Speech Processing

Karen Livescu

TOYOTA TECHNOLOGICAL INSTITUTE AT CHICAGO

Plan for this tutorial

- Yesterday: Intro to speech, historical tour of speech recognition research, speech recognition with hidden Markov models
- Today: Speech recognition with hybrid HMM/NNs and end-to-end recurrent neural networks, representation learning for speech
  - (Sorry, no language models)
- Lab exercise: Speech signals, speech recognition with HMMs and RNNs

Question from last time

Speech recognition error rates on low(er)-resource languages?

Question from last time

Speech recognition error rates on low(er)-resource languages?

Language	Language-specific	Joint	Joint + MTL
Bengali	19.1	16.8	**16.5**
Gujarati	26.0	**18.0**	18.2
Hindi	16.5	**14.4**	**14.4**
Kannada	35.4	**34.5**	34.6
Malayalam	44.0	36.9	**36.7**
Marathi	28.8	27.6	**27.2**
Tamil	13.3	10.7	**10.6**
Telugu	37.4	**22.5**	22.7
Urdu	29.5	26.8	**26.7**
Weighted Avg.	29.05	**22.93**	**22.91**

[Toshniwal+ 2018]

Model	Kazakh		Turkish		Haitian		Mongolian	
	WER	PER	WER	PER	WER	PER	WER	PER
Mono-lingual	55.9	40.9	53.1	36.2	49.0	36.9	58.2	45.2
Multi-lingual (MLing)	53.2	36.5	52.8	34.4	47.8	34.9	55.9	41.1
MLing & FineTuning (FT)	50.6	35.1	49.0	32.2	46.6	33.2	53.4	39.6
MLing + SWBD	52.3	36.6	51.3	33.0	45.8	33.9	54.5	40.2
MLing + SWBD & FT	**48.2**	**33.5**	**48.7**	**31.9**	**44.3**	**31.9**	**51.5**	**37.8**

[Dalmia+ 2018]

Outline

1. Speech recognition with hybrid HMM/NNs
2. Speech recognition with end-to-end neural models
3. Representation learning for speech

Hybrid HMM/NN models

Starting in the 1990s, long line of research by groups at ICSI Berkeley, IDIAP, and elsewhere (e.g., [Bourlard & Morgan 1994])

Hybrid HMM/NN models

Starting in the 1990s, long line of research by groups at ICSI Berkeley, IDIAP, and elsewhere (e.g., [Bourlard & Morgan 1994])

1. Train a frame-based neural network classifier of HMM states to produce a posterior $p(q|\mathbf{o})$, where q is the class and $\mathbf{o}$ is a frame feature vector

Hybrid HMM/NN models

Starting in the 1990s, long line of research by groups at ICSI Berkeley, IDIAP, and elsewhere (e.g., [Bourlard & Morgan 1994])

1. Train a frame-based neural network classifier of HMM states to produce a posterior $p(q|\mathbf{o})$, where q is the class and $\mathbf{o}$ is a frame feature vector
2. Convert $p(q|\mathbf{o})$ to something like an observation model (a "likelihood"): $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$

Hybrid HMM/NN models

Starting in the 1990s, long line of research by groups at ICSI Berkeley, IDIAP, and elsewhere (e.g., [Bourlard & Morgan 1994])

1. Train a frame-based neural network classifier of HMM states to produce a posterior $p(q|\mathbf{o})$, where q is the class and $\mathbf{o}$ is a frame feature vector
2. Convert $p(q|\mathbf{o})$ to something like an observation model (a "likelihood"): $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$
3. Use the result in place of the observation model in an HMM

Hybrid models

Main idea is simple:

- Use a NN to produce a posterior for each class ($=$ HMM state) given an input frame of acoustic features, $p(q|\mathbf{o})$
- Posterior is converted to a scaled likelihood via $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$

Hybrid models

Main idea is simple:

- Use a NN to produce a posterior for each class (= HMM state) given an input frame of acoustic features, $p(q|\mathbf{o})$
- Posterior is converted to a scaled likelihood via $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$

Where do the class labels for all frames come from?

Hybrid models

Main idea is simple:

- Use a NN to produce a posterior for each class (= HMM state) given an input frame of acoustic features, $p(q|\mathbf{o})$
- Posterior is converted to a scaled likelihood via $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$

Where do the class labels for all frames come from?

- Can be produced via a Viterbi alignment ("forced alignment") using an existing HMM/GMM system

Hybrid models

Main idea is simple:

- Use a NN to produce a posterior for each class (= HMM state) given an input frame of acoustic features, $p(q|\mathbf{o})$
- Posterior is converted to a scaled likelihood via $p(\mathbf{o}|q) \propto \frac{p(q|\mathbf{o})}{p(q)}$

Where do the class labels for all frames come from?

- Can be produced via a Viterbi alignment ("forced alignment") using an existing HMM/GMM system
- Or we can use "soft labels" = posteriors produced by running forward-backward using an existing HMM/GMM system
- (The latter makes sense if using cross-entropy loss)

How are HMMs used in speech research today?

- Hybrid HMM/NN models are close to (or at) state of the art for many benchmark data sets

How are HMMs used in speech research today?

- Hybrid HMM/NN models are close to (or at) state of the art for many benchmark data sets
- HMMs also used for **unsupervised** learning, e.g. discovering sound units in a low-resource language

How are HMMs used in speech research today?

