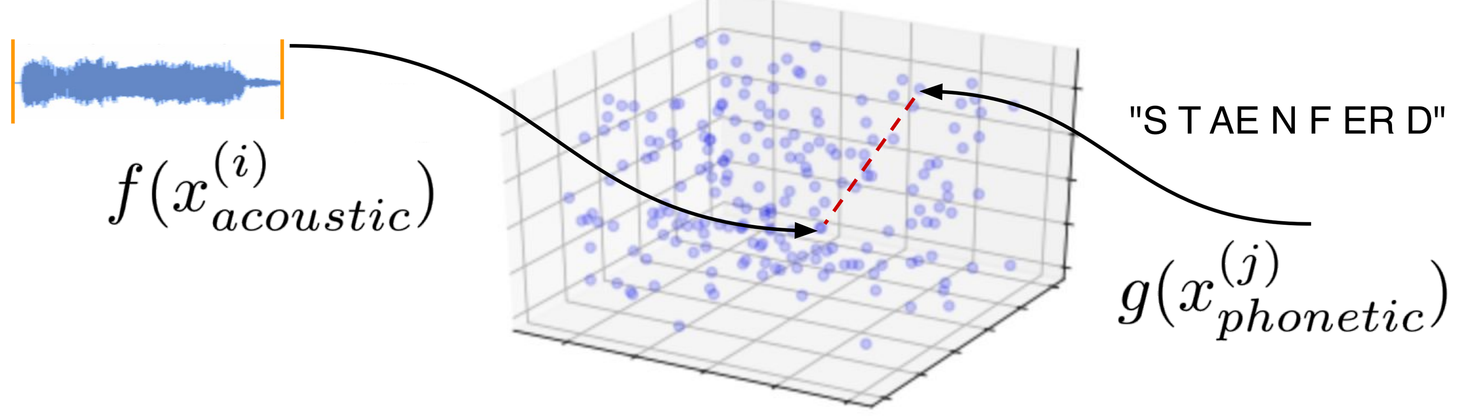
Learning Joint Acoustic-Phonetic Word Embeddings

Mohamed G. Mahmoud (elgeish@stanford.edu)

https://youtu.be/HmDT3NE3dck

OVERVIEW

We learned **encoders** of variable-length, **acoustic or phonetic**, sequences that represent words into fixed-dimensional vectors in a **shared latent space**; such that the distance between two word vectors represents **how closely the two words sound**.



Embeddings are used in a plethora of **downstream tasks**; in speech recognition, they're used in KWS, ASR, and search. Using **weakly supervised binary classification of same- and different- word pairs** as a surrogate task, we learned encoders whose last layers in a deep **Siamese CNN** produce word embeddings — achieving an F_1 score of **0.95** on the test set.

DATA

The raw data consist of 25k short, mono, 16kHz recordings and **transcripts by non-experts**, which are then force-aligned using ASR hypotheses into words to form **654,224** word pairs. We mined for **hard negative examples** in ASR hypotheses, then we synthesized more examples given ones that share the same true label, and models **self-labeled** even more examples.

