THE HONG KONG POLYTECHNIC UNIVERSITY
香港理工大學

Robust Speaker Verification Using Population-based Data Augmentation

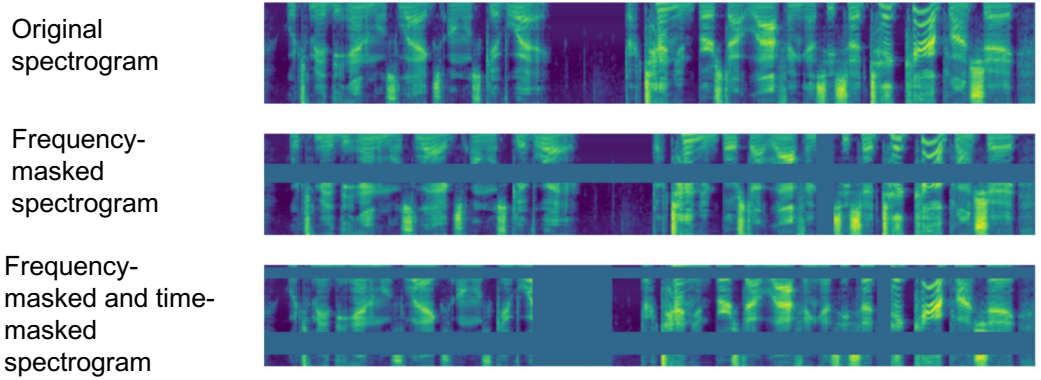
Weiwei LIN and Man-Wai MAK

ICASSP 2022

Department of Electronic and Information Engineering
The Hong Kong Polytechnic University

Data Augmentation for SV

- Data Augmentation is an important procedure in the training of speaker embedding networks.
- It is one of the reasons behind the x-vector's success
- Most of the augmentation methods add various noise and reverberation effects to waveforms:
  - A speech file is convolved with room impulse responses to emulate various reverberation effects.
  - Noise (speech, music, or babble) is added to the recording at different SNRs.
- Recently, it was found that we can also perform augmentation on speech features by performing masking along the frequency and time axes.



Problems with DA

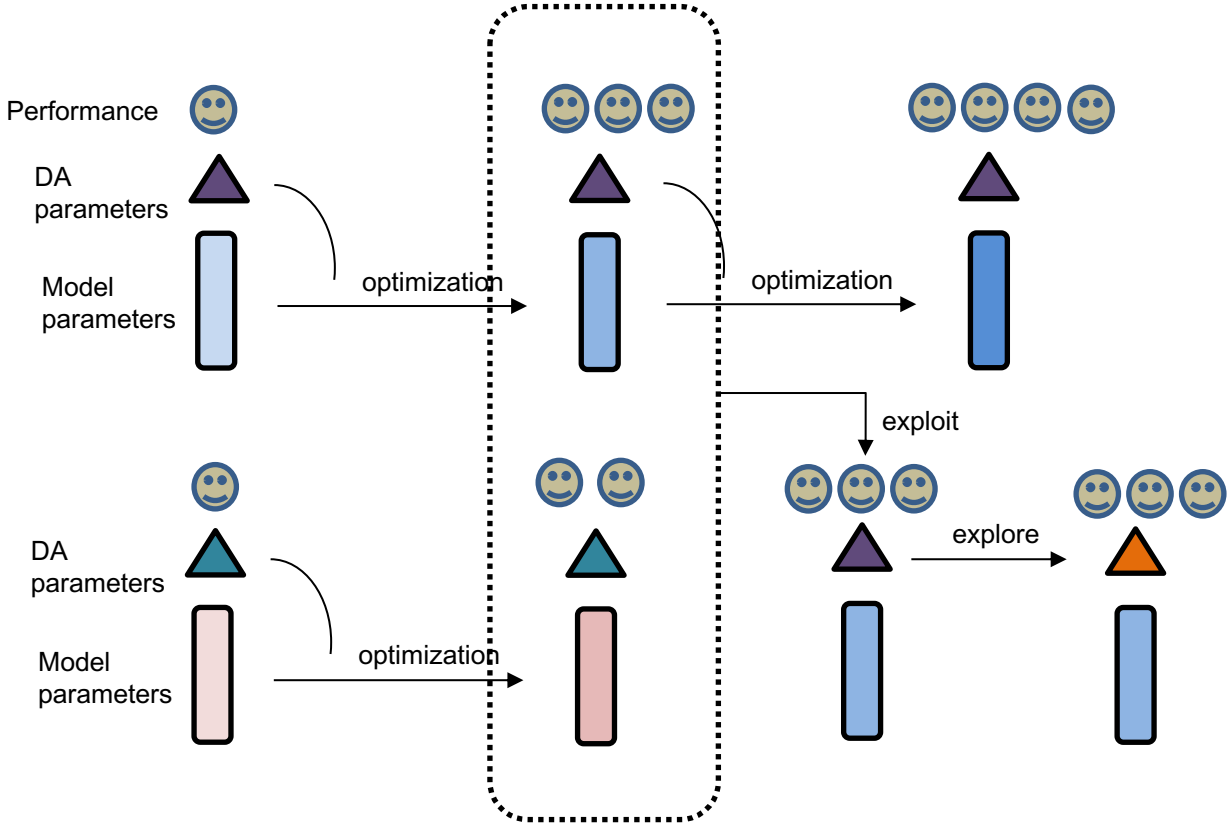
- The parameters of DA include magnitude and probability.
- **Magnitude** refers to how aggressive the augmentation is. For adding noise, it is the signal-to-noise ratio. For reverberation, it is the room size.
- **Probability** refers to how frequently we should apply a particular augmentation. A probability of 0 means we do not use that augmentation. A probability of 1 means we always apply the augmentation.
- DA should simulate the noise and reverberation levels of the deployment environment.
- However, we often use a set of pre-defined DA parameters whose values were intuitively set instead of optimally determined.

