \frac{\partial \mathcal{L}_n}{\partial b_\ell i} \right) v_{\ell k} f'(a_{ki}) \]
Outline

1. Exam 3 Overview
2. Voice Conversion
3. CTC: Connectionist Temporal Classification
4. Transformer
5. Self-Supervised Learning
6. Summary
Recurrent Neural Net (RNN)

A recurrent neural net defines nonlinear recurrence of a hidden vector, $h_t$:

$$h_t = \sigma (Ux_t + Vh_{t-1})$$

$$y_t = \text{softmax} (Wh_t)$$

The weight matrices, $U$, $V$, and $W$, are chosen to minimize the loss function. For example, suppose we’re using a cross-entropy loss with target sequence $z_t$, then

$$\mathcal{L} = - \sum_{t=1}^{T} \ln y_{z_t}^t$$
Back-propagation through time does this:

$$\frac{d \mathcal{L}}{dh_t} = \frac{d \mathcal{L}}{dy_t} \frac{\partial y_t}{\partial h_t} + \frac{d \mathcal{L}}{dh_{t+1}} \frac{\partial h_{t+1}}{\partial h_t}$$
Many-to-One Mapping

The key idea of CTC is that, since $U \leq T$, the mapping from $y$ to $z$ is many-to-one. CTC makes repeated letters possible by using a blank character, $-$. The many-to-one mapping now has two steps: (1) eliminate all duplicate characters, (2) THEN eliminate all blanks.

$y = [f, -, e, e, d] \rightarrow \mathcal{B} \rightarrow z = [f, e, d]$  
$y = [f, e, -, e, d] \rightarrow \mathcal{B} \rightarrow z = [f, e, e, d]$  
$y = [f, -, f, e, d] \rightarrow \mathcal{B} \rightarrow z = [f, f, e, d]$
The temporal classification problem is now just:

\[ h(x) = \arg\max_{l \in L \leq T} p(l|x) \]

\[ = \arg\max_{l \in L \leq T} \sum_{\pi \in B^{-1}(l)} p(\pi|x) \]

\[ = \arg\max_{l \in L \leq T} \sum_{\pi \in B^{-1}(l)} \prod_{t=1}^{T} y_{\pi_t}^{t} \]

- \( l = [l_1, \ldots, l_V] \) is a label sequence of any length \( V \leq T \) where \( l_v \in L \).
- \( \pi = [\pi_1, \ldots, \pi_T] \) is a path of length \( T \) where \( \pi_t \in L \cup \{-\} \).
In order to express the CTC forward algorithm, we need to define a modified label sequence, $\mathbf{l}'$. $\mathbf{l}'$ is equal to $\mathbf{l}$ with blanks inserted between every pair of letters. Thus if

$$\mathbf{l} = [f, e, d],$$

then

$$\mathbf{l}' = [-, f, -, e, -, d, -].$$

If the length of $\mathbf{l}$ is $|\mathbf{l}|$, then the length of $\mathbf{l}'$ is $2|\mathbf{l}| + 1$. 
Repeating the same character ($\alpha_{t-1}(l'_{1:s})$) or adding one more character ($\alpha_{t-1}(l'_{1:s-1})$) are always possible. Adding two more characters ($\alpha_{t-1}(l'_{1:s-2})$) is OK if the current character is not a blank or a repeat.
Differentiating the CTC Loss

\[
\frac{d\mathcal{L}}{dy_k^\tau} = \left( \frac{-1}{p(z|x)} \right) \left( \frac{1}{y_k^\tau} \sum_{\pi \in \mathcal{B}^{-1}(z), \pi_\tau = k} \prod_{t=1}^{T} y_{\pi_t}^t \right)
\]

\[
= -\frac{\gamma_\tau(k)}{y_k^\tau},
\]

where

\[
\gamma_\tau(k) = p(\pi_\tau = k, z|x) = \frac{1}{y_k^\tau} \sum_{s: z_s' = k} \alpha_\tau(z'_{1:s}) \beta_\tau(z'_{s:(2U+1)}) \tag{1}
\]