REPRESENTATION

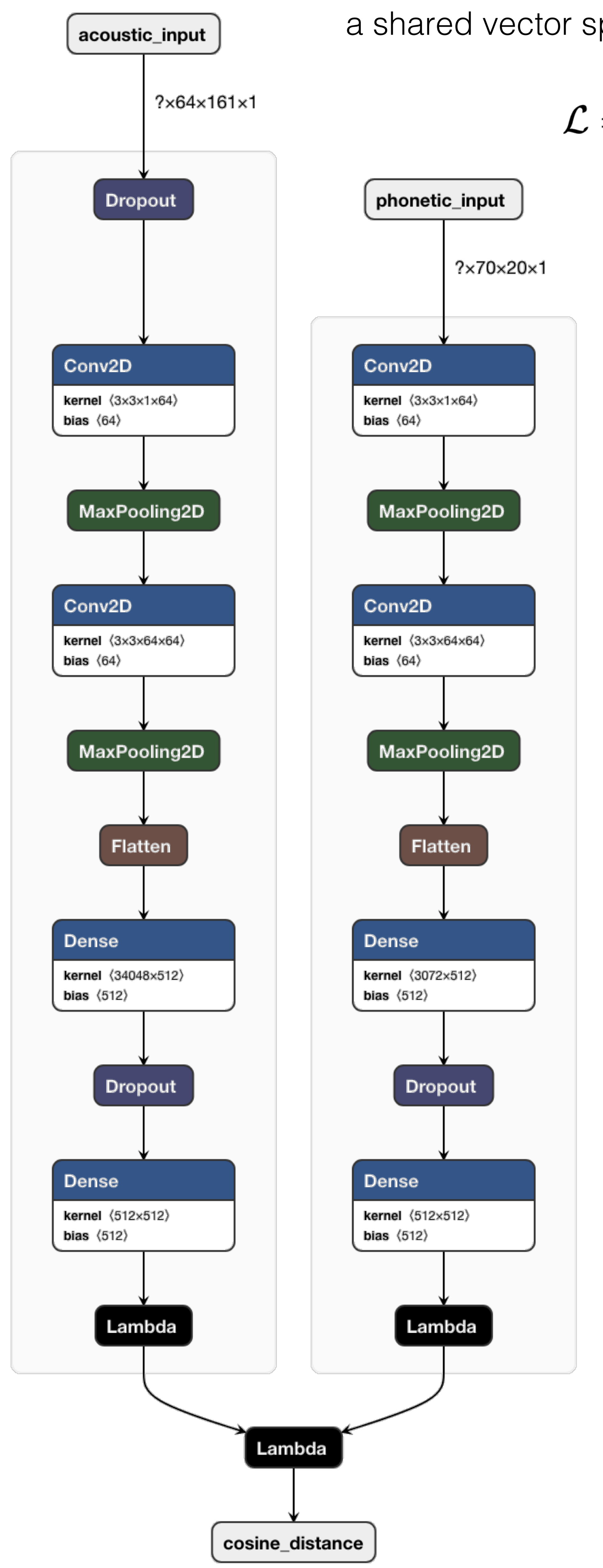
Audio: 64-band mel-spectrograms; FFT = 25ms; hop = 12ms; centered & padded to fit in a 2s window; CMVN and sphering.
Phones: one-hot encoded matrix; sentinel value for padding.

MODELS

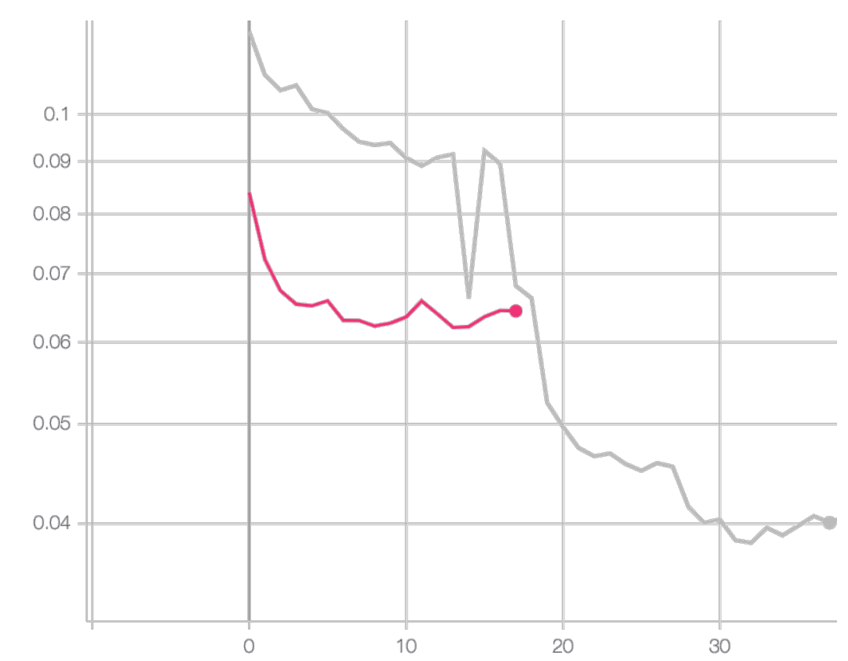
Our models minimize a **contrastive loss** to bring together similar words and separate dissimilar ones in a shared vector space; our best-performing model minimized the following loss:

$$\mathcal{L} = \frac{1}{N}\sum_{i}^{N}\left[(1-y^{(i)})\,(\mathcal{D}^{(i)})^2 + y^{(i)}\,\max(0,\,m-\mathcal{D}^{(i)})^2\right]$$

