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

BRNO UNIVERSITY OF TECHNOLOGY

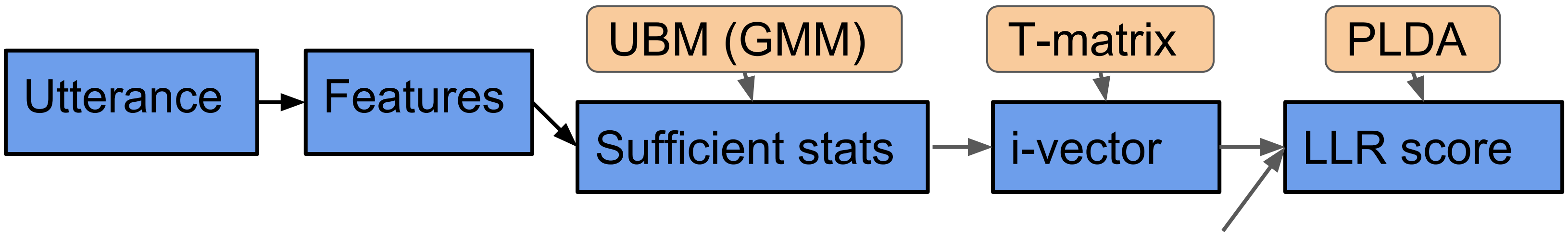
South Moravian Region

MARIE CURIE ACTIONS

18th April 2018

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- Parts of i-vector+PLDA systems have been replaced by NNs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

[1] Lozano-Diez et al. Odyssey 2016; [2] Lei et al. ICASSP 2014; [3] Ghahabi et al. ICASSP 2014; [4] Variani et al. ICASSP 2014; [5] Snyder et al. SLT 2016; [6] Heighold et al. 2016; [7] Snyder et al. SLT 2016

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

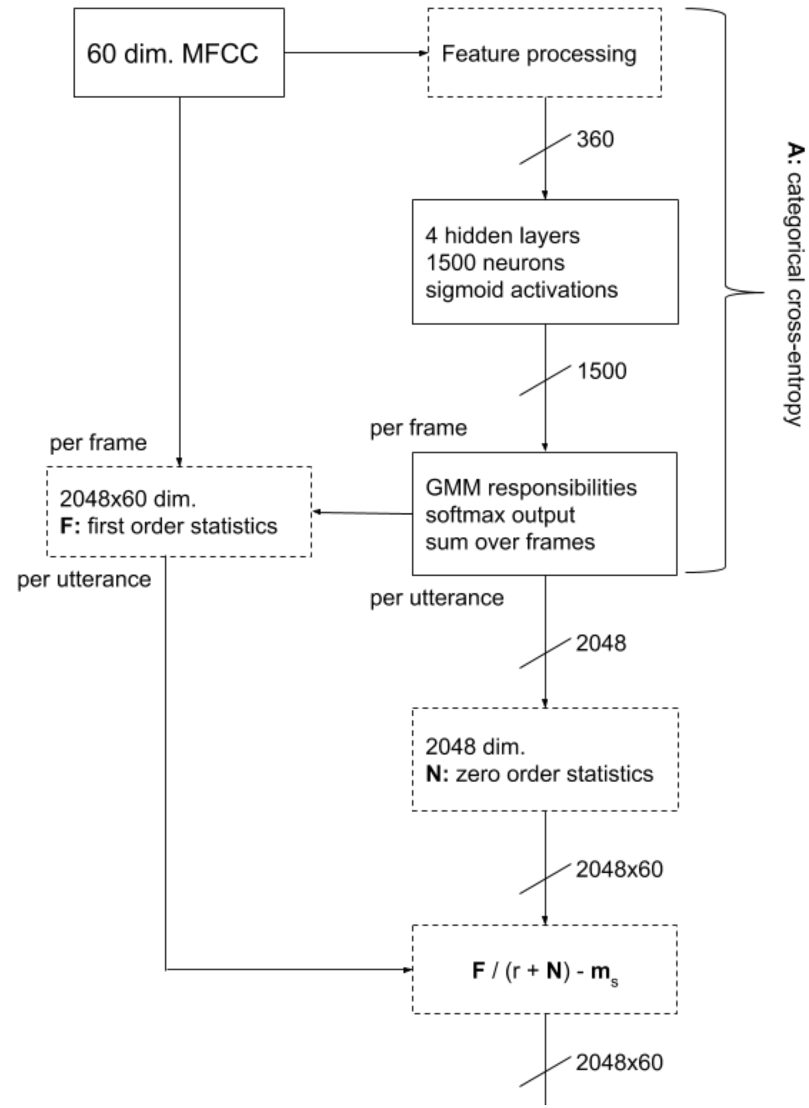
- To find good architecture and initialization for end-to-end training
- Avoid overfitting by regularizing towards initial model
- Good performance on long and short multi language conditions