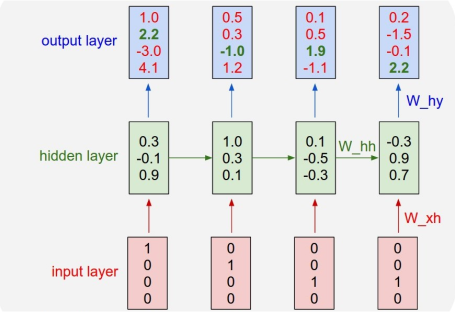
- Hybrid HMM/NN models are close to (or at) state of the art for many benchmark data sets
- HMMs also used for **unsupervised** learning, e.g. discovering sound units in a low-resource language
    - Given: a set of untranscribed speech
    - Train a single HMM on all of it
    - Look for repeated sequences of states to discover phones/words

Outline

1. Speech recognition with hybrid HMM/NNs
2. Speech recognition with end-to-end neural models
3. Representation learning for speech

Recurrent neural networks

Maintain a state vector in each frame, i.e. "remember" the past

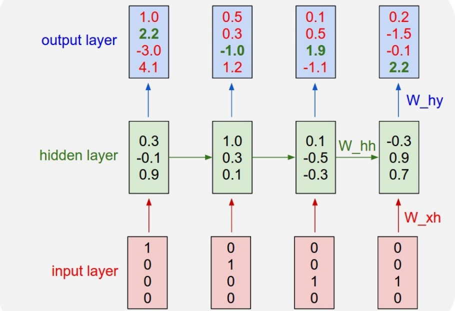


[Karpathy 2015]

$$
\begin{aligned}
\mathbf{h}_t &= \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\
\mathbf{y}_t &= \sigma_y(\mathbf{W}_{hy}\mathbf{h}_t + \mathbf{b}_y)
\end{aligned}
$$

Recurrent neural networks

Maintain a state vector in each frame, i.e. “remember” the past



[Karpathy 2015]

- Can be made deep, by stacking multiple layers of hidden states
- Can be made bidirectional, by combining a layer with forward connections and one with backward connections
- Variants: Long short-term memory (LSTM) networks, gated recurrent unit (GRU) networks

Using RNNs in hybrid HMM/NN models

If we add a softmax at the output layer and train with cross-entropy loss, the outputs approximate posterior probabilities of labels (HMM states):

$$y_t^i \approx p(q_t = i | \mathbf{x}_1, \ldots, \mathbf{x}_t)$$

Using RNNs in hybrid HMM/NN models

If we add a softmax at the output layer and train with cross-entropy loss, the outputs approximate posterior probabilities of labels (HMM states):

$$y_t^i \approx p(q_t = i | \mathbf{x}_1, \ldots, \mathbf{x}_t)$$

- Given this, we can produce scaled likelihoods and use them in a hybrid model, as before
- Or, we can drop the HMMs and use RNNs for **end-to-end** speech recognition...

End-to-end RNNs for speech recognition

Hybrid models involve a lot of machinery...

- Train a simple HMM/GMM recognizer
- Run Viterbi with HMM/GMM to get per-frame state labels
- Train neural network state classifier
- Compute state prior probabilities
- Scale classifier outputs by state priors to get scaled HMM observation model
- Train HMM/NN

End-to-end RNNs for speech recognition

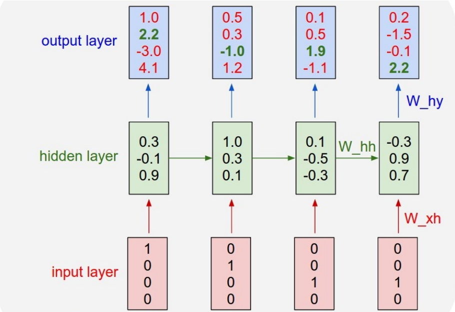
Hybrid models involve a lot of machinery...

- Train a simple HMM/GMM recognizer
- Run Viterbi with HMM/GMM to get per-frame state labels
- Train neural network state classifier
- Compute state prior probabilities
- Scale classifier outputs by state priors to get scaled HMM observation model
- Train HMM/NN

Can we train an RNN to map directly from acoustic input sequence to output text sequence?

End-to-end RNNs for speech recognition

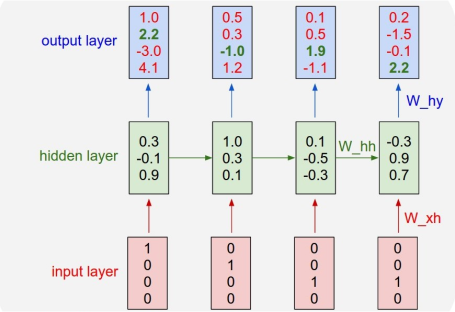
We could train an RNN to output a text label (word, character) at each frame...



[Karpathy 2015]

End-to-end RNNs for speech recognition

We could train an RNN to output a text label (word, character) at each frame...

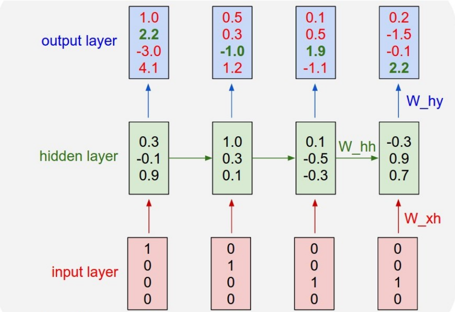


[Karpathy 2015]

- But what does this mean? Words and characters usually span many frames of speech

End-to-end RNNs for speech recognition

We could train an RNN to output a text label (word, character) at each frame...



[Karpathy 2015]

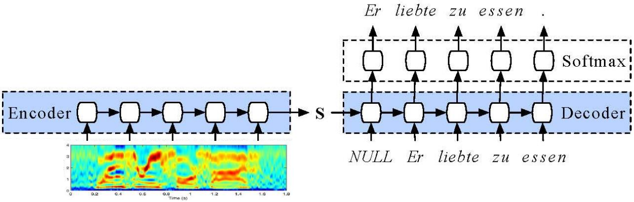
- But what does this mean? Words and characters usually span many frames of speech
- And the alignment between frames and characters/words is not simple

End-to-end RNNs for speech recognition

Two typical approaches:

- Encoder-decoder (“sequence-to-sequence”) models
- Connectionist temporal classification (CTC)

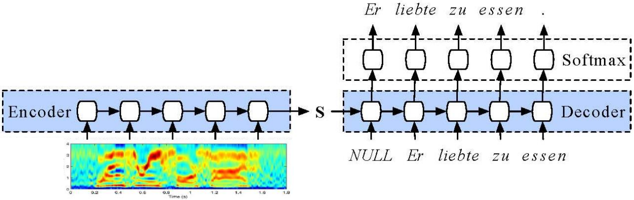
Encoder-decoder ("sequence-to-sequence") RNNs



[Fig. credit smerity.com]