- \(\beta_t(z'_{s:2U+1}) = p(z'_{s:(2U+1)}|x_t:T)\)
- Notice that \(\alpha_\tau(z'_{1:s})\) and \(\beta_\tau(z'_{s:(2U+1)})\) both include the fact that the network is producing \(z_s' = k\) at time \(\tau\). To compensate for that duplication, Eq. (1) has a \(\frac{1}{y_k^\tau}\) factor.
Outline

1. Exam 3 Overview
2. Voice Conversion
3. CTC: Connectionist Temporal Classification
4. Transformer
5. Self-Supervised Learning
6. Summary
The Cauchy-Schwartz Inequality

The Cauchy-Schwartz inequality says that, for any two vectors \( \vec{x} = [x_1, \ldots, x_N]^T \) and \( \vec{y} = [y_1, \ldots, y_N]^T \),

\[
|\vec{x}^T \vec{y}| \leq \|\vec{x}\|^2 \|\vec{y}\|^2
\]

If we define the unit vectors as follows,

\[
\hat{x} = \frac{\vec{x}}{\|\vec{x}\|}, \quad \hat{y} = \frac{\vec{y}}{\|\vec{y}\|},
\]

then the Cauchy-Schwartz inequality says that

\[-1 \leq \hat{x}^T \hat{y} \leq 1\]
**Attention**

Context Vector: \[ c(q^i) = \sum_{j=1}^{n} \alpha_{i,j} v^j \]

Attention: \[ \alpha_{i,j} = \frac{\exp \left( \text{Similarity}(q^i, k^j) \right)}{\sum_{j=1}^{n} \exp \left( \text{Similarity}(q^i, k^j) \right)} \]

- The query, \( q \) (sometimes \( q^i \)), is the vector whose context we want.
- The key, \( k \) (sometimes \( k^j \)), tells us whether or not \( v^j \) is useful context.
- The value, \( v \) (sometimes \( v^j \)), provides the actual context.
We assume that $q$ and $k$ have been transformed by some preceding neural net, so $q^T k$ is large if and only if they should be considered similar. Therefore the similarity score is

$$e_{i,j} = \frac{1}{\sqrt{d_k}} q^i k^j, T,$$

and the corresponding attention weight is

$$\alpha_{i,j} = \text{softmax}(e_{i,j}) = \frac{\exp(e_{i,j})}{\sum_{j=1}^{n} \exp(e_{i,j})}$$

$$\begin{bmatrix}
    \alpha_{1,1} & \cdots & \alpha_{1,n} \\
    \vdots & \ddots & \vdots \\
    \alpha_{n,1} & \cdots & \alpha_{n,n}
\end{bmatrix} = \text{softmax} \left( \frac{QK^T}{\sqrt{d_k}} \right)$$
Masking

If $q$, $k$, and $v$ are decoder vectors being produced autoregressively (e.g., decoder self-attention), then $c(q^i)$ can only depend on values of $v^j$ for $j < i$:

$$c(q^i) = \sum_{j=1}^{i-1} \alpha_{i,j} v^j$$

This can be done by setting $\alpha_{i,j} = 0$ for $j \geq i$. In turn, this can be done by masking the similarity scores as follows:

$$e_{i,j} = \frac{1}{\sqrt{d_k}} q^i k^j, T + m_i^j,$$

where

$$m_i^j = \begin{cases} 0 & j < i \\ -\infty & j \geq i \end{cases}$$
Multi-Head Attention

\[ \text{head}_i = \text{Attention} \left( QW_i^Q, KW_i^K, VW_i^V \right) \]
\[ = \text{softmax} \left( \frac{QW_i^Q W_i^K T K^T}{\sqrt{d_k}} \right) VW_i^V, \]

where the weight matrices \( W_i^Q, W_i^K, \) and \( W_i^V, \) for \( 1 \leq i \leq h, \) are learned matrices summarizing the type of context accumulated in each head. Then

\[ \text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_h) W^O, \]

where \( W^O \) is a final transformation that can, e.g., combine information from different heads in a learned manner.
Outline