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+Δ+$\Delta\Delta$ (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



f2s Architecture

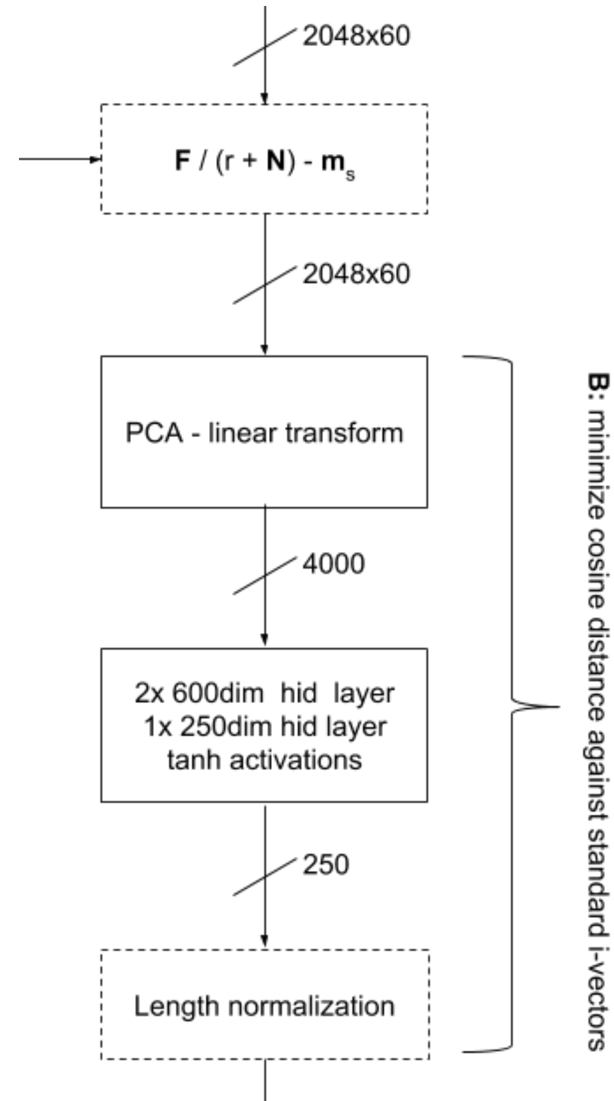
Developments on SRE10, core-core condition 5

Model	EER [%]	mindcf0.01	mindcf0.005
Baseline (GMM)	2.37	0.245	0.294
NN (60_1500_1500_2048)	2.27	0.242	0.293
NN (360_1500_1500_2048)	2.20	0.231	0.278
NN (360_1500_1500_1500_1500_2048)	**2.17**	**0.228**	**0.279**

- 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

	EER [%]	mdcf0.01	mdcf0.005
BASELINE	2.41	0.246	0.295

Developments on SRE10, core-core condition 5

- PCA dimension: 4000 (higher was not better), NN (4000_600_600_600)
- Mean square error objective