- Can be trained directly to optimize $p(\mathbf{y}|\mathbf{x})$ without aligning the input and output
- Input (acoustic frames) and output (characters/words) don't even have to operate at the same rate!
- Introduced for machine translation [Cho+ 2014, Sutskever+ 2014]

Encoder-decoder RNNs in more detail

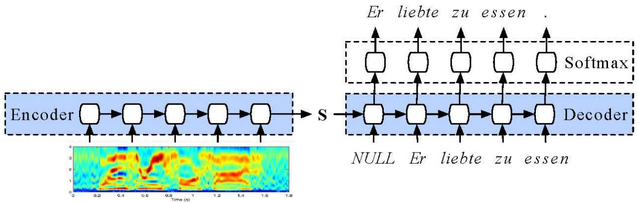


[Fig. credit smerity.com]

"Vanilla" RNN encoder-decoder equations:

$$\begin{aligned}
&\text{Encoder}: \\
\mathbf{h}_t &= \sigma_h(\mathbf{W}_{xh}\mathbf{x}_t + \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{b}_h) \\
\mathbf{s} &= \mathbf{h}_T = \mathbf{s}_0 \\
&\text{Decoder}: \\
\mathbf{s}_j &= \sigma_s(\mathbf{W}_{ys}\mathbf{y}_{j-1} + \mathbf{W}_{ss}\mathbf{s}_{j-1} + \mathbf{b}_s) \\
\mathbf{f}_j &= \text{softmax}(\mathbf{W}_{sy}\mathbf{s}_j + \mathbf{b}_y) \\
\hat{y}_j &= \text{argmax}\, \mathbf{f}_j
\end{aligned}$$

Encoder-decoder RNNs in more detail



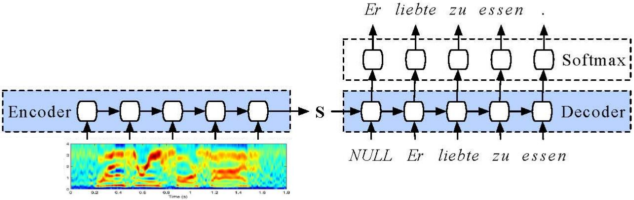
“Vanilla” RNN encoder-decoder equations:

$$\text{Decoder}:$$

$$\begin{aligned}
\mathbf{s}_j &= \sigma_s(\mathbf{W}_{ys}\mathbf{y}_{j-1} + \mathbf{W}_{ss}\mathbf{s}_{j-1} + \mathbf{b}_s) \\
\mathbf{f}_j &= \mathrm{softmax}(\mathbf{W}_{sy}\mathbf{s}_j + \mathbf{b}_y) \\
\hat{y}_j &= \mathrm{argmax}\,\mathbf{f}_j
\end{aligned}$$

- Here $\mathbf{y}_j$ is a “one-hot” vector representing $\hat{y}_j$
- Interpretation: $\mathbf{f}_{jd}$ is the probability of the next word being the d^{th} word in the vocabulary, given the previous words and the acoustic input

Encoder-decoder RNNs in more detail

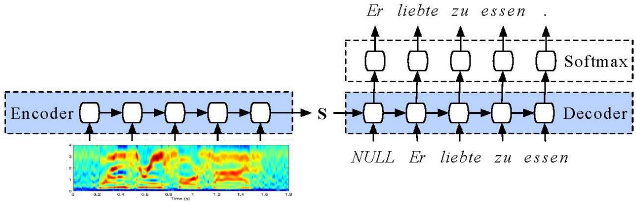


"Vanilla" RNN encoder-decoder equations:

$$
\begin{aligned}
&\text{Decoder}: \\
\mathbf{s}_j &= \sigma_s(\mathbf{W}_{ys}\mathbf{y}_{j-1} + \mathbf{W}_{ss}\mathbf{s}_{j-1} + \mathbf{b}_s) \\
\mathbf{f}_j &= \mathrm{softmax}(\mathbf{W}_{sy}\mathbf{s}_j + \mathbf{b}_y) \\
\hat{y}_j &= \mathrm{argmax}\,\mathbf{f}_j
\end{aligned}
$$

- Typical loss: cross-entropy (log loss)
  $-\log p(y_{1:J}|\mathbf{x}_{1:T}) = -\sum_{j=1}^{J} \log \mathbf{f}_{jd}$
- where $y_{1:J}$ = ground-truth label sequence corresponding to input sequence $\mathbf{x}_{1:T}$ and d is the index of y_j in the vocabulary

Encoder-decoder RNNs in more detail



Can be extended to a variety of types of encoder and decoder RNNs

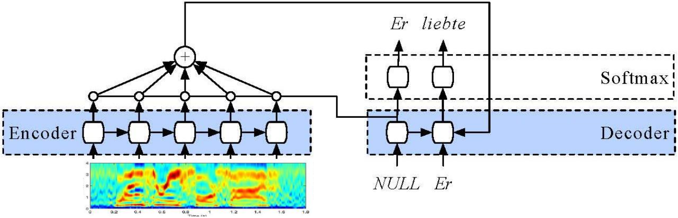
- LSTM/GRU instead of vanilla RNN units
- Deep encoder, deep decoder (less typical)
- Bidirectional encoder

Attention models

- Basic encoder-decoder models represent the entire input sequencey with a single vector
- That's a lot to ask of a single vector...
- Basic encoder-decoders don't work well out of the box. Much better when endowed with **attention**:

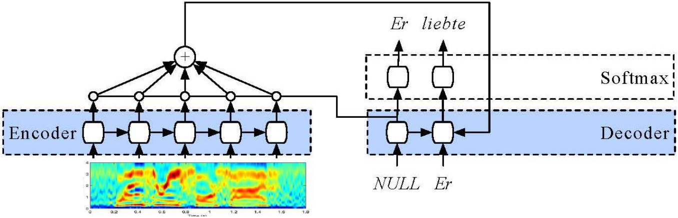
Attention models

- Basic encoder-decoder models represent the entire input sequencey with a single vector
- That's a lot to ask of a single vector...
- Basic encoder-decoders don't work well out of the box. Much better when endowed with **attention**:



- In attention models, each decoder state depends on a weighted combination of encoder states (a "context vector")
- These weights are an "attention vector"
- The attention vector is itself a function of the input and output, with learned parameters

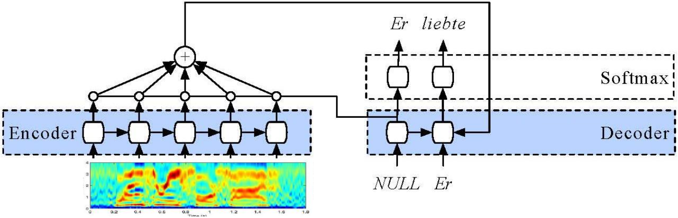
Attention models



Example:

$$
\begin{aligned}
\text{Decoder}: & & \\
\mathbf{s}_j &= \sigma_s(\mathbf{W}_{ys}\mathbf{y}_{j-1} + \mathbf{W}_{ss}\mathbf{s}_{j-1} + \mathbf{b}_s) \\
\mathbf{f}_j &= \text{softmax}(\mathbf{W}_{sy}[\mathbf{c}_j;\mathbf{s}_j] + \mathbf{b}_y) \\
\hat{y}_j &= \text{argmax}\, \mathbf{f}_j \\
\text{Contextvector}: & & \\
\mathbf{c}_j &= \sum_{t=1}^{T} \alpha_{jt}\mathbf{h}_t \\
\mathbf{alpha}_j &= \text{softmax}(\mathbf{u}_j) \\
\mathbf{u}_{jt} &= \mathbf{h}_t^T \mathbf{s}_j
\end{aligned}
$$

Attention models



Other ways of computing context vector:

$$
\begin{aligned}
\mathbf{c}_j &= \sum_{t=1}^{T} \alpha_{jt} \mathbf{h}_t \\
\mathbf{alpha}_j &= \mathrm{softmax}(\mathbf{u}_j) \\
\mathbf{u}_{jt} &= \mathbf{h}_t^T \mathbf{s}_j \\
\text{OR } \mathbf{u}_{jt} &= \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \mathbf{s}_j + \mathbf{b}_a) \\
\text{OR } \mathbf{u}_{jt} &= \mathbf{v}^T \tanh(\mathbf{W}_1 \mathbf{h}_t + \mathbf{W}_2 \mathbf{s}_j + \mathbf{W}_f \mathbf{f}_{jt} + \mathbf{b}_a) \\
\text{where } \mathbf{f}_j &= \mathbf{F} * \mathbf{alpha}_{j-1}
\end{aligned}
$$

(The last is sometimes called "location-aware" or "convolutional" attention)

Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

- To match test-time model, should set $\mathbf{y}_j$ to a one-hot vector representing $\hat{y}_{j-1}$

Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

- To match test-time model, should set $\mathbf{y}_j$ to a one-hot vector representing $\hat{y}_{j-1}$
- ... but then the training model would produce garbage output until it is trained well

Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

- To match test-time model, should set $\mathbf{y}_j$ to a one-hot vector representing $\hat{y}_{j-1}$
- ... but then the training model would produce garbage output until it is trained well
- Or, we could set $\mathbf{y}_j$ to a one-hot vector representing the ground-truth y_{j-1}

Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

- To match test-time model, should set $\mathbf{y}_j$ to a one-hot vector representing $\hat{y}_{j-1}$
- ... but then the training model would produce garbage output until it is trained well
- Or, we could set $\mathbf{y}_j$ to a one-hot vector representing the ground-truth y_{j-1}
- ... but then we are training and testing with different models

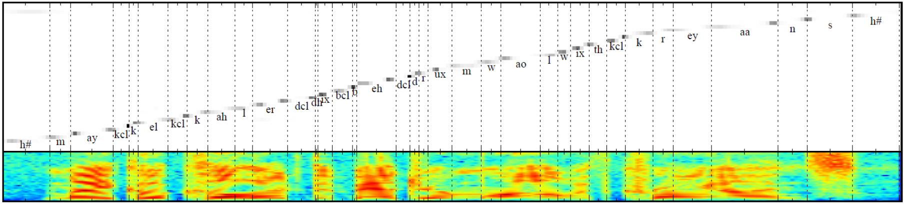
Scheduled sampling [Bengio+ 2015]

During training, what should the input $\mathbf{y}_j$ to the decoder be?

- To match test-time model, should set $\mathbf{y}_j$ to a one-hot vector representing $\hat{y}_{j-1}$
- ... but then the training model would produce garbage output until it is trained well
- Or, we could set $\mathbf{y}_j$ to a one-hot vector representing the ground-truth y_{j-1}
- ... but then we are training and testing with different models
- Scheduled sampling: At each iteration of training, use the ground-truth label with some probability ϵ and the model's previous prediction with probability $1 - \epsilon$

Visualizing attention

FDHC0_SX209: Michael colored the bedroom wall with crayons.



[Chorowski+ 2015]

Attention models: Summary

Attention models have achieved state-of-the-art performance on some speech benchmarks. But:

- They are computationally demanding: Each output label considers entire input sequence
- But the alignment between the acoustics and labels is largely monotonic
- So maybe attention models are overkill?

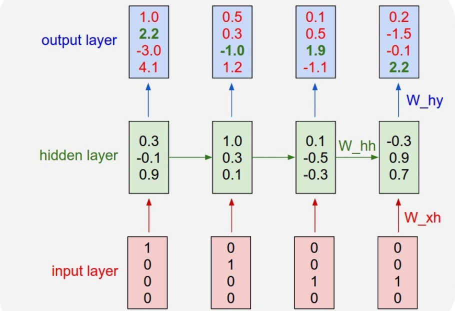
Rewind: End-to-end RNNs for speech recognition

Two typical approaches:

- Encoder-decoder (“sequence-to-sequence”) models
- Connectionist temporal classification (CTC)

Rewind: End-to-end RNNs for speech recognition

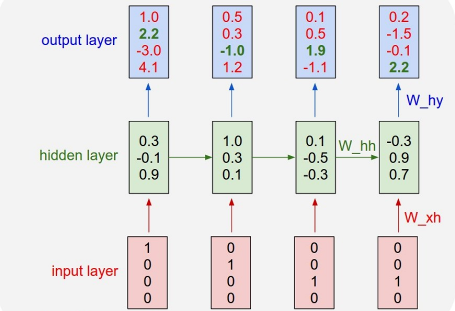
We could train an RNN to output a text label (word, character) at each frame...



