Deep convolutional acoustic word embeddings using word-pair side information

Herman Kamper\textsuperscript{1}, Weiran Wang\textsuperscript{2}, Karen Livescu\textsuperscript{2}

\textsuperscript{1}CSTR and ILCC, School of Informatics, University of Edinburgh, UK
\textsuperscript{2}Toyota Technological Institute at Chicago, USA

ICASSP 2016
Introduction

- Most speech processing systems rely on deep architectures to classify speech frames into subword units (HMM triphone states).

- Requires pronunciation dictionary for breaking words into subwords; in many cases still makes frame-level independence assumptions.

- Some studies have started to reconsider whole words as basic modelling unit [Heigold et al., 2012; Chen et al., 2015].
Segmental automatic speech recognition

Segmental conditional random field ASR [Maas et al., 2012]:

Whole-word lattice rescoring [Bengio and Heigold, 2014]:

The model was trained on 90% of the training set, for about 5 days on a single machine, using stochastic gradient descent.
Segmental query-by-example search

From [Levin et al., 2015]:

[Chen et al., 2015]: Similar scheme for “Okay Google” using LSTMs.

Fig. 1. Diagram of the S-RAILS audio search system.
Segmental query-by-example search

From [Levin et al., 2015]:

Fig. 1. Diagram of the S-RAILS audio search system.

[Chen et al., 2015]: Similar scheme for “Okay Google” using LSTMs.

In this work, we also use a query-related task for evaluation.
Acoustic word embedding problem

\[ \textbf{x}_i \in \mathbb{R}^d \text{ in } d\text{-dimensional space} \]

\[ f(\textbf{Y}_1) \]

\[ f(\textbf{Y}_2) \]
Reference vector method [Levin et al., 2013]
Reference vector method [Levin et al., 2013]

Segment we want to embed:

\[ y_{t_1:t_2} \]
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $\mathcal{Y}_{\text{ref}}$:

Embedding: in fixed $d$-dimensional space $x_i = f(y_{t_1:t_2})$
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $Y_{ref}$:

Dimensionality reduction:

Embedding: in fixed $d$-dimensional space $x_i = f(y_{t_1:t_2})$
Reference vector method [Levin et al., 2013]

Segment we want to embed:

\[ y_{t_1:t_2} \]

Reference set \( \mathcal{Y}_{\text{ref}} \):

\[ \text{Dist}_1, \text{Dist}_2, \ldots, \text{Dist}_m \in \mathbb{R}^m \]

Dimensionality reduction:

\[ \mathbb{P} \in \mathbb{R}^m \times d \]

Embedding:
in fixed \( d \)-dimensional space

\[ x_i = f(y_{t_1:t_2}) \]
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $\mathcal{Y}_{\text{ref}}$:

Dist_1

Dist_2

Dist_3

Dist_4

\ldots

Dist_m

Embedding: in fixed $d$-dimensional space

\[ x_i = f(y_{t_1:t_2}) \]
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $\mathcal{Y}_{\text{ref}}$:

Dist 1
Dist 2
Dist 3
Dist 4
\vdots
Dist $m$

$\in \mathbb{R}^m$

Dimensionality reduction:

Embedding: in fixed $d$-dimensional space

$x_i = f(y_{t_1:t_2})$
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $\mathcal{Y}_{\text{ref}}$:

$\text{Dist}_1$

$\text{Dist}_2$

$\text{Dist}_3$

$\text{Dist}_4$

$\text{Dist}_m$

Dimensionality reduction:

$\mathbf{P} \in \mathbb{R}^{m \times d}$

Embedding:
in fixed $d$-dimensional space

$\mathbf{x}_i = f(y_{t_1:t_2})$
Reference vector method [Levin et al., 2013]

Segment we want to embed: $y_{t_1:t_2}$

Reference set $\mathcal{Y}_{\text{ref}}$:

- $\text{Dist}_1$
- $\text{Dist}_2$
- $\text{Dist}_3$
- $\text{Dist}_4$
- $\cdots$
- $\text{Dist}_m$

$\mathcal{Y}_{\text{ref}} \in \mathbb{R}^m \times P$

$P \in \mathbb{R}^{m \times d}$

Dimensionality reduction:

$\times P$

Embedding:

$x_i = f(y_{t_1:t_2})$

$\mathbb{R}^d$

in fixed $d$-dimensional space
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]

\[
\begin{bmatrix}
w_i \\
0 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0
\end{bmatrix}
\]

\[
x_i = f(Y_i)
\]
Word classification CNN [Bengio and Heigold, 2014]

$$\text{softmax} \begin{bmatrix} w_i \\ 0 \ 0 \ 0 \ \cdots \ 1 \ \cdots \ 0 \ 0 \end{bmatrix}$$

$$\mathbf{Y}_i = f(\mathbf{Y}_i)$$
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]

\[ w_i \]

\[
\begin{bmatrix}
0 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0
\end{bmatrix}
\]

\[ Y_i \]
Word classification CNN [Bengio and Heigold, 2014]

\[
\begin{bmatrix}
0 & 0 & 0 & \cdots & 1 & \cdots & 0 & 0
\end{bmatrix}
\]

\[
x_i = f(Y_i)
\]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]
Word classification CNN [Bengio and Heigold, 2014]

\[
x_i = f(Y_i)
\]
Supervision and side information

- The word classifier CNN assumes a corpus of labelled word segments.

- In some cases these might not be available.

- Weaker form of supervision we sometimes have (e.g. [Thiollière et al., 2015]) are known word pairs: \( S_{\text{train}} = \{(m, n) : (Y_m, Y_n) \text{ are of the same type}\} \)

- Also aligns with query / word discrimination task: does two speech segments contain instances of the same word? (Don’t care about word identity.)
The word classifier CNN assumes a corpus of labelled word segments.

In some cases these might not be available.

Weaker form of supervision we sometimes have (e.g. [Thiollière et al., 2015]) are known word pairs: $S_{\text{train}} = \{(m, n) : (Y_m, Y_n) \text{ are of the same type}\}$

Also aligns with query / word discrimination task: does two speech segments contain instances of the same word? (Don’t care about word identity.)

Can we use this weak supervision (sometimes called side information) to train an acoustic word embedding function $f$?
Word similarity Siamese CNN

Use idea of *Siamese networks* [Bromley *et al.*, 1993].
Use idea of *Siamese networks* [Bromley et al., 1993].
Word similarity Siamese CNN

Use idea of Siamese networks [Bromley et al., 1993].
Loss functions

The cosine loss [Synnaeve et al., 2014]:

\[ l_{\cos} = 1 - \cos(x_1, x_2) \]

if same

\[ l_{\cos} = \cos^2(x_1, x_2) \]

if different

Margin-based hinge loss [Mikolov, 2013]:

\[ l_{\cos \text{ hinge}} = \max\{0, m + d_{\cos}(x_1, x_2) - d_{\cos}(x_1, x_3)\} \]

where \[ d_{\cos}(x_1, x_2) = 1 - \cos(x_1, x_2) \]
is the cosine distance between \( x_1 \) and \( x_2 \), and \( m \) is a margin parameter. Pair \((x_1, x_2)\) are same, \((x_1, x_3)\) are different.
Loss functions

The coscos\(^2\) loss [Synnaeve et al., 2014]:

\[
 l_{\text{cos cos}^2}(x_1, x_2) = \begin{cases} 
 \frac{1 - \cos(x_1, x_2)}{2} & \text{if same} \\
 \cos^2(x_1, x_2) & \text{if different}
\end{cases}
\]

same
different
Loss functions

The coscos² loss [Synnaeve et al., 2014]:

\[
    l_{\text{cos cos}^2}(x_1, x_2) = \begin{cases} 
    \frac{1 - \cos(x_1, x_2)}{2} & \text{if same} \\
    \cos^2(x_1, x_2) & \text{if different}
    \end{cases}
\]

Margin-based hinge loss [Mikolov, 2013]:

\[
    l_{\text{cos hinge}} = \max \{0, m + d_{\text{cos}}(x_1, x_2) - d_{\text{cos}}(x_1, x_3)\}
\]

where \(d_{\text{cos}}(x_1, x_2) = \frac{1 - \cos(x_1, x_2)}{2}\) is the cosine distance between \(x_1\) and \(x_2\), and \(m\) is a margin parameter. Pair \((x_1, x_2)\) are same, \((x_1, x_3)\) are different.
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

Cosine distance:

Treat as query

Treat as terms to search

✓

×

✓

×

✓
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

Cosine distance:

Treat as query

Treat as terms to search

\[ d_i < \text{threshold?} \]

predict:

\[ \text{different} \] ✓

\[ \text{same} \] ×
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

Cosine distance:

<table>
<thead>
<tr>
<th>d_i</th>
<th>&lt; threshold?</th>
<th>predict</th>
</tr>
</thead>
<tbody>
<tr>
<td>d_1</td>
<td>different</td>
<td>✓</td>
</tr>
<tr>
<td>d_2</td>
<td>same</td>
<td>×</td>
</tr>
</tbody>
</table>
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

