Speaker embeddings by modeling channel-wise correlations

Themos Stafylakis, Johan Rohdin, Lukas Burget

Omilia
Conversational Intelligence

BRNO
UNIVERSITY
OF TECHNOLOGY

Outline

- Overview of pooling methods in speaker embeddings
- Intuition behind the proposed pooling
- Implementation details
- Experimental results on VoxCeleb
- Conclusions

Statistics pooling

- **The dominant approach to pooling:**
  - In the **statistics pooling layer** of 2D-ConvNets (e.g. ResNets) **mean & std** of all **frequency-channel** pairs are estimated.
  - The network is trained E2E to learn features having speaker discriminative **1st** & **2nd order statistics**.
- **Existing methods** for improving statistics pooling:
  - Include **higher order** stats (skewness, kurtosis).
  - Segment the space into regions and calculate "**local**" stats, (Multihead attentive pooling, Net- & Ghost-VLAD).

Statistics pooling

- **The dominant approach to pooling:**
  - In the **statistics pooling layer** of 2D-ConvNets (e.g. ResNets) **mean & std** of all **frequency-channel** pairs are estimated.
  - The network is trained E2E to learn features having speaker discriminative **1st** & **2nd order statistics**.
- **Existing methods** for improving statistics pooling:
  - Include **higher order** stats (skewness, kurtosis).
  - Segment the space into regions and calculate "**local**" stats, (Multihead attentive pooling, Net- & Ghost-VLAD).

Correlation pooling

- **Main Idea**:
  - **Don't learn features** with speaker discriminative -1st, 2nd or high order, global or local- statistics.
  - **Learn features** with speaker discriminative **pairwise correlations**.

- Consider a **parametrization** of a **Multivariate Normal Distribution**:
  - $\mu_i = \mathrm{E}\{x_i\}$ (mean)
  - $\sigma_i = \mathrm{E}\{(x_i - \mu_i)^2\}^{1/2}$ (std)
  - $\mathbf{R}_{ij} = \mathrm{E}\{(x_i - \mu_i)(x_j - \mu_j)\}/\sigma_i\sigma_j$ (corr. coeff.)

- **The proposed pooling method uses $\mathbf{R}_{ij}$ instead of μ_i and/or σ_i.**

Correlation pooling

- **Main Idea**:
  - **Don't learn features** with speaker discriminative -1st, 2nd or high order, global or local- statistics.
  - **Learn features** with speaker discriminative **pairwise correlations**.
- Consider a **parametrization** of a **Multivariate Normal Distribution**:
  - $\boldsymbol{\mu}_i = \mathrm{E}\{\boldsymbol{x}_i\}$ (mean)
  - $\boldsymbol{\sigma}_i = \mathrm{E}\{(\boldsymbol{x}_i - \boldsymbol{\mu}_i)^2\}^{1/2}$ (std)
  - $\mathbf{R}_{ij} = \mathrm{E}\{(\boldsymbol{x}_i - \boldsymbol{\mu}_i)(\boldsymbol{x}_j - \boldsymbol{\mu}_j)\}/\boldsymbol{\sigma}_i\boldsymbol{\sigma}_j$ (corr. coeff.)
- **The proposed pooling method uses $\mathbf{R}_{ij}$ instead of $\boldsymbol{\mu}_i$ and/or $\boldsymbol{\sigma}_i$.**

Style-transfer in computer vision

The proposed pooling method is inspired by style-transfer in computer vision:

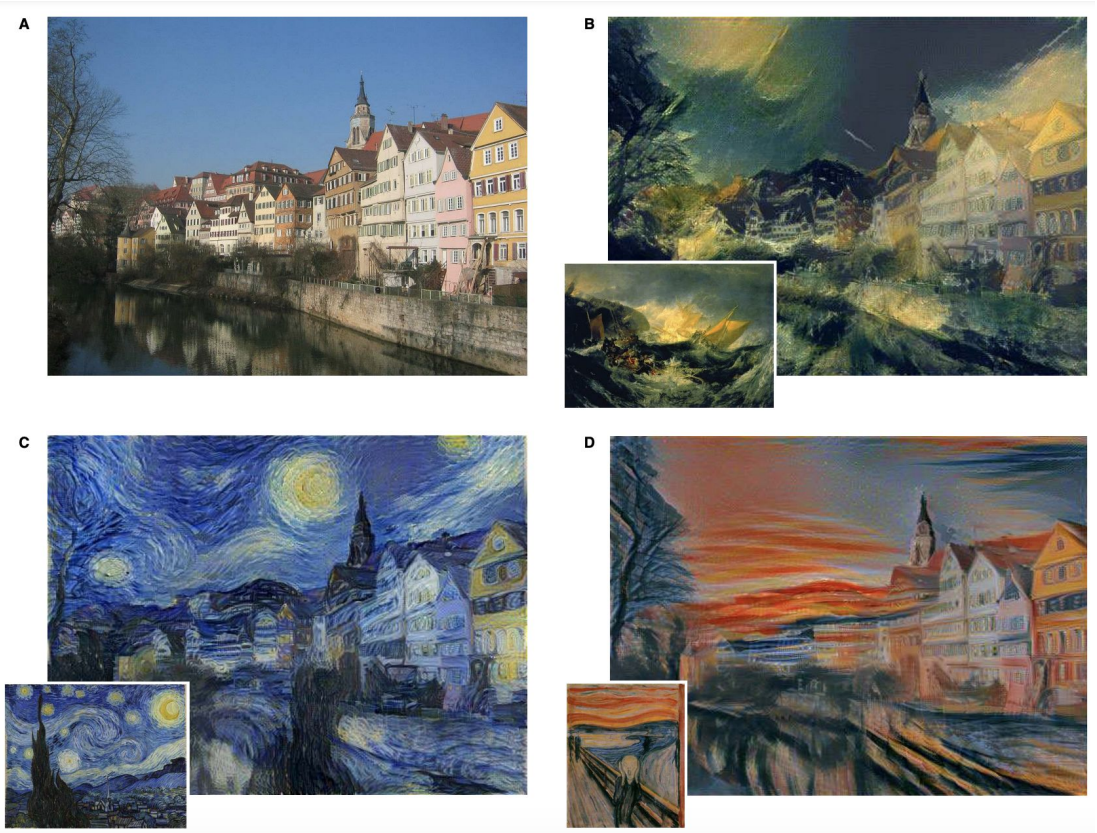


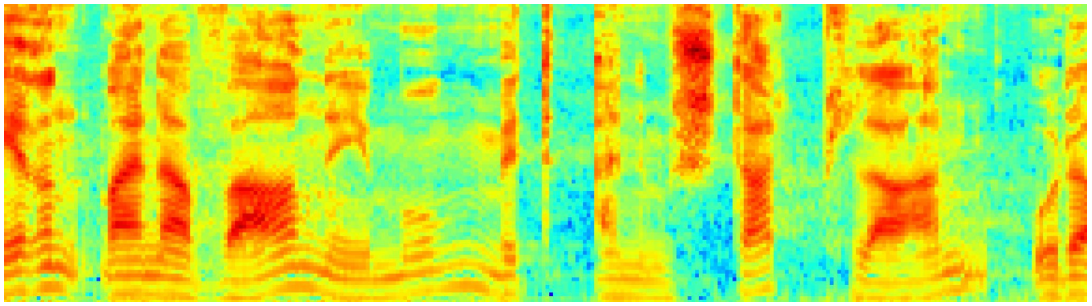
Figure from “L. A. Gatys, *et al.* Image style transfer using convolutional neural networks, CVPR 2016”

Modeling style in images & spectrograms

- The input to the ResNet is 80-dim fbanks of tensor shape $(400, 80, 1)$ and the output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- Applying "**style-transfer pooling**" directly results in a $(256, 256)$-shaped tensor (a symmetric positive semidefinite matrix):

$$\mathbf{S}_{c,c'} = \frac{1}{TF} \sum_{t,f} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- But spectrograms' characteristics are only invariant in the time axis:



Modeling style in images & spectrograms

- The input to the ResNet is 80-dim fbanks of tensor shape $(400, 80, 1)$ and the output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- Applying "**style-transfer pooling**" directly results in a $(256, 256)$-shaped tensor (a symmetric positive semidefinite matrix):

$$\mathbf{S}_{c,c'} = \frac{1}{TF} \sum_{t,f} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- But spectrograms' characteristics are only invariant in the time axis.
- So we may employ a $(10, 256, 256)$-shaped tensor:

$$\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- The shape of the tensor is **too large** (about 330K free variables).

Modeling style in images & spectrograms

- The input to the ResNet is 80-dim fbanks of tensor shape $(400, 80, 1)$ and the output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- Applying "**style-transfer pooling**" directly results in a $(256, 256)$-shaped tensor (a symmetric positive semidefinite matrix):

$$\mathbf{S}_{c,c'} = \frac{1}{TF} \sum_{t,f} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- But spectrograms' characteristics are only invariant in the time axis.
- So we may employ a $(10, 256, 256)$-shaped tensor:

$$\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- However, the tensor is **too large** (about 330K free variables).

Regularizing & scaling down the tensor

- The output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- **Channel-wise dropout:** We drop whole channels with probability 0.25.
- **Merge frequency bins**: We reshape the tensor $\mathbf{Y}$ by merging 2 consecutive frequency bins. The shape becomes:

$$(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256).$$

- **Reduce #channels**: We reduce the #channels by projecting $\mathbf{Y}$ to a 3D tensor $\mathbf{L}$ of shape (F_r, C, C'), where $C = 256$, $C' = 64$:

- **Rationale for $\mathbf{L}$ being a 3D tensor**: Estimate a different linear combinations of channels for each frequency range.

Regularizing & scaling down the tensor

- The output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- **Channel-wise dropout:** We drop whole channels with probability 0.25.
- **Merge frequency bins**: We reshape the tensor $\mathbf{Y}$ by merging 2 consecutive frequency bins. The shape becomes:

$$(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256).$$

- **Reduce #channels**: We reduce the #channels by projecting $\mathbf{Y}$ to a 3D tensor $\mathbf{L}$ of shape (F_r, C, C'), where $C = 256$, $C' = 64$:

- **Rationale for $\mathbf{L}$ being a 3D tensor**: Estimate a different linear combinations of channels for each frequency range.

Regularizing & scaling down the tensor

- The output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- **Channel-wise dropout:** We drop whole channels with probability 0.25.
- **Merge frequency bins**: We reshape the tensor $\mathbf{Y}$ by merging 2 consecutive frequency bins. The shape becomes:

$$(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256).$$

- **Reduce #channels**: We reduce the #channels by projecting $\mathbf{Y}$ to a 3D tensor $\mathbf{L}$ of shape (F_r, C, C'), where $C = 256$, $C' = 64$:

- **Rationale for $\mathbf{L}$ being a 3D tensor**: Estimate a different linear combinations of channels for each frequency range.

Regularizing & scaling down the tensor

- The output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- **Channel-wise dropout:** We drop whole channels with probability 0.25.
- **Merge frequency bins**: We reshape the tensor $\mathbf{Y}$ by merging 2 consecutive frequency bins. The shape becomes:

$$(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256).$$

- **Reduce #channels**: We reduce the #channels by projecting $\mathbf{Y}$ to a 3D tensor $\mathbf{L}$ of shape (F_r, C, C'), where $C = 256$, $C' = 64$:

$$\mathbf{Y}_{t,f,c'} \leftarrow \sum_{c} \mathbf{L}_{f,c,c'} \mathbf{Y}_{t,f,c}$$

- **Rationale for $\mathbf{L}$ being a 3D tensor**: Estimate a different linear combinations of channels for each frequency range.