[Karpathy 2015]

Rewind: End-to-end RNNs for speech recognition

We could train an RNN to output a text label (word, character) at each frame...

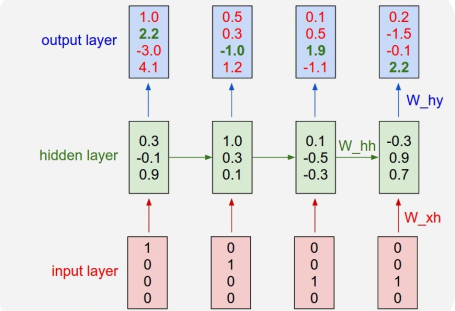


[Karpathy 2015]

- But what does this mean? Words and characters usually span many frames of speech

Rewind: End-to-end RNNs for speech recognition

We could train an RNN to output a text label (word, character) at each frame...



[Karpathy 2015]

- But what does this mean? Words and characters usually span many frames of speech
- And the alignment between frames and characters/words is not simple, and often ambiguous. Consider the word **through**: Which frames does the **g** correspond to?
- But given the mostly monotonic alignment between frames and labels, maybe we were too dismissive?

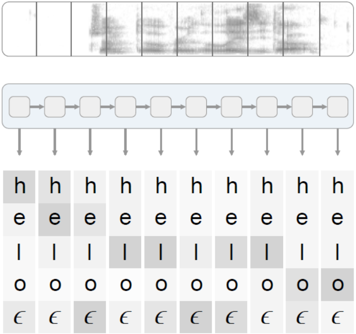
Connectionist temporal classification (CTC) [Graves+ 2006]

CTC modifies the per-frame RNN labeler idea in two key ways:

- An extra "blank" label ϵ
- A mapping from frame-level label sequences to true label sequences

Connectionist temporal classification (CTC) [Graves+ 2006]

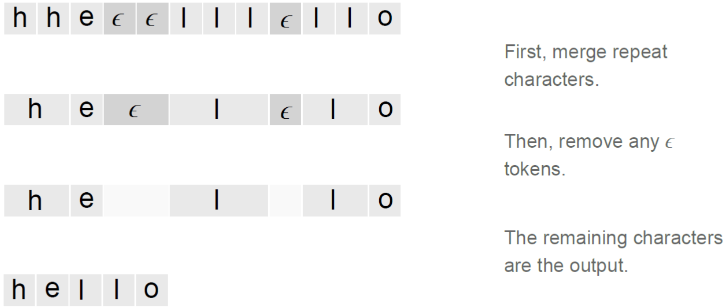
From `https://distill.pub/2017/ctc/` [Hannun 2017]:



RNN with softmax output layer produces a posterior probability for each label $+\ \epsilon$

Basic ("greedy") CTC decoding

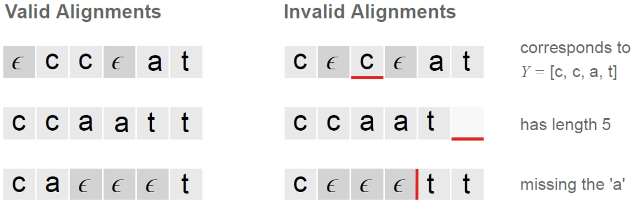
- RNN with softmax output layer produces a posterior probability for each label + ϵ
- At each time frame, output the most likely frame label
- Finally, map frame labels to "collapsed" label sequence as follows:



[Hannun 2017]

CTC training

Given a sequence X of N acoustic frames and a corresponding label sequence Y with $L < N$ labels, consider the set of all of the valid frame label sequences ("alignments") $\mathcal{A}_{X,Y}$



[Hannun 2017]

(Example for the word **cat**)

CTC training

Given a sequence X of T acoustic frames and a corresponding label sequence Y with $L < N$ labels, e.g. the word **cat**, consider the set of all of the valid frame label sequences ("alignments") $\mathcal{A}_{X,Y}$.

Then the CTC loss is a *marginal log loss*:

$-\log p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X)$

where $p_t(a_t|X)$ is the softmax output of the RNN at frame t

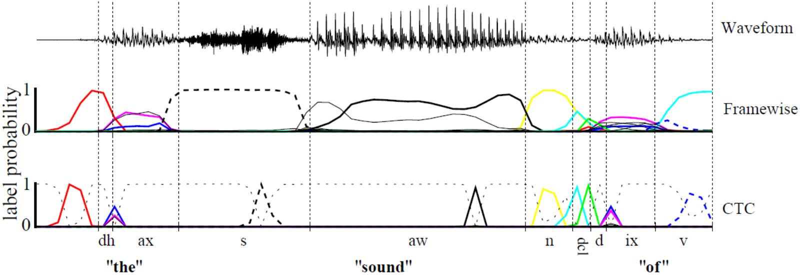
CTC training

CTC loss:
$-\log p(Y|X) = \sum_{A \in \mathcal{A}_{X,Y}} \prod_{t=1}^{T} p_t(a_t|X)$

Looks hard to backprop, but it turns out to be equivalent to a forward-backward-like HMM algorithm!

CTC posterior visualization

CTC posteriors vs. posteriors from an RNN trained with frame-level log loss (e.g. for a hybrid HMM/NN):



[Graves+ 2006]

CTC not so different from HMMs...

Putting HMM into CTC-like notation (A = state sequence):

$p(X|Y) = \sum_{A \in \mathcal{A}} p(X, A|Y)$

Dropping the conditioning on Y:

$p(X) = \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t|a_t) p(a_t|a_{t-1})$

Suppose transition probabilities are uniform:

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t|a_t)$

CTC not so different from HMMs...