<table>
<thead>
<tr>
<th>Cosine distance:</th>
<th>$d_i &lt; \text{threshold?}$</th>
<th>predict:</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$d_1$</td>
<td>different ✓</td>
</tr>
<tr>
<td></td>
<td>$d_2$</td>
<td>same ×</td>
</tr>
<tr>
<td></td>
<td>$d_3$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$d_4$</td>
<td></td>
</tr>
<tr>
<td></td>
<td>$d_N$</td>
<td></td>
</tr>
</tbody>
</table>

Treat as query

Treat as terms to search
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

Cosine distance:

\[
d_1 \quad \checkmark
\]

\[
d_2 \quad \times
\]

\[
d_3 \quad \checkmark
\]

\[
d_4 \quad \checkmark
\]

\[
d_N \quad \checkmark
\]

\[
d_i < \text{threshold?}
\]

\[
predict:
\]

\[
different \quad \checkmark
\]

\[
same \quad \times
\]

\[
same
\]
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].

<table>
<thead>
<tr>
<th>Cosine distance:</th>
<th>$d_i &lt; \text{threshold?}$</th>
<th>predict:</th>
</tr>
</thead>
<tbody>
<tr>
<td>$d_1$</td>
<td>different</td>
<td>✓</td>
</tr>
<tr>
<td>$d_2$</td>
<td>same</td>
<td>×</td>
</tr>
<tr>
<td>$d_3$</td>
<td>same</td>
<td>✓</td>
</tr>
</tbody>
</table>
Embedding evaluation: the same-different task

Proposed in [Carlin et al., 2011] and also used in [Levin et al., 2013].
Experimental setup

- Speech from Switchboard is used for evaluation.
- Training set: 10k word tokens; sampled 100k training word pairs.
- Test set for same-different evaluation: 11k word tokens, 60.7M pairs, 3% produced by same speaker.
- Used a comparable development set.
Network architectures: Word classifier CNN

- 39-dimensional padded MFCCs, $n_{pad} = 200$
  - 1-D convolution: 96 ReLU filters over 9 frames
  - Max-pooling: 3 units
  - 1-D convolution: 96 ReLU filters over 8 units
  - Max-pooling: 3 units
  - 1024 ReLU
  - Linear Bottleneck (optional)
  - Softmax: 1061 classes
Network architectures: Word classifier CNN

39-dimensional padded MFCCs, \( n_{pad} = 200 \)

1-D convolution: 96 ReLU filters over 9 frames
Max-pooling: 3 units

1-D convolution: 96 ReLU filters over 8 units
Max-pooling: 3 units

1024 ReLU

Linear Bottleneck (optional)

softmax: 1061 classes
Network architectures: Word classifier CNN

39-dimensional padded MFCCs,
\( n_{\text{pad}} = 200 \)

1-D convolution: 96 ReLU filters over 9 frames
Max-pooling: 3 units

1-D convolution: 96 ReLU filters over 8 units
Max-pooling: 3 units

1024 ReLU
Linear Bottleneck (optional)
softmax: 1061 classes
Network architectures: Siamese CNN

\[ l(x_1, x_2) \]

- 39-dimensional padded MFCCs, \( n_{\text{pad}} = 200 \)
- 1-D convolution: 96 ReLU filters over 9 frames
- Max-pooling: 3 units
- 1-D convolution: 96 ReLU filters over 8 units
- Max-pooling: 3 units
- 2048 ReLU
- 1024 Linear

Distance calculation
Network architectures: Siamese CNN

Distance: $d(x_1, x_2)$

1-D convolution: 96 ReLU filters over 9 frames
Max-pooling: 3 units
1-D convolution: 96 ReLU filters over 8 units
Max-pooling: 3 units
2048 ReLU
1024 Linear

39-dimensional padded MFCCs, $n_{pad} = 200$
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td>Acoustic word embed.</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
<tr>
<td>Reference vector approach</td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td>LDA on: lcos hinge loss</td>
<td>1024</td>
<td>0.504 ± 0.011</td>
</tr>
<tr>
<td>Siamese CNN, lcos hinge loss</td>
<td>1024</td>
<td>0.545 ± 0.011</td>
</tr>
<tr>
<td>Siamese CNN, lcos hinge loss</td>
<td>1024</td>
<td>0.545 ± 0.011</td>
</tr>
</tbody>
</table>
### Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
</tbody>
</table>

