ROBUST SPEAKER VERIFICATION USING POPULATION-BASED DATA AUGMENTATION

Weiwei Lin and Man-Wai Mak

Dept. of Electronic and Information Engineering,
The Hong Kong Polytechnic University
Hong Kong SAR

ABSTRACT

Speaker recognition under environments with a low signal-to-noise ratio (SNR) and high reverberation level has always been challenging. Data augmentation can be applied to simulate the adverse environments that a speaker recognition system may encounter. Typically, the augmentation parameters are manually set. Recently, automatic hyper-parameter optimization using population-based learning has shown promising results. This paper proposes a population-based searching strategy for optimizing the augmentation parameters. We refer to the resulting augmentation as population-based augmentation (PBA). Instead of finding a fixed set of hyper-parameters, PBA learns a scheduler for setting the hyper-parameters. This strategy offers a considerable computation advantage over the grid search. We obtained high-performance augmentation policies using a population of six networks only. With PBA, we achieved an EER of 3.98% on the VOiCES19 evaluation set.

Index Terms— Robust speaker verification, data augmentation, population-based learning, hyper-parameter optimization

1. INTRODUCTION

One of the challenges of speaker recognition is to deal with adverse environmental factors such as noise and room reverberation. It has been shown that noise and room reverberation can severely degrade the performance of speaker recognition systems [1]. A simple solution is to use speech collected from noisy environments to train the speaker recognition systems. However, such data may be difficult to collect. A more convenient solution is to emulate the noisy environments by adding pre-collected noise. Reverberation effects can also be added to the speech by convolving the speech with room impulse responses [2]. This approach to simulating acoustic environments is referred to as data augmentation. Although data augmentation is typically applied to labeled data, it was shown in [3] that it can also be applied to unlabeled data.

This work was supported by the RGC of Hong Kong SAR, Grant No. PolyU 152137/17E and National Natural Science Foundation of China (NSFC), Grant No. 61971371.

Typically, the augmentation parameters, such as the signal-to-noise ratio (SNR) and room size, are manually set, which require a lot of trial-and-error and intuitions. Conventional hyper-parameter optimization techniques such as grid search are computationally intensive. Several automatic hyper-parameter search techniques for data augmentation have been proposed recently [4–6]. For example, the AutoAugment in [4] automatically optimizes the augmentation procedures for images classification. AutoAugment uses reinforcement learning to search for the best policy and sub-policies. Although AutoAugment shows substantial improvement over human-designed augmentation, it requires thousands of GPUs hours even on a moderate-size dataset [4].

Population based augmentation (PBA) was proposed in [5] for computer vision. Instead of using reinforcement learning to search for the augmentation strategies, PBA uses a population of potential models. PBA goes through the exploitation and exploration stages to select the best candidate model. The performance of the augmentation strategies discovered by PBA rivals that of AutoAugment. But PBA only requires a fraction of AutoAugment's computation. In this paper, we show that PBA can be used to find powerful data augmentation strategies for speaker verification. With only 6 GPUs, we were able to find an augmentation strategy that significantly outperforms the one used in the Kaldi recipes [2, 7].

2. METHODOLOGY

2.1. Data Augmentation on Waveforms

Most speaker verification systems use the data augmentation strategies in [2]. Specifically, Every speech file in the original dataset is accompanied by one of the following augmentations:

- *Reverberation*: The speech file is convolved with a randomly selected room impulse response (RIR) to emulate reverberation effects [8].
- *Music*: The signal from a randomly selected MUSAN's music file [9] is added to the speech file at an SNR between 5dB and 15dB.

- *Noise*: Noise from MUSAN is added to the recording intermittently with an SNR between 0dB and 10dB.
- *Babble*: Babble noise is added to the speech file with an SNR of 0dB–10dB.

If we have prior knowledge about the SNR and room condition of the deployment environment, we can set the augmentation parameters accordingly. However, this kind of manual setting is often not optimal and requires a lot of expert knowledge. Therefore, data-driven optimization is preferable.

2.2. Spectrum Augmentation

Spectrum augmentation was proposed in [10] for automatic speech recognition (ASR). We employed frequency masking and time masking for spectrum augmentation.

1. Frequency masking: f_0 is chosen from $[0, F - \nu)$, where F is the number of Mel frequency channels and ν is the size of the frequency mask. Then, Mel-frequency channels from f_0 to $f_0 + \nu$ are set to zero.
2. Time masking: t_0 is chosen from $[0, T - \tau)$, where T is the number of frames in the speech segment and τ is the size of the time mask. Then, frames from t_0 to $t_0 + \tau$ are set to zero.