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(\mathbf{x}_t | a_t)$

2 differences from CTC:

- $p(\mathbf{x}_t | a_t)$ vs. $p(a_t | \mathbf{x}_t)$
- Definition of $\mathcal{A}$

Rewrite using Bayes rule (as we did for hybrid models):

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(a_t | \mathbf{x}_t) / p(a_t)$

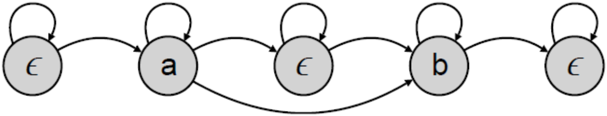
Assuming uniform priors of the labels (states):

$p(X) \propto \sum_{A \in \mathcal{A}} \prod_{t=1}^{T} p(a_t | \mathbf{x}_t)$

So computing the marginal probability in CTC is just like computing likelihood in HMMs, hence forward-backward algorithm!

Equivalent HMM state diagram for CTC

Assuming the ground-truth sequence “a b”:



Connection between CTC and encoder-decoder models

- Encoder = all but last layer of the RNN
- Decoder = softmax + label collapsing function

CTC: Summary

- CTC-based models are state-of-the-art on many speech benchmarks
- Much faster than attention models
- Tend to require more data to train well

End-to-end neural speech recognition: Addendum

This is a very active area of research

- Models we haven't discussed: transformers, dilated convolution-based models, ...
- Issues we haven't discussed:
  - Choice of output labels (words, characters, other sub-word units)
  - How to combine with a language model

Outline

1. Speech recognition with hybrid HMM/NNs
2. Speech recognition with end-to-end neural models
3. Representation learning for speech

“Tandem” models

- Grew out of early work on hybrid HMM/NN models
- Developed at ICSI Berkeley (e.g., Hermansky et al. 2000)
- Main idea: Never mind the whole posterior conversion business; use the NN classifier outputs as features!

"Tandem" models

- Grew out of early work on hybrid HMM/NN models
- Developed at ICSI Berkeley (e.g., Hermansky et al. 2000)
- Main idea: Never mind the whole posterior conversion business; use the NN classifier outputs as features!
    - Then use standard HMM/GMMs with these features as inputs

“Tandem” models

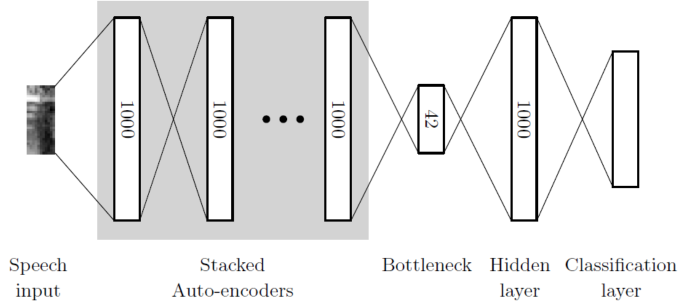
- Grew out of early work on hybrid HMM/NN models
- Developed at ICSI Berkeley (e.g., Hermansky et al. 2000)
- Main idea: Never mind the whole posterior conversion business; use the NN classifier outputs as features!
    - Then use standard HMM/GMMs with these features as inputs
    - $\mathbf{o}' = [y_1(\mathbf{o})\ y_2(\mathbf{o})\ \ldots y_n(\mathbf{o})]$

"Tandem" models

- Grew out of early work on hybrid HMM/NN models
- Developed at ICSI Berkeley (e.g., Hermansky et al. 2000)
- Main idea: Never mind the whole posterior conversion business; use the NN classifier outputs as features!
  - Then use standard HMM/GMMs with these features as inputs
  - $\mathbf{o}' = [y_1(\mathbf{o})\ y_2(\mathbf{o})\ \dots y_n(\mathbf{o})]$
  - If the $y_i(\mathbf{o})$ represent probabilities, then we typically take their logs: $\mathbf{o}' = [\log(f_1(\mathbf{o}))\ \log(f_2(\mathbf{o}))\ \dots]$

Tandem models

Alternatively, use outputs from a lower layer, and make that layer narrow (a "bottleneck layer") to reduce dimensionality



[Gehring+ 2013]

Tandem models: More details

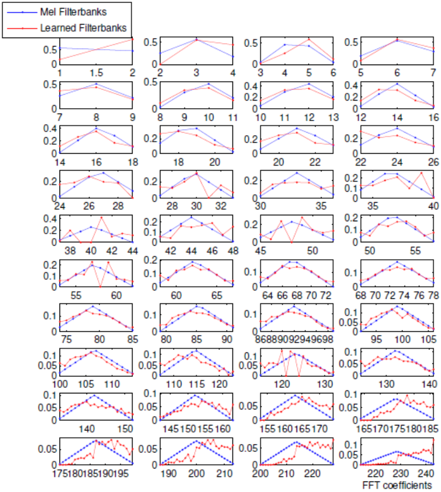
- These features are often appended to the original features, so the new feature vector is $[\mathbf{o}\ \mathbf{o}']$ (hence, "tandem"!)

Tandem models: More details

- These features are often appended to the original features, so the new feature vector is $[\mathbf{o}\ \mathbf{o}']$ (hence, "tandem"!)
- Typically, the input is a concatenation of acoustic vectors over a window of 7-20 frames around the current frame (very high-dimensional!)

Tandem models: Learned representations

When using spectrograms as input representation, the learned filters are similar to the triangular filters of the inner ear:



nath+ 2013]

Tandem models: Summary

- Tandem models were state-of-the-art around the late 2000s
- Both supervised and unsupervised "bottleneck" representations were attempted, but only the supervised ones performed well
- Extensions to multilingual learning of bottleneck features have been quite successful

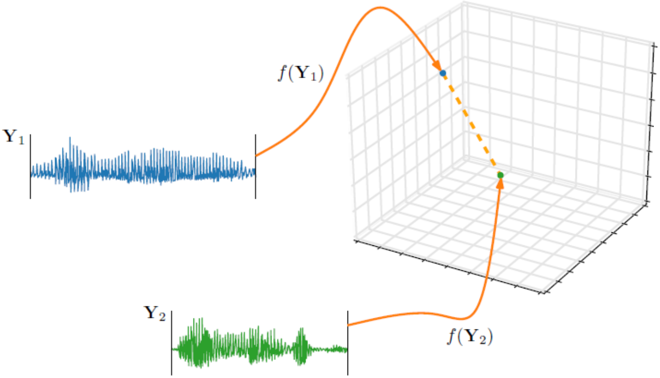
Current research: Acoustic word embeddings

