End-to-End DNN based speaker recognition inspired by i-vector and PLDA

ICASSP 2018, Calgary

Johan Rohdin, Anna Silnova, Mireia Diez, Oldrich Plchot, Pavel Matejka, Lukas Burget
Background

- i-vectors and PLDA have been the state-of-the-art for many years

  - Parts of i-vector+PLDA systems have been replaced by NNs
    - MFCCs → bottleneck features\(^1\), UBM → DNN acoustic models\(^2\)
      - PLDA → DBNs\(^3\), UBM and T-matrix → single NN\(^4,5\)

- End-to-end systems replace the whole system by one NN
  - Successful for short utterances\(^6,7\) but less successful for long\(^7\)
  - Usually trained on short utterances
  - Training on long utterances may overfit and requires large memory

This work

- Develop an end-to-end system that is initialized to mimic an i-vector + PLDA system, then refined with end-to-end training

1. First develop the individual blocks:
   - Feature to stats ($f2s$) NN: Collection of sufficient statistics
   - Stats to ivector ($i2s$) NN: i-vector calculation
   - DPLDA: Scoring

2. Plug the blocks together and optimize them jointly for the speaker verification task, i.e., with end-to-end training on long and short utterances

- To find good architecture and initialization for end-to-end training
- Avoid overfitting by regularizing towards initial model
- Good performance on long and short multi language conditions
Data and baselines

- Training data based on PRISM dataset
  - SRE 04-10, Fisher, Switchboard
  - UBM, iXtractor uses all training data
  - PLDA and DPLDA use only telephone data but use also short cuts created from non-English and non-native-English data
- Testing on language PRISM condition and SRE16 single enroll
  - We also cut PRISM lang into short segments to mimic SRE16
- All of our features are standard MFCCs+$\Delta+\Delta\Delta$ (60 dimensions)
- Baselines are generative and discriminative PLDA based on 600-dimensional i-vectors extracted with 2048-component diagonal-covariance UBM
Features to sufficient statistics (f2s)

- Train NN to predict UBM responsibilities
- Input: processed and expanded features
  - Dimensionality: 360
  - Context: 30 Frames
- Output: GMM responsibilities
- Training objective: Categorical cross-entropy (soft targets)
- Given features and responsibilities, calculate sufficient statistics
Developments on SRE10, core-core condition 5

<table>
<thead>
<tr>
<th>Model</th>
<th>EER [%]</th>
<th>mindcf0.01</th>
<th>mindcf0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline (GMM)</td>
<td>2.37</td>
<td>0.245</td>
<td>0.294</td>
</tr>
<tr>
<td>NN (60_1500_1500_2048)</td>
<td>2.27</td>
<td>0.242</td>
<td>0.293</td>
</tr>
<tr>
<td>NN (360_1500_1500_2048)</td>
<td>2.20</td>
<td>0.231</td>
<td>0.278</td>
</tr>
<tr>
<td>NN (360_1500_1500_1500_1500_2048)</td>
<td>2.17</td>
<td>0.228</td>
<td>0.279</td>
</tr>
</tbody>
</table>

- Larger context results in better predictions of the responsibilities
  - Probably because of increased robustness to unseen test conditions
Sufficient statistics to i-vectors (s2i)

- Model for initializing e2e system is trained on the output from f2s
- Input preprocessing
  a. Calculate relevance MAP adapted supervector (r=16)
  b. Reduce it by PCA from 2048 x 60 = 122880 to 4000 dim.
- 2 hidden layers with 600 units, tanh activation functions followed by affine transform and “length-norm”
- Output: LDA reduced and length-normalized i-vectors
- Training objective: Cosine distance
### S2i architecture

Developments on SRE10, core-core condition 5
- PCA dimension: 4000 (higher was not better), NN (4000_600_600_600)
- Mean square error objective

<table>
<thead>
<tr>
<th>Target ivectors</th>
<th>NN Output</th>
<th>EER [%]</th>
<th>mdcf0.01</th>
<th>mdcf0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length norm</td>
<td>Affine</td>
<td>2.86</td>
<td>0.290</td>
<td>0.346</td>
</tr>
<tr>
<td>WC norm. + Length norm</td>
<td>Affine</td>
<td>2.76</td>
<td>0.276</td>
<td>0.321</td>
</tr>
<tr>
<td>WC norm. + Length norm</td>
<td>Affine + Length norm.</td>
<td>2.59</td>
<td>0.270</td>
<td>0.313</td>
</tr>
</tbody>
</table>