2.3. Population-based Augmentation

Training of a deep neural network involves the optimization of the network's hyper-parameters and its network parameters. Conventionally, the optimization of these two sets of parameters is separated, and the hyper-parameters are fixed while training the network parameters. Population-based training (PBT) [11] is an asynchronous optimization method that jointly optimizes a population of models and their hyper-parameters. Instead of fixing the hyper-parameters throughout the entire training process, PBT discovers a schedule of hyper-parameter settings optimal for training a task-specific model. The advantage is that the hyperparameters can be evolved with the network parameters during the training process.

Population-based augmentation [5] makes use of PBT to find an optimal schedule or policy for setting the augmentation parameters during data augmentation and model training. In the context of speaker verification, the noise level and the reverberation level of the augmented data are varied to increase training data diversity. Also, a set of mask parameters (ν and τ) in spectrum augmentation can further diversify the training data. Because each augmentation transformation has at least one hyper-parameter, the search space of the augmentation parameters is vast, which makes optimizing the augmentation parameters very challenging.

For each augmentation policy, we have two parameters to be optimized, namely, "prob" and "mag". The former is the probability that a particular augmentation is applied, whereas the latter is the magnitude of the augmentation. For additive-noise augmentation such as adding music, noise, and babble, magnitude refers to the signal-to-noise ratio. For reverberation augmentation, magnitude refers to the reverberation level. For frequency and time masking, magnitude refers to the mask size. Each policy is represented by an $(trans, prob, mag)$-tuples, where $trans$ stands for transformation, which can be one of the augmentations stated in Sections 2.1 and 2.2.

PBA learns a schedule for the $prob$ and mag parameters when training a population of models. The schedule is represented by a list of (trans, prob, mag)-tuples. Specifically, PBA involves the following steps:

1. *Initialization*: The augmentation parameters for each model in a population are randomly initialized with some predefined ranges.
2. *Optimization*: The network parameters of each model are optimized independently (using stochastic gradient descent (SGD)) on the augmented data produced by Algorithm 1.
3. *Evaluation*: Each of the models in the population is evaluated on a validation set. The performance of each model is ranked based on the EER on the validation set.
4. *Exploitation*: The network parameters of the models in the bottom 25% of the ranked list are replaced by those at the top 25%.
5. *Exploration*: Apply the "explore" function in Algorithm 2 and Algorithm 3 to the hyper-parameters.

Step 2 *Optimization* is performed for each mini-batch. Steps 3–5 *Evaluation*, *Exploitation*, and *Exploration* are performed

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 = 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 $\mathcal{L}$

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_{new} = max$\{0, m_{\text{new}}\}$ ▷ Clip m_{new} within [0, 9]
  m_{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_{new} = max$\{0, p_{\text{new}}\}$ ▷ Clip p_{new} within [0,1]
  p_{new} = min$\{p_{\text{new}}, 1\}$ ▷ Clip p_{new} within [0,1]
  $\mathcal{P}_{\text{new}}$ = append($\mathcal{P}_{\text{new}}$, p_{new})
end for
Return $\mathcal{P}_{\text{new}}$

every 40 epochs. Step 2 to Step 5 are repeated until convergence.

3. EXPERIMENTS AND RESULTS

3.1. Data Preparation

The training data include the VoxCeleb1 development set and the VoxCeleb2 development set [12, 13]. For the baseline system, we followed the data augmentation strategy in the Kaldi SRE16 recipe.[1] The training data were augmented by adding noise, music, reverb, and babble to the original speech files in the datasets. After filtering out the utterances shorter than 400ms and the speakers with less than eight utterances, we were left with 7,302 speakers. We used filter-bank features with a frame length of 25ms and a frame shift of 10ms. The number of Mel-scale filters is 80. Mean normalization was applied to the filter-bank features.

3.2. PBA Setting

We used six GPUs and one GPU to train the PBA models and the regular x-vector network, respectively. It is worth noting that although PBA requires more computational resources during training, there is no additional cost during inference. We follow the Kaldi receipt for the configuration of reverberation. The starting values for ν and τ (see Section 2.2) in spectrum augmentation are 25 and 100, respectively. The initial probability ($prob$ in Algorithm 1) for all augmentations were uniformly sampled from 0.2 to 0.8. The initial SNR (mag in Algorithm 1) were uniformly sampled from 0dB to 10dB. We trained the PBA models for 320 epochs. After 80 epochs, we started validating the models every 40 epochs, using the VOiCES2019 development set as the validation data. Each model in the PBA population went through the same optimization step as the baseline model.