1. Exam 3 Overview
2. Voice Conversion
3. CTC: Connectionist Temporal Classification
4. Transformer
5. Self-Supervised Learning
6. Summary
Unsupervised Learning Algorithms

- Manifold learning, e.g., Autoencoder:
  \[
  \mathcal{L}_{\text{MAE}} = E \left[ |x - x'| \right] \\
  \mathcal{L}_{\text{MSE}} = E \left[ \|x - x'\|^2 \right]
  \]

- Clustering
  - Classify each token to its nearest mean
  - Recompute each mean as the average of its tokens

- Self-supervised learning
  - The hyperplane is \( H_n^T Z_n = \text{threshold} \).
  - \( H_n \) is the average of all of the \( Z \) vectors on the right side of the hyperplane.
Recent Self-Supervised Learning for Speech

- **Contrastive Predictive Coding:** “Representation Learning with Contrastive Predictive Coding,” Oord, Li & Vinyals, 2018

\[
\mathcal{L}_{CPC} = - \sum_t \ln \frac{\exp (\text{Score}(x_{t+k}, c_t))}{\sum_{x \in X} \exp (\text{Score}(x, c_t))}
\]

- **Autoregressive Predictive Coding:** “Generative Pre-training for Speech with Autoregressive Predictive Coding,” Chung & Glass, 2020

\[
\mathcal{L} = \sum_{i=1}^{N-n} |x_{i+n} - y_i|
\]

- **Masked Language Modeling (HuBERT):** “HuBERT: Self-Supervised Speech Representation Learning by Masked Prediction of Hidden Units,” Hsu et al., 2021

\[
\mathcal{L} = - \sum_{t \in M} \ln \frac{\exp (\text{Score}(A_{ot}, e_c))}{\sum_{c' = 1}^{C} \exp (\text{Score}(A_{ot}, e_{c'}))}
\]
Summary: Discrete Disentangled Self-Supervised Representations

- Content is encoded using CPC, HuBERT, or VQ-VAE.
- Pitch is encoded using a VQ-VAE:
  
  $$z_q(x) = \arg\min_{j} \|z_e(x) - e_j\|_2$$

  $$\nabla_{z_e(x)} \mathcal{L} \approx \nabla_{z_q(x)} \mathcal{L}$$

- Speaker ID is encoded using a neural net trained to perform speaker discrimination.
Summary: Speech Resynthesis

- Speech resynthesis uses transposed convolution:
  \[
  [h_1, \ldots, h_{2L+1}] = [0, z_1, 0, z_2, 0, \ldots, 0, z_L, 0]
  \]
  \[
  y_i = \sum_{j=-D}^{D} K_j h_{i+j}
  \]

- Reconstruction minimizes absolute error of the mel-frequency spectrogram:
  \[
  \mathcal{L}_{\text{recon}} = \sum_{i=1}^{L} \| \phi(x) - \phi(\hat{x}) \|_1,
  \]

- GAN is used to avoid “regression to the mean,” to make sure speech sounds good:
  \[
  \mathcal{L}_G(D, G) = \sum_{j=1}^{J} [\mathcal{L}_{\text{adv}}(G, D_j) + \lambda_{fm}\mathcal{L}_{fm}(G, D_j)] + \lambda_r\mathcal{L}_{\text{recon}}(G)
  \]
Outline

1. Exam 3 Overview
2. Voice Conversion
3. CTC: Connectionist Temporal Classification
4. Transformer
5. Self-Supervised Learning
6. Summary
Topics to be covered

- Material from exam 1: loudness, voder, pitch, speech production
- Material from exam 2: DTW, LPC, HMM
- New material:
  - Voice conversion: formant synthesis, neural nets
  - CTC: RNNs, CTC labeling, CTC forward-backward, loss gradient
  - Transformer: Dot-product similarity, Attention, Masking, Positional encoding
  - Self-supervised learning: CPC, HuBERT, VQ-VAE, Transposed Convolution