- Cosine distance objective

<table>
<thead>
<tr>
<th>WC norm. + Length norm</th>
<th>Linear -&gt; Length norm</th>
<th>EER [%]</th>
<th>mdcf0.01</th>
<th>mdcf0.005</th>
</tr>
</thead>
<tbody>
<tr>
<td>+ LDA</td>
<td>Affine + Length norm.</td>
<td>2.55</td>
<td>0.257</td>
<td>0.310</td>
</tr>
<tr>
<td>+ L1 reg</td>
<td>Affine + Length norm.</td>
<td>2.43</td>
<td>0.256</td>
<td>0.306</td>
</tr>
</tbody>
</table>
The DPLDA baseline is trained iteratively using full batches (L-BFGS).

For joint training with other blocks we use minibatches.

Minibatch approach in experiments:

- Group all utterances into pairs of the same speakers
- Shuffle the pairs
- Select $N$ pairs (without replacement) to form a minibatch

Training objective: Binary cross-entropy for all trials in the batch.
Effect on target trials

Alternative method
All utterances of the same speaker in one batch

Used method
Generally 2 utterances per speaker in each batch

- Total weight of each speaker may change for the used method (and sets if their average number of utterances per speaker differs)
- In DPLDA experiments the alternative method did not work well
Memory issues in end-to-end system

- **f2s** processes frames. Number of intermediate values needed in training:
  \[ \text{#Frames} \times (360+1500+1500+1500+1500+2048) \]
- When **f2s** is trained independently, one frame from a many different utterances can be used.
- For **e2e** we need many full utterances per batch so the number of frames is large.
- We discard intermediate values from forward prop. of **f2s** and recalculate them during backprop. (Similar to Theano’s `scan_checkpoints`)
- With this trick we can use around ~30 utterances per minibatch instead of ~5 on a GPU with 4GB.
- The parameters (mainly the PCA matrix) of the network itself uses about 3GB.
## Results

Average of minDCF0.01 and minDCF0.005

<table>
<thead>
<tr>
<th>System</th>
<th>UBM</th>
<th>i-extractor</th>
<th>PLDA</th>
<th>SRE16</th>
<th>PRISM Short</th>
<th>PRISM Long</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline</td>
<td>GMM</td>
<td>T</td>
<td>Gen.</td>
<td>0.988</td>
<td>0.699</td>
<td>0.411</td>
</tr>
<tr>
<td>Baseline DPLDA</td>
<td>GMM</td>
<td>T</td>
<td>Discr.</td>
<td>0.975</td>
<td>0.616</td>
<td>0.360</td>
</tr>
<tr>
<td>f2s</td>
<td>NN</td>
<td>T</td>
<td>Gen.</td>
<td>0.980</td>
<td>0.687</td>
<td>0.394</td>
</tr>
<tr>
<td>s2i</td>
<td>GMM</td>
<td>NN</td>
<td>Gen.</td>
<td>0.988</td>
<td>0.788</td>
<td>0.430</td>
</tr>
<tr>
<td>f2s+s2i</td>
<td>NN</td>
<td>NN</td>
<td>Gen</td>
<td>0.982</td>
<td>0.780</td>
<td>0.432</td>
</tr>
<tr>
<td>f2s+s2i+DPLDA</td>
<td>NN</td>
<td>NN</td>
<td>Discr.</td>
<td>0.953</td>
<td>0.597</td>
<td>0.300</td>
</tr>
<tr>
<td>s2i+DPLDA - joint N=5000</td>
<td>NN</td>
<td>NN</td>
<td>Discr.</td>
<td>0.936</td>
<td>0.586</td>
<td>0.287</td>
</tr>
<tr>
<td>All - joint, N=10</td>
<td>NN</td>
<td>NN</td>
<td>Discr.</td>
<td>0.936</td>
<td>0.587</td>
<td>0.289</td>
</tr>
</tbody>
</table>
Conclusions

- Neural networks can mimic estimation of responsibilities and i-vector extraction reasonably well.
- Fine-tuning of the initialized network with binary cross-entropy criteria improves the performance.
- Main improvement of joint training comes from refining of s2i module:
  - DPLDA module does not change much.
  - f2s module hard to train since we can use only small batches.
- Future work:
  - Better joint training of the three blocks.
  - Selection of suitable (difficult) training trials.
  - Explore different training objectives, multiple enrollment sessions.
  - Update PCA matrix and feature transform.
  - Replace f2s with lighter network.
  - Experiment with less constrained/regularized network.
Thank you!
Questions?