Frame representations are useful, but for some tasks we would like a represention of whole words (or other units) of arbitrary duration

Current research: Acoustic word embeddings

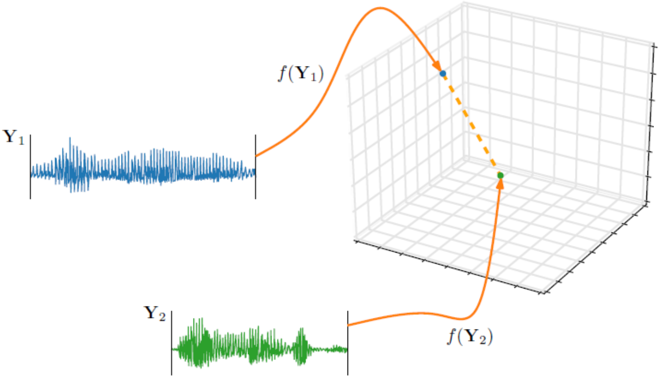
Frame representations are useful, but for some tasks we would like a represention of whole words (or other units) of arbitrary duration

- Acoustic word embeddings: map from a spoken word to a vector
- "Spoken word" = speech signal of arbitrary length corresponding to a word



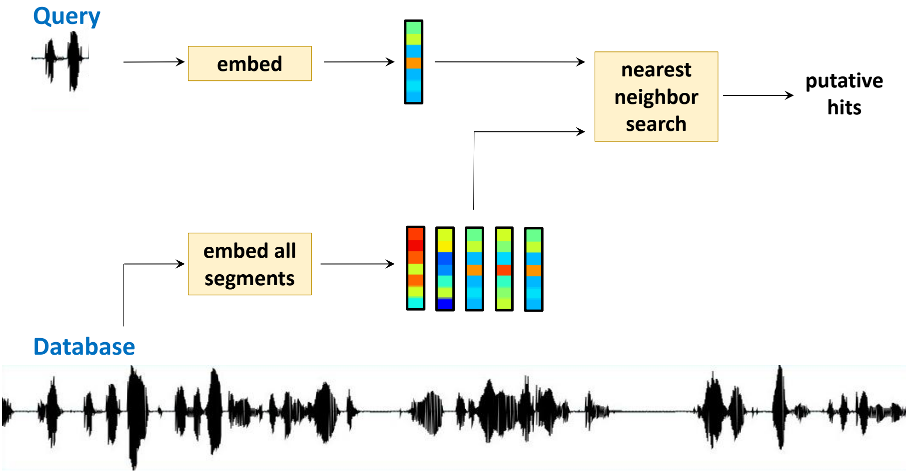
Applications of acoustic word embeddings

- Speech search [Parada+ 2015, Settle+ 2017, Audhkhasi+ 2017]
- Whole-word speech recognition [Maas+ 2012, Bengio & Heigold 2014, Settle+ 2019]
- Spoken term discovery [Kamper+ 2014-2018]



[Figure credit: Herman Kamper]

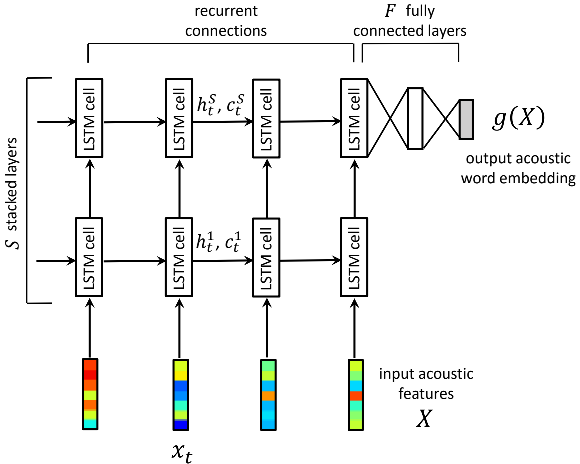
Query-by-example search with acoustic word embeddings [Settle+ 2017]



[Figure credit: Herman Kamper]

Neural embeddings: RNN-based [SLT 2016]

- **Input:** MFCCs (without padding)
- **Model:** n_{rec} recurrent + n_{full} fully connected layers
- **Embedding** is activation vector of final fully connected layer



Training objectives

Word classifier log loss

- Add a softmax layer to predict word w
- $l(\mathbf{x}, w) = \log p(w|\mathbf{x})$

Training objectives

Word classifier log loss

- Add a softmax layer to predict word w
- $l(\mathbf{x}, w) = \log p(w|\mathbf{x})$

Contrastive (triplet) loss

- Bring together same-word pairs, separate different ones

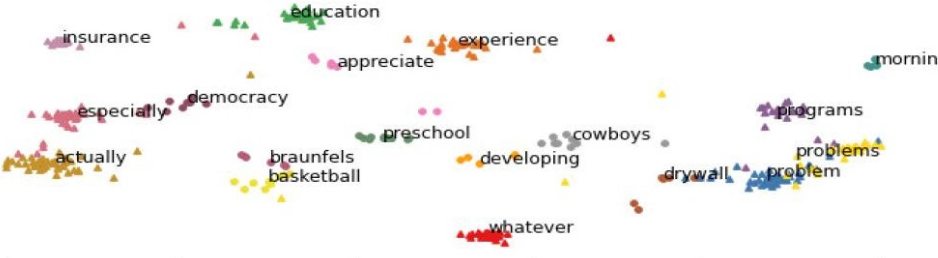
$$l(\mathbf{x}_1, \mathbf{x}_2) = max\{0, m + d_{\cos}(\mathbf{x}_1, \mathbf{x}_2) - d_{\cos}(\mathbf{x}_1, \mathbf{x}^-)\}$$

  where $\mathbf{x}^- =$ random (or hard) negative example, $m =$ margin
- Weaker supervision (no word labels, only same-word pairs)

Visualization: RNN embeddings

2-dimensional t-SNE embeddings [van der Maaten & Hinton 2008]