Correspondence autoencoder [Kamper et al., 2015] 100 0.469

Acoustic word embed. Reference vector approach [Levin et al., 2013] 50 0.365

Word classifier CNN 1061 0.532 ± 0.014

Siamese CNN, l_{cos cos} loss 1024 0.342 ± 0.026

Siamese CNN, l_{cos hinge} loss 1024 0.549 ± 0.011

LDA on: l_{cos hinge}, d = 1024 100 0.545 ± 0.011
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
</tbody>
</table>
### Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td>Reference vector approach [Levin et al., 2013]</td>
<td>50</td>
<td>0.365</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td>Reference vector approach [Levin et al., 2013]</td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td>Word classifier CNN</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
</tbody>
</table>

Acoustic word embed.
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DTW</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td>Reference vector approach [Levin et al., 2013]</td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td><strong>Acoustic word embed.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word classifier CNN</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.474 ± 0.012</td>
</tr>
</tbody>
</table>

---
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper <em>et al.</em>, 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td>Reference vector approach [Levin <em>et al.</em>, 2013]</td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td>Word classifier CNN</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.474 ± 0.012</td>
</tr>
<tr>
<td>Siamese CNN, $l_{\cos\cos^2}$ loss</td>
<td>1024</td>
<td>0.342 ± 0.026</td>
</tr>
<tr>
<td>Siamese CNN, $l_{\cos\text{hinge}}$ loss</td>
<td>1024</td>
<td><strong>0.549 ± 0.011</strong></td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DTW</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td><strong>Acoustic word embed.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reference vector approach [Levin et al., 2013]</td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td>Word classifier CNN</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.474 ± 0.012</td>
</tr>
<tr>
<td>Siamese CNN, $l_{\text{cos cos}^2}$ loss</td>
<td>1024</td>
<td>0.342 ± 0.026</td>
</tr>
<tr>
<td>Siamese CNN, $l_{\text{cos hinge}}$ loss</td>
<td>1024</td>
<td><strong>0.549 ± 0.011</strong></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.504 ± 0.011</td>
</tr>
</tbody>
</table>
## Results

<table>
<thead>
<tr>
<th>Representation</th>
<th>Dim</th>
<th>AP</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>DTW</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>MFCCs with CMVN</td>
<td>39</td>
<td>0.214</td>
</tr>
<tr>
<td>Correspondence autoencoder [Kamper et al., 2015]</td>
<td>100</td>
<td>0.469</td>
</tr>
<tr>
<td><strong>Reference vector approach [Levin et al., 2013]</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.365</td>
</tr>
<tr>
<td><strong>Acoustic word embed.</strong></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Word classifier CNN</td>
<td>1061</td>
<td>0.532 ± 0.014</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.474 ± 0.012</td>
</tr>
<tr>
<td>Siamese CNN, $\ell_{\cos \cos^2}$ loss</td>
<td>1024</td>
<td>0.342 ± 0.026</td>
</tr>
<tr>
<td>Siamese CNN, $\ell_{\cos \text{hinge}}$ loss</td>
<td>1024</td>
<td>0.549 ± 0.011</td>
</tr>
<tr>
<td></td>
<td>50</td>
<td>0.504 ± 0.011</td>
</tr>
<tr>
<td>LDA on: $\ell_{\cos \text{hinge}}, d = 1024$</td>
<td>100</td>
<td>0.545 ± 0.011</td>
</tr>
</tbody>
</table>
Varying dimensionalities on development data

Average precision (AP)

Word classifier CNN

Siamese CNN: $l_{\cos \cos^2}$

Siamese CNN: $l_{\cos \text{hinge}}$

Dimensionality of acoustic embedding (log scale)
Varying dimensionalities on development data

![Graph showing the relationship between average precision (AP) and dimensionality of acoustic embedding for different models.

- **Word classifier CNN**
- **Siamese CNN: \(l_{\cos \cos^2}\)**
- **Siamese CNN: \(l_{\cos \text{hinge}}\)**
- **Siamese CNN with LDA**

The graph plots the average precision (AP) against the dimensionality of the acoustic embedding, on a log scale. The y-axis represents the average precision, ranging from 0.05 to 0.50, while the x-axis represents the dimensionality of the acoustic embedding, ranging from 10 to 3000.
Summary and conclusion

▶ Introduced the Siamese CNN for obtaining acoustic word embeddings, and evaluated different cost functions.

▶ Evaluated using word discrimination task, and showed similar performance to word classifier CNN.

▶ For smaller dimensionalities: Siamese CNN outperformed classifier CNN.

▶ Self-criticism: evaluated on a small dataset (low-resource setting).

▶ Future work: sequence models, using embeddings for search and ASR.
Code

Neural networks (Theano): https://github.com/kamperh/couscous

Complete recipe: https://github.com/kamperh/recipe_swbd_wordembeds