Population-based Augmentation

PBA learns a schedule for changing the DA parameters. It involves the following steps.

- **Initialization**: The augmentation parameters for each model in a population are randomly initialized with some predefined ranges.
- **Optimization**: The network parameters of each model are optimized independently (using stochastic gradient descent (SGD)) on the augmented data.
- **Evaluation**: Each of the models in the population is evaluated on a validation set.
- **Exploitation**: The network parameters of the models in the bottom 25% of the ranked list are replaced by those in the top 25%.
- **Exploration**: Apply the “explore” function to the augmentation parameters.

Population-based Augmentation



- "Exploit" selects the top-ranked models and DA params.
- "Explore" randomly perturbs the DA params.

PBA Exploit Function

Algorithm 1 Applying data augmentation to a mini-batch

Input: mini-batch $\mathcal{X}$, **parameters** $\mathcal{H}$ ▷

$\mathcal{H}$ is a list of augmentation hyperparameters comprising $(trans, prob, mag)$

$\mathcal{L}$ = [ ] ▷ Empty List

for x in $\mathcal{X}$ **do**

  $x = \text{sample\_segment}(x, T)$ ▷ Sample a random segment with duration T

  for $(trans, prob, mag)$ in $\mathcal{H}$ **do**

    if random(0, 1) $< prob$ **then**

      $z = trans(x, mag)$

      $\mathcal{L}$ = append($\mathcal{L}$, z)

    else

      $\mathcal{L}$ =append($\mathcal{L}$, x)

    end if

  end for

end for

Return $\mathcal{L}$

PBA Explore Functions

Algorithm 2 PBA "explore" function for magnitude parameters. Magnitude parameters can be any from 0 to 9 inclusive.

Input: MagParams $\mathcal{M}$ ▷ $\mathcal{M}$ is a list of magnitude parameters
$\mathcal{M}_{\text{new}}$ = [ ] ▷ Initialize an empty list for new parameters
for m in $\mathcal{M}$ **do**
  if random(0, 1) < 0.2 **then**
    m_{new} = random_int(0, 9) ▷ resample a new parameter
  else
    inc = random_int(0, 3) ▷ Randomly choose an increment value
    if random(0, 1) < 0.5 **then**
      $m_{\text{new}} = m + inc$ ▷ Increase the aug. parameter
    else
      $m_{\text{new}} = m - inc$ ▷ Decrease the aug. parameter
    end if
  end if
  $m_{\text{new}} = \max\{0, m_{\text{new}}\}$ ▷ Clip m_{new} within [0, 9]
  $m_{\text{new}} = \min\{m_{\text{new}}, 9\}$ ▷ Clip m_{new} within [0, 9]
  $\mathcal{M}_{\text{new}} = \text{append}(\mathcal{M}_{\text{new}}, m_{\text{new}})$
end for
Return $\mathcal{M}_{\text{new}}$

Algorithm 3 PBA "explore" function for probability parameters. Probability parameters have possible values from 0 to 1.

Input: ProbParams $\mathcal{P}$ ▷ $\mathcal{P}$ is a list of probability parameters for augmentation
$\mathcal{P}_{\text{new}}$ = [ ] ▷ Initialize an empty list for new parameters
for p in $\mathcal{P}$ **do**
  if random(0, 1) < 0.2 **then**
    p_{new} = random(0, 1) ▷ resample a new parameter
  else
    inc = random(0, 0.3) ▷ Randomly choose an increment value
    if random(0, 1) < 0.5 **then**
      $p_{\text{new}} = p + inc$ ▷ Increase the aug. parameter
    else
      $p_{\text{new}} = p - inc$ ▷ Decrease the aug. parameter
    end if
  end if
  $p_{\text{new}} = \max\{0, p_{\text{new}}\}$ ▷ Clip p_{new} within [0,1]
  $p_{\text{new}} = \min\{p_{\text{new}}, 1\}$ ▷ Clip p_{new} within [0,1]
  $\mathcal{P}_{\text{new}} = \text{append}(\mathcal{P}_{\text{new}}, p_{\text{new}})$
end for
Return $\mathcal{P}_{\text{new}}$

Experiments

- **Training data for DNNs**: Voxceleb1 and Voxceleb2 development sets.
- **Test data**: VOiCES19 evaluation set
- **Validation data**: VOiCES19 development set
- **Acoustic vectors**: 40-dim filter bank
- **VAD**: Kaldi's energy-based VAD

Results and Conclusions

Comparison with Kaldi Augmentation

Network	Aug	EER
X-vector	Kaldi+SpecAug	6.87%
X-vector	PBA	4.82%
DenseNet121	Kaldi+SpecAug	5.53%
DenseNet121	PBA	3.98%

Ablation Study

Without	EER
None (using all aug.)	3.98%
Additive noise	4.42%
Reverb	4.63%
Time masking	4.03%
Freq masking	3.88%

- Deeper networks, such as DenseNet121, achieve much better performance than the X-vector networks.
- For both X-vector and DenseNet121, PBA obtains better performance than Kaldi augmentation.
- Ablation study shows that reverberation is the most important augmentation and frequency mask is the least important augmentation. Removing frequency masking actually improves the performance. This could be that it does not work well with other augmentation operations.