△ = word types seen at training time
◯ = not seen at training time



These embeddings far outperform a standard (dynamic time warping-based) approach to query-by-example search

Joint learning of acoustic + written word embeddings [He+ 2017, Settle+ 2019, Collobert+ 2019]

Motivation:

- Learn better acoustic embeddings by relating them to a written character sequence
- Some tasks involve "distances" between speech segments and written words
  - Spoken term detection ("Query-by-text")
  - Automatic speech recognition

"Barack Obama" $\stackrel{?}{=}$ 

Joint learning of acoustic + written word embeddings [He+ 2017, Settle+ 2019, Collobert+ 2019]

Motivation:

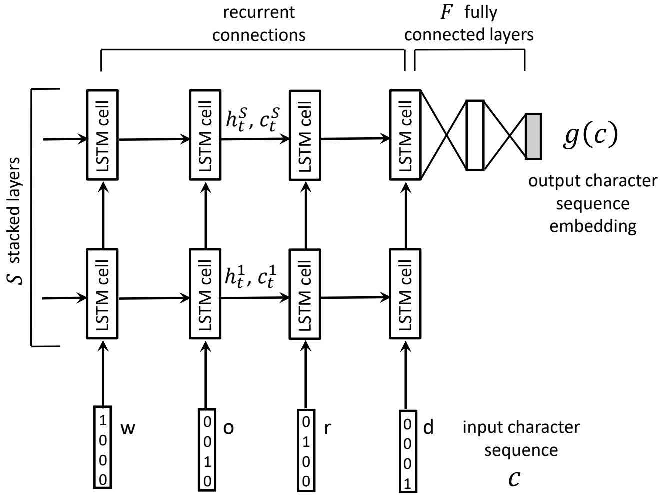
- Learn better acoustic embeddings by relating them to a written character sequence
- Some tasks involve "distances" between speech segments and written words
    - Spoken term detection ("Query-by-text")
    - Automatic speech recognition

Approach: Learn a pair of RNN-based embedding functions

- Acoustic word embedding (speech $\rightarrow$ vector)
- Acoustically grounded word embedding (character sequence $\rightarrow$ vector)

"Barack Obama" $\stackrel{?}{=}$ 

Character RNN-based acoustically grounded word embedding



Joint learning of acoustic and acoustically grounded word embeddings

Given a matched (acoustic, written) word pair $(\mathrm{x}, \mathbf{c})$

$$l_0(\mathbf{x}, \mathbf{c}) \quad = \quad max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{c}^-)\}$$

Joint learning of acoustic and acoustically grounded word embeddings

Given a matched (acoustic, written) word pair $(\mathbf{x}, \mathbf{c})$

$$\begin{aligned} l_0(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{c}^-)\} \\ l_1(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{c}^-, \mathbf{c})\} \end{aligned}$$

Joint learning of acoustic and acoustically grounded word embeddings

Given a matched (acoustic, written) word pair $(\mathbf{x}, \mathbf{c})$

$$\begin{aligned}
l_0(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{c}^-)\} \\
l_1(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{c}^-, \mathbf{c})\} \\
l_2(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}^-, \mathbf{c})\} \\
l_3(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{x}^-)\}
\end{aligned}$$

Joint learning of acoustic and acoustically grounded word embeddings

Given a matched (acoustic, written) word pair $(\mathbf{x}, \mathbf{c})$

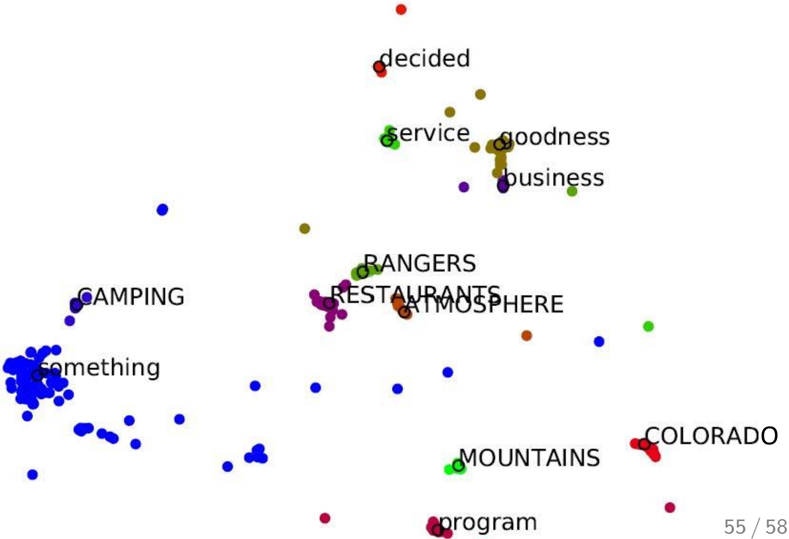
$$
\begin{aligned}
l_0(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{c}^-)\} \\
l_1(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{c}^-, \mathbf{c})\} \\
l_2(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}^-, \mathbf{c})\} \\
l_3(\mathbf{x}, \mathbf{c}) &= max\{0, m + d_{\cos}(\mathbf{x}, \mathbf{c}) - d_{\cos}(\mathbf{x}, \mathbf{x}^-)\}
\end{aligned}
$$

Variants:

- Weighted combination of these losses
- Cost-sensitive margin that scales with orthographic distance

TTIC

Visualization of acoustic + acoustically grounded word embeddings



Whole-word speech recognition with joint acoustic/written word embeddings

Whole-word ASR [Audhkhasi+ 2017]

- Output labels are whole words
- Final layer weights represent a word embedding matrix
- Many rare words $\Longrightarrow$ many rows are learned very poorly
- **Idea:** Pre-train the RNN with acoustic word embeddings and the softmax layer weights with jointly trained written word embeddings
- This improves speech recognition performance [Settle+ 2019, Collobert+ 2019]
- **Bonus**: Can recognize previously unseen words

Other exciting current speech research...

- Unsupervised representation learning (is starting to work!)
- Learning speech representations from multimodal data
- Learning **semantic** speech representations from visually grounded data
- Multilingual models
- End-to-end spoken language understanding
- Speech generation

THE END