3.3. DNN Speaker Embedding

We experimented with two different networks, namely, the standard x-vector network as in [2] and DenseNet121. For the structure of the DenseNet121, readers can refer to [14, 15]. The networks were trained using additive margin softmax with a margin of 0.35 and a scale factor of 30. The networks were optimized using stochastic gradient descent (SGD). For each mini-batch, we randomly selected 64 utterances from the training set, and for each utterance, we randomly cropped a 400ms segment from the utterance, i.e., each mini-batch has 64 speech segments and each segment has 400ms. We defined one epoch as looping through 120,000 such segments. We trained the networks for 320 epochs. The embedding dimension of x-vectors is 128. The embedding dimension of DenseNet-121 is 256. Cosine scoring was used as a backend.

[1] https://github.com/kaldi-asr/kaldi/tree/master/egs/sre16/v2

Network	Aug. Policy	VOiCE19-EVAL	
		EER (%)	minDCF
X-vector	Kaldi+SpecAug	6.87%	0.521
X-vector (early stop)	Kaldi+SpecAug	6.43%	0.541
X-vector	PBA	4.82%	0.385
X-vector [18]	Modified Kaldi	4.90%	0.365
ResNet [19]	Modified Kaldi	5.09%	0.417
DenseNet121	Kaldi+SpecAug	5.53%	0.412
DenseNet121	PBA	3.98%	0.334

Table 1: Results on the VOiCE19 evaluation set. For "X-vector (early stop)", the VOiCES19 development set was used for determining when to stop the training.

Network	Aug. Policy	VoxCeleb1-Test	
		EER (%)	minDCF
X-vector	Kaldi+SpecAug	2.21%	0.199
X-vector	PBA	2.33%	0.187

Table 2: Results on the VoxCeleb1 test set.

3.4. Evaluations

We evaluated the performance of the augmentation methods on the VOiCES19 evaluation set [16].[2] The VOiCES19 development set was used as a validation set for PBA model selection. VOiCES19 focuses on speaker verification under high distracting noise and room reverberation. It was obtained by simultaneously playing the clean audio files from an audiobook corpus (LibriSpeech) [17] and playing distractor noise in several real rooms of various sizes. The reverberated and noise-contaminated speech was recorded by several far-field microphones placed at different locations in the rooms. We report results in terms of equal error rate (EER) and minimum detection cost function (minDCF).

4. RESULTS

4.1. Comparison with Kaldi's Augmentation Policy

In this section, we present the results of PBA and Kaldi's default augmentation baseline. We can see from Table 1 that compared with Kaldi's augmentation, PBA significantly improves the performance for both the x-vector network and DenseNet121 on the VOiCES19 evaluation set. We have also used the VOiCES19 development set as a validation set for the early stopping of x-vector network's training. However, its performance is still far behind PBA.

We may also compare the results obtained by PBA with some recent work [18, 19] in Table 1. Note that apart from the baselines (Xvector-Kaldi+SpecAug) and (DenseNet121-Kaldi+SpecAug), all of the other methods used the VOiCES19 development data for system development. For [18, 19], the development data were used for PLDA adaptation. For PBA, the development data were used for searching augmentation policies. The "Modified Kaldi" in Table 1 means that the augmentation strategies are based on Kaldi but were modified by the authors manually to optimize their systems. In [18], there are more augmented speech files than the original Kaldi's receipt. In [19], a portion of Mixer 6 speech was used to create babble noise. We can see that the x-vector system powered by PBA is on par with the highly optimized x-vector system in [18]. Note that our x-vector system does not use PLDA or any additional adaptation methods. Our DenseNet121 with PBA performs the best among all of the methods in the table.

We also presented the performance of the x-vector networks on the VoxCeleb1 test set in Table 2. Here, we do not observe obvious advantage of PBA, which is not surprising as the PBA models were selected using the VOiCES19 development set.

[2]https://voices18.github.io/images/VOiCES_eval_plan.v6.pdf

Removed Augmentation	VOiCE19-EVAL	
	EER(%)	minDCF
None	3.98%	0.334
Additive noise	4.42%	0.351
Reverb	4.63%	0.374
Time masking	4.03%	0.332
Frequency masking	3.88%	0.328

Table 3: Ablation study of augmentation policies on the VOiCE19 evaluation set.

4.2. Ablation study of augmentation policies

In this section, we present an ablation study of the augmentation policies used in PBA. Specifically, for each experiment, we removed "additive noise", "reverberation", "time masking", or "frequency masking" from the augmentation policies. Table 3 shows the results of the ablation study. It is apparent from the results that removing additive noise and reverberation has the most detrimental effect on the system while removing time masking does not deteriorate the performance.