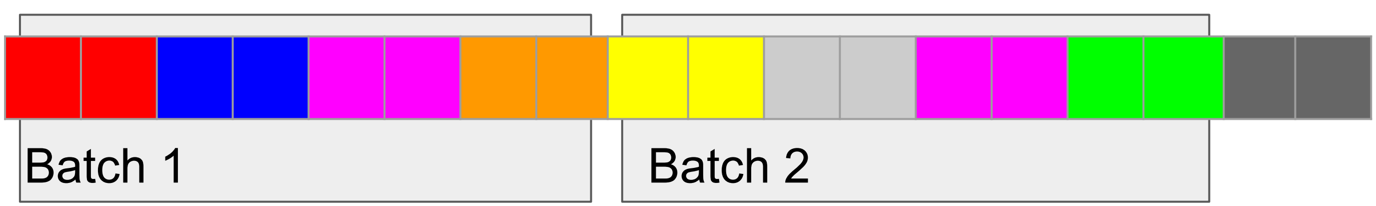
Target ivectors	NN Output	EER [%]	mdcf0.01	mdcf0.005
Length norm	Affine	2.86	0.290	0.346
WC norm. + Length norm	Affine	2.76	0.276	0.321
WC norm. + Length norm	Affine + Length norm.	2.59	0.270	0.313

- Cosine distance objective

WC norm. + Length norm	Linear -> Length norm	2.56	0.269	0.311
+ LDA	Affine + Length norm.	2.55	0.257	0.310
+ L1 reg	Affine + Length norm.	**2.43**	**0.256**	**0.306**

I-vectors to scores (DPLDA)

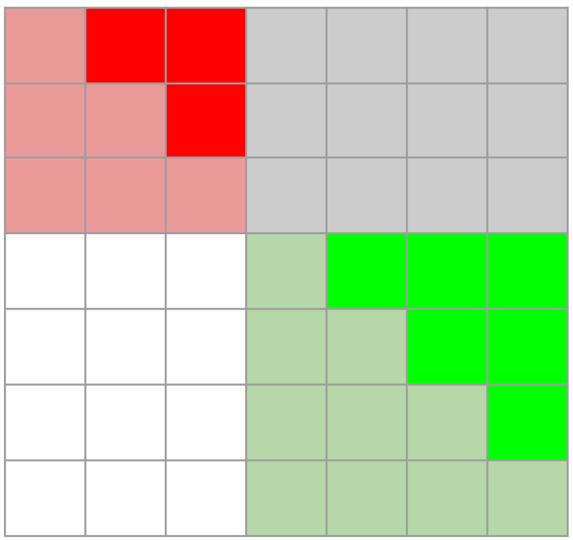
- The DPLDA baseline is trained iteratively using full batches (L-BFGS)
- For joint training with other blocks we use minibatches
- Minibatch approach in experiments:
  a. Group all utterances into pairs of the same speakers
  b. Shuffle the pairs
  c. Select *N* pairs (without replacement) to form a minibatch



- Training objective: Binary cross-entropy for all trials in the batch

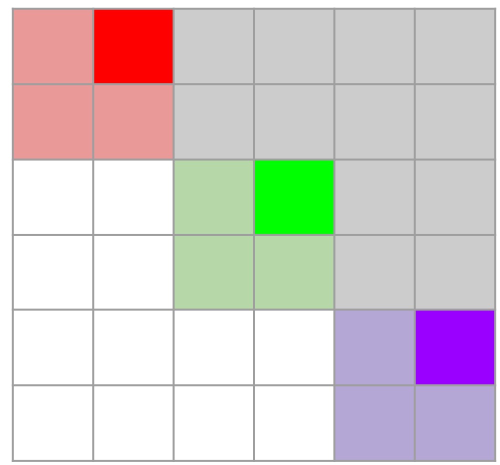
Effect on target trials

Alternative method
All utterances of the same speaker in one batch



Many but dependent

Used method
Generally 2 utterances per speaker in each batch



Fewer but less dependent

- 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: #Frames*(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 =Joint training

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

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?