Regularizing & scaling down the tensor

- The output of the ResNet $\mathbf{Y}$ is of shape $(T,F,C) = (50, 10, 256)$.
- **Channel-wise dropout:** We drop whole channels with probability 0.25.
- **Merge frequency bins**: We reshape the tensor $\mathbf{Y}$ by merging 2 consecutive frequency bins. The shape becomes:

$$(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256).$$

- **Reduce #channels**: We reduce the #channels by projecting $\mathbf{Y}$ to a 3D tensor $\mathbf{L}$ of shape (F_r, C, C'), where $C = 256$, $C' = 64$:

$$\mathbf{Y}_{t,f,c'} \leftarrow \sum_{c} \mathbf{L}_{f,c,c'} \mathbf{Y}_{t,f,c}$$

- **Rationale for $\mathbf{L}$ being a 3D tensor**: Estimate a different linear combinations of channels for each frequency range.

Correlation-based pooling

- The tensor $\mathbf{Y}$ with shape $(T_r, F_r, C') = (100, 5, 64)$ is **mean and variance normalized** along the time axis.
- The **channel-wise correlations** are calculated as

$$\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- Due to symmetry and variance normalization, the number of free variables is $N_v = F_r C'(C'-1)/2 = 10080$, variables in [-1,+1].
- **Flattening the tensor** and keeping only the unique N_v variables results in the proposed correlation-based pooling layer.
- The **embedding** is just a linear projection of size 256.

Correlation-based pooling

- The tensor $\mathbf{Y}$ with shape $(T_r, F_r, C') = (100, 5, 64)$ is **mean and variance normalized** along the time axis.
- The **channel-wise correlations** are calculated as

$$\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- Due to symmetry and variance normalization, the number of free variables is $N_v = F_r C'(C'-1)/2 = 10080$, variables in [-1,+1].
- **Flattening the tensor** and keeping only the unique N_v variables results in the proposed correlation-based pooling layer.
- The **embedding** is just a linear projection of size 256.

Correlation-based pooling

- The tensor $\mathbf{Y}$ with shape $(T_r, F_r, C') = (100, 5, 64)$ is **mean and variance normalized** along the time axis.
- The **channel-wise correlations** are calculated as

$$\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$$

- Due to symmetry and variance normalization, the number of free variables is $N_v = F_r C'(C'-1)/2 = 10080,$ variables in [-1,+1].
- **Flattening the tensor** and keeping only the unique N_v variables results in the proposed correlation-based pooling layer.
- The **embedding** is just a linear projection of size 256.

Network architecture & training

- **Architecture**
  - A **34-layer** full preactivation ResNet with 80-dim fbanks.
  - All convolutional **kernels** are 3×3.
  - **Squeeze and Excitation** in the first 2 blocks.
  - The number of **channels** is $(64, 128, 256, 256)$.
- **Training**
  - **SGD** with Momentum = 0.9.
  - **Additive Angular Margin** loss, scale: 30, margin: 0.1-0.3.
  - **Single GPU**, minibatch 256 (with **gradient accumulation**).
  - **Initial learning rate:** 0.2, divide by 2 when no progress on the dev set for 3000 model updates.

Network architecture & training

- **Architecture**
  - A **34-layer** full preactivation ResNet with 80-dim fbanks.
  - All convolutional **kernels** are 3×3.
  - **Squeeze and Excitation** in the first 2 blocks.
  - The number of **channels** is $(64, 128, 256, 256)$.
- **Training**
  - **SGD** with Momentum = 0.9.
  - **Additive Angular Margin** loss, scale: 30, margin: 0.1-0.3.
  - **Single GPU**, minibatch 256 (with **gradient accumulation**).
  - **Initial learning rate:** 0.2, divide by 2 when no progress on the dev set for 3000 model updates.