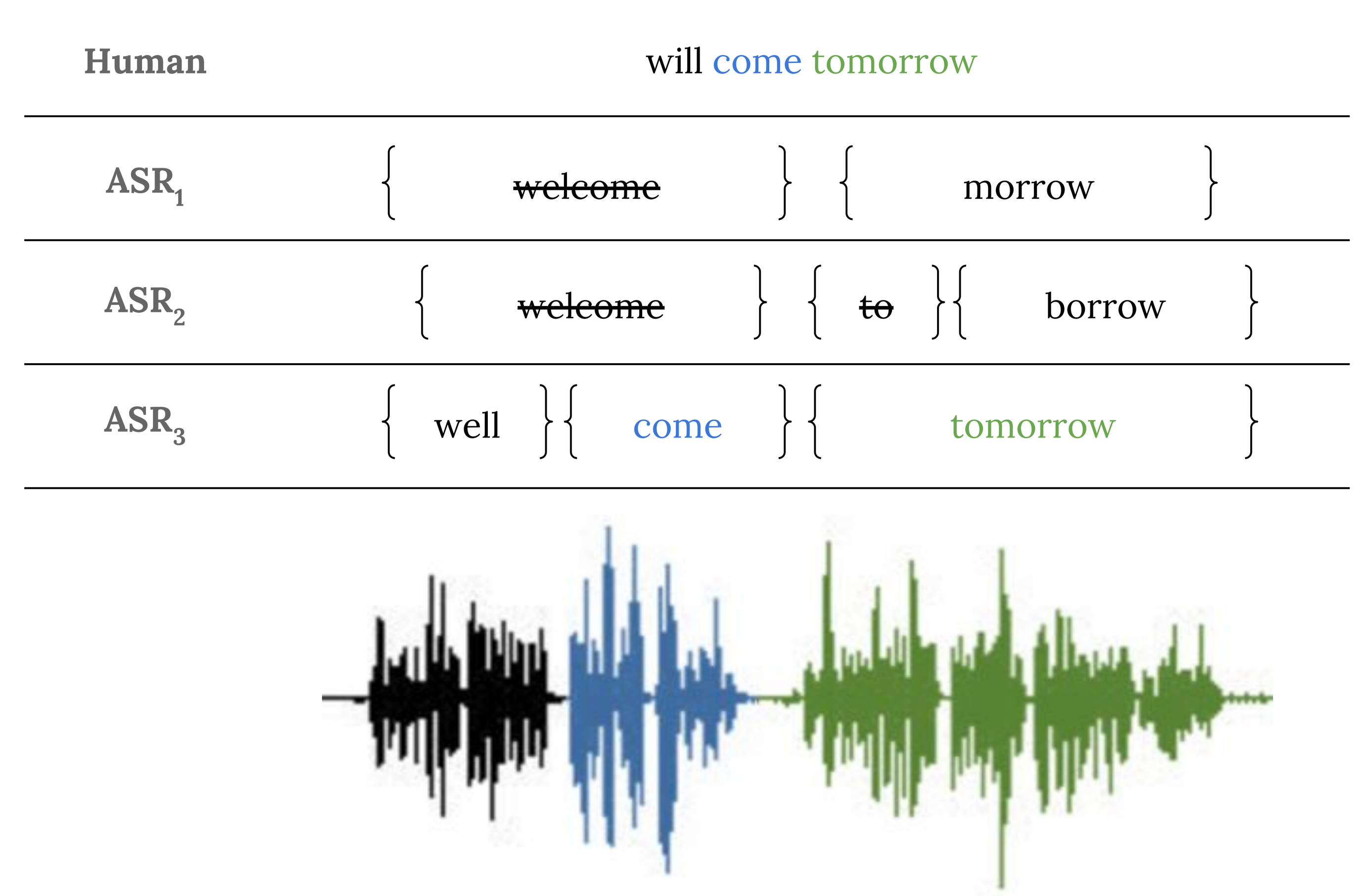
Our **Siamese NNs** feed forward **acoustic and phonetic** inputs to encode words into ℓ^2-normalized vectors & compute the distance $\mathcal{D}$ between words in each pair. Dissimilar pairs can contribute to the loss function only when $\mathcal{D}$ < margin ($m > 0$).