5. CONCLUSIONS

In this paper, we proposed a population-based data augmentation method for speaker verification. By training and selecting from a population of models, PBA learns a scheduler for setting the augmentation parameters. The experimental results on the VOiCES19 evaluation set show that the proposed method significantly outperforms setting the augmentation parameters manually. In the future, we will explore using population-based methods to optimize the other aspects of speaker verification systems, such as the length of training segments and learning rates.

6. REFERENCES

[1] Colleen Richey, Maria A Barrios, Zeb Armstrong, Chris Bartels, Horacio Franco, Martin Graciarena, Aaron Lawson, Mahesh Kumar Nandwana, Allen Stauffer, Julien van Hout, et al., "Voices obscured in complex environmental settings (VOiCES) corpus," *arXiv preprint arXiv:1804.05053*, 2018.

[2] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust DNN embeddings for speaker recognition," in *Proc. ICASSP*, 2018, pp. 5329–5333.

[3] Weiwei Lin, Man-Wai Mak, Na Li, Dan Su, and Dong Yu, "Multi-level deep neural network adaptation for speaker verification using MMD and consistency regularization," in *Proc. ICASSP*, 2020, pp. 6839–6843.

[4] Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le, "AutoAugment: Learning augmentation policies from data," *arXiv preprint arXiv:1805.09501*, 2018.

[5] Daniel Ho, Eric Liang, Xi Chen, Ion Stoica, and Pieter Abbeel, "Population based augmentation: Efficient learning of augmentation policy schedules," in *Proc. International Conference on Machine Learning*, 2019, pp. 2731–2741.

[6] Ekin D Cubuk, Barret Zoph, Jonathon Shlens, and Quoc V Le, "RandAugment: Practical automated data augmentation with a reduced search space," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 2020, pp. 702–703.

[7] Daniel Povey, Arnab Ghoshal, Gilles Boulianne, Lukas Burget, Ondrej Glembek, Nagendra Goel, Mirko Hannemann, Petr Motlicek, Yanmin Qian, Petr Schwarz, et al., "The Kaldi speech recognition toolkit," in *Proc. IEEE Workshop on Automatic Speech Recognition and Understanding*, 2011.

[8] Tom Ko, Vijayaditya Peddinti, Daniel Povey, Michael L Seltzer, and Sanjeev Khudanpur, "A study on data augmentation of reverberant speech for robust speech recognition," in *Proc. ICASSP*, 2017, pp. 5220–5224.

[9] David Snyder, Guoguo Chen, and Daniel Povey, "Musan: A music, speech, and noise corpus," *arXiv preprint arXiv:1510.08484*, 2015.

[10] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D Cubuk, and Quoc V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," *arXiv preprint arXiv:1904.08779*, 2019.

[11] Max Jaderberg, Valentin Dalibard, Simon Osindero, Wojciech M Czarnecki, Jeff Donahue, Ali Razavi, Oriol Vinyals, Tim Green, Iain Dunning, Karen Simonyan, et al., "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[12] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Proc. Interspecch*, 2017, pp. 2616–2620.

[13] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Proc. Interspecch*, 2018, pp. 1086–1090.

[14] Weiwei Lin, Man-Wai Mak, Na Li, Dan Su, and Dong Yu, "A framework for adapting DNN speaker embedding across languages," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2810–2822, 2020.

[15] Weiwei Lin, Man-Wai Mak, and Lu Yi, "Learning mixture representation for deep speaker embedding using attention," in *Proc. Odyssey 2020 The Speaker and Language Recognition Workshop*, 2020, pp. 210–214.

[16] Mahesh Kumar Nandwana, Julien van Hout, Mitchell McLaren, Allen R Stauffer, Colleen Richey, Aaron Lawson, and Martin Graciarena, "Robust speaker recognition from distant speech under real reverberant environments using speaker embeddings.," in *Proc. Interspeech*, 2018, pp. 1106–1110.

[17] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books," in *Proc. ICASSP*, 2015, pp. 5206–5210.

[18] Pavel Matějka, Oldřich Plchot, Hossein Zeinali, Ladislav Mošner, Anna Silnova, Lukáš Burget, Ondřej Novotnỳ, and Ondřej Glembek, "Analysis of BUT submission in far-field scenarios of VOiCES 2019 challenge," in *Proc. Interspecch*, 2019, pp. 2448–2452.

[19] David Snyder, Jesús Villalba, Nanxin Chen, Daniel Povey, Gregory Sell, Najim Dehak, and Sanjeev Khudanpur, "The JHU speaker recognition system for the VOiCES 2019 challenge.," in *Proc. Interspeech*, 2019, pp. 2468–2472.