Experiments on VoxCeleb

- **Covariance vs Correlation pooling**

	Extended set		Hard set	
Pooling method	EER(%)	minDCF	EER(%)	minDCF
Mean & std stats (baseline)	1.43	0.090	2.48	0.145
Covariance	1.48	0.092	2.49	0.148
Correlation	**1.22**	**0.079**	**2.17**	**0.128**

- Covariance: mean-only norm, $N_v = F_r C'(C'+1)/2$
- Correlation: mean & variance norm, $N_v = F_r C'(C'-1)/2$

Experiments on VoxCeleb

- **Frequency merging**

	Extended set		Hard set	
Pooling method	EER(%)	minDCF	EER(%)	minDCF
Mean & std stats (baseline)	1.43	0.090	2.48	0.145
Correlation w/o freq merging	1.39	0.087	2.33	0.137
Correlation w/ freq merging	**1.22**	**0.079**	**2.17**	**0.128**

- Frequency merging: $(T_r, F_r, C) = (2T, F/2, C) = (100, 5, 256)$

Experiments on VoxCeleb

- **Time vs Time-frequency pooling**

	Extended set		Hard set	
Pooling method	EER(%)	minDCF	EER(%)	minDCF
Mean & std stats (baseline)	1.43	0.090	2.48	0.145
Correlation, time-freq pooling	1.57	0.101	2.63	0.158
Correlation, time pooling	**1.22**	**0.079**	**2.17**	**0.128**

- Time-freq pooling: $\mathbf{S}_{c,c'} = \frac{1}{TF} \sum_{t,f} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$
- Time pooling: $\mathbf{S}_{f,c,c'} = \frac{1}{T} \sum_{t} \mathbf{Y}_{t,f,c} \mathbf{Y}_{t,f,c'}$

Experiments on VoxCeleb

- **2D vs 3D channel-wise projection**

	Extended set		Hard set	
Pooling method	EER(%)	minDCF	EER(%)	minDCF
Mean & std stats (baseline)	1.43	0.090	2.48	0.145
Correlation, 2D projection	1.41	0.091	2.40	0.143
Correlation, 3D projection	**1.22**	**0.079**	**2.17**	**0.128**

- 2D projection: $$\mathbf{Y}_{t,f,c'} \leftarrow \sum_{c} \mathbf{L}_{c,c'} \mathbf{Y}_{t,f,c}$$

- 3D projection: $$\mathbf{Y}_{t,f,c'} \leftarrow \sum_{c} \mathbf{L}_{f,c,c'} \mathbf{Y}_{t,f,c}$$

Experiments on VoxCeleb

- **Score normalization**

	Extended set		Hard set	
Pooling method	EER(%)	minDCF	EER(%)	minDCF
Mean & std stats (baseline)	1.43	0.090	2.48	0.145
Correlation	**1.22**	**0.079**	**2.17**	**0.128**
Mean & std stats (as-norm)	1.31	0.080	2.23	0.127
Correlation (as-norm)	**1.13**	**0.071**	**1.99**	**0.115**

- **AS-norm**: Adaptive Symmetric Score Normalization

Conclusions & Future work

Conclusions

- We proposed a **new pooling method** for speaker recognition that is based on **channel-wise correlations**.
- The method is **inspired by style-transfer** in computer vision, however several modifications were required to make it applicable to our task.
- Experiments show **notable** and **consistent improvement** on VoxCeleb.

Future work

- Replace dot-products with **kernel functions** (Gaussian, multinomial, etc).
- Combine it with **multihead attention**.
- Apply losses that make use of **statistical & geometric** properties of correlations (e.g. use KL, Jensen-Shannon, a.o. divergences).

Conclusions & Future work

Conclusions

- We proposed a **new pooling method** for speaker recognition that is based on **channel-wise correlations**.
- The method is **inspired by style-transfer** in computer vision, however several modifications were required to make it applicable to our task.
- Experiments show **notable** and **consistent improvement** on VoxCeleb.

Future work

- Replace dot-products with **kernel functions** (Gaussian, multinomial, etc).
- Combine it with **multihead attention**.
- Apply losses that make use of **statistical & geometric** properties of correlations (e.g. use KL, Jensen-Shannon, a.o. divergences).

Thank you!
Happy to answer your questions!

Themos Stafylakis, Johan Rohdin, Lukas Burget

Omilia
Conversational Intelligence

BRNO UNIVERSITY OF TECHNOLOGY