Our best model's architecture



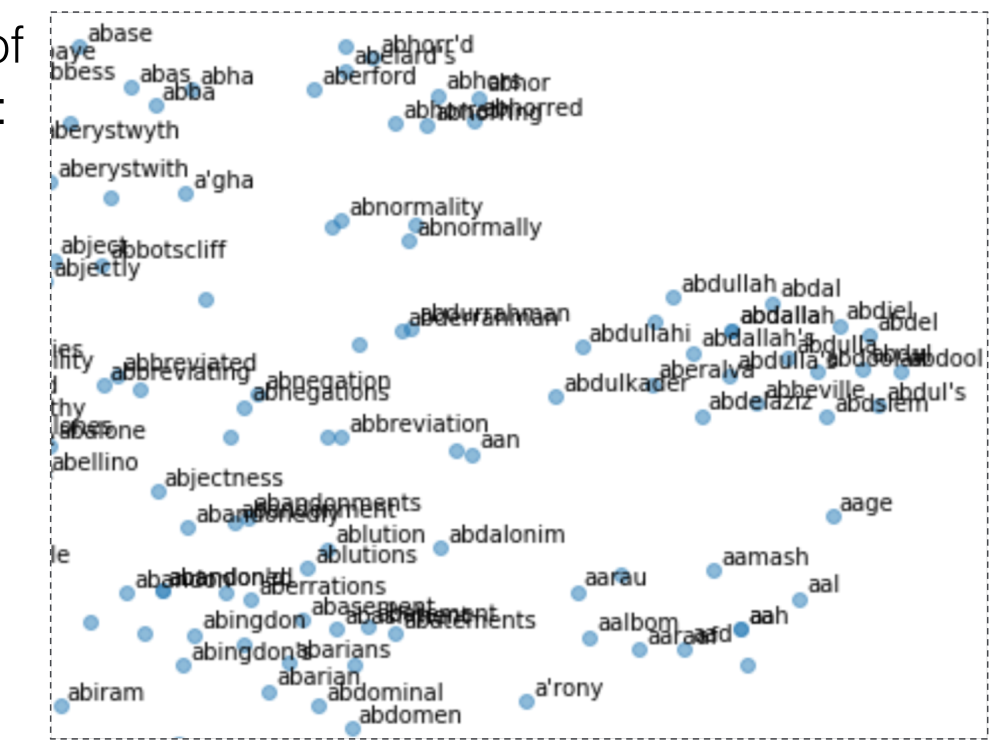
Development set loss minimization used to **stagnate** early (in red); **self-labeling** hard negative examples improved learning overall (in gray).



Mining hard negative examples from ASRs' hypotheses

DISCUSSION

A t-SNE projection of a sample of vectors:

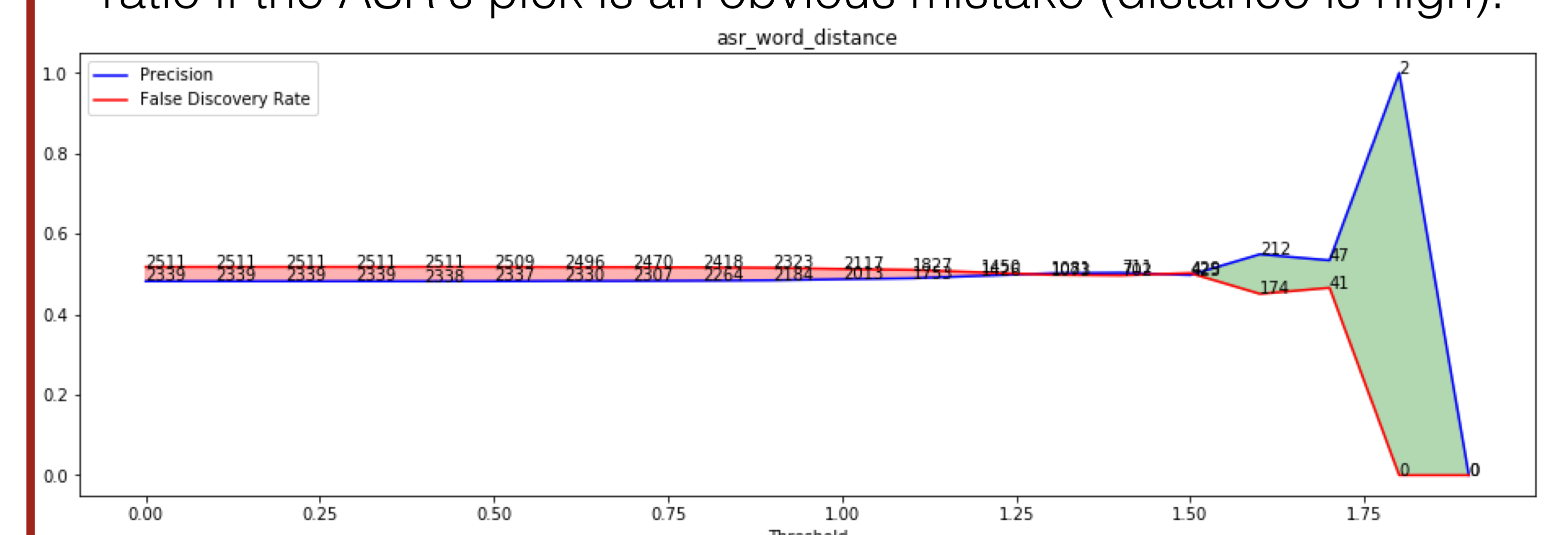


Interesting phonetic analogies:

"cat" to "cool" ≈ "pat" to "pool"

"book" to "took" ≈ "bar" to "tahr"

Reranking ASR hypotheses by testing top result from ASR vs. ours given an audio segment; we get higher precision-to-FDR ratio if the ASR's pick is an obvious mistake (distance is high).



RESULTS

Summary of notable experiments and their results for training (622k+ examples) and testing (19k examples), respectively:

#	Notable Experiment Details	F_1 Scores
1	CNN (3 × 3 × 32); dense layer; 256-D embedding; batch size = 32	0.97, 0.91
2	CNN (3 × 3 × 32) -> (3 × 3 × 64); dense layer; 256-D embedding; batch size = 32	0.99, 0.93
3	Same as #2 but for a dropout with a rate of 0.5 after the first hidden layer	0.99, 0.94
4	Same as #3 but with another dropout of 0.5 after the second hidden layer	0.95, 0.91
5	Same as #3 but with margin = the phonetic-edit distance for the pair	0.96, 0.91
6	Same as #3 but with incoming weights constrained to a maximum norm of 3	0.99, 0.94
7	2 unidirectional LSTM layers with 128 hidden units; dense layer; 256-D embedding; batch size = 32; 24 epochs (in 99 hours)	0.91, 0.87
8	2 Bidirectional LSTM layers with 512 hidden units and a dropout of 0.4 in between; a dropout of 0.2 for the acoustic input; 512-D embedding; 28 epochs (in 47 hours)	0.95, 0.91
9	CNN with 2 blocks [(3 × 3 × 64) -> (2 × 2) max pooling]; two dense layers with 512 hidden units and a dropout of 0.4 in between; a dropout of 0.2 for the acoustic input; 512-D embedding; cosine distance; batch size = 128; 64 epochs (in 4.8 hours)	0.99, **0.95**
10	Same as #9 but with additional dropout of 0.4 between convolutional layers as well; trained for much longer (142 epochs in 19 hours)	0.96, 0.93