A COMPREHENSIVE STUDY ON SELF-SUPERVISED DISTILLATION FOR SPEAKER REPRESENTATION LEARNING

Zhengyang Chen[1,2] *, Yao Qian*[2]*, Bing Han*[1]*, Yanmin Qian*[1,*] *, Michael Zeng*[2]

[1] MoE Key Lab of Artificial Intelligence, AI Institute ,
X-LANCE Lab, Department of Computer Science and Engineering, Shanghai Jiao Tong University
[2]Microsoft Cognitive Services Research, USA

2022 IEEE Spoken Language Technology Workshop (SLT) | 979-8-3503-9690-4/23/$31.00 ©2023 IEEE | DOI: 10.1109/SLT54892.2023.10022470

ABSTRACT

In real application scenarios, it is often challenging to obtain a large amount of labeled data for speaker representation learning due to speaker privacy concerns. Self-supervised learning with no labels has become a more and more promising way to solve it. Compared with contrastive learning, self-distilled approaches use only positive samples in the loss function and thus are more attractive. In this paper, we present a comprehensive study on self-distilled self-supervised speaker representation learning, especially on critical data augmentation. Our proposed strategy of audio perturbation augmentation has pushed the performance of the speaker representation to a new limit. The experimental results show that our model can achieve a new SoTA on Voxceleb1 speaker verification evaluation benchmark (i.e., equal error rate (EER) 2.505%, 2.473%, and 4.791% for trial Vox1-O, Vox1-E and Vox1-H , respectively), discarding any speaker labels in the training phase.

Index Terms— Speaker representation learning, self-supervised learning, self-distillation, audio perturbation

1. INTRODUCTION

Benefiting from the advantages of deep learning, speaker verification systems have achieved great progress in the recent few years. Researchers have proposed different architectures [1, 2, 3, 4, 5] to encode speaker information and designed different kinds of supervised loss functions [6, 7, 8] to learn discriminative speaker representations. However, deep-learning-based methods always require a lot of labeled data for training to deliver a good-performance model. However, due to the data privacy issue and labeling cost, it is challenging for us to access speaker-labeled data. On the other hand, there is a massive amount of unlabeled data on the internet. There appeared many emerging technologies that leverage unlabeled data in the training process.

Self-supervised learning is one of the approaches. The most commonly used method for self-supervised speaker representative learning is the contrastive-learning based method [9, 10, 11, 12], which assumes that there is only one speaker in one utterance and different utterances contain different speakers. Then the system can do contrastive learning by maximizing the similarity between positive pairs sampled from the same utterance and the discrepancy between negative pairs sampled from the different utterances. However, it cannot guarantee that different utterances contain different speakers when no speaker labels are available. Recently, researchers in the computer vision field have proposed a bootstrap-based method [13], called BYOL, which can do self-supervised learning by only minimizing the distance between positive pairs. Following this study, another work called self-distillation with no labels (DINO) [14] simplified the model structure of BYOL and proposed a more effective training strategy to further improve the system performance. Researchers have also applied the DINO strategy in self-supervised speaker representation learning [15, 16].

Although the above attempts to use DINO for speaker representation learning are successful, there are still few investigations on the characteristics of this method to further push its performance. In this paper, we have done a comprehensive study on the DINO strategy used in self-supervised speaker representation learning. The major findings are listed as followings,

- Audio augmentation by adding noise and reverberation can significantly improve the performance of the DINO-based speaker representation learning system.
- Among the perturbation-based augmentations, pitch perturbation can further improve the performance only when more iterations are used in the training and tempo perturbation has little effect on the system performance.
- When the size of the dataset is fixed, optimizing the DINO objective becomes more difficult when there are more speakers and thus the system performance degrades.
- The output DINO distribution can be approximately considered as a one-hot speaker label when we ignore the small values, which indicates a strong correlation with the true label.
- The DINO strategy shows a great superiority over the contrastive learning-based method.

Based on the above findings, we pushed the self-supervised learning performance on the Voxceleb1 evaluation benchmark to a new limit.

2. RELATED WORK

In recent years, researchers have made great efforts to leverage the unlabeled data in the speaker verification task. In the early work, Stafylakis et al. [17] trained the model to extract meaningful speaker embedding by using a feature reconstruction loss and additional phone information. Nowadays, the contrastive-learning-based method and the bootstrap [13, 14] based method are two dominant approaches for self-supervised speaker representation learning.

For the self-supervised constrastive learning, Huh et al. [9] proposed augmentation adversarial training to help the network learn

This work was supported in part by the China NSFC projects (Grant No. 62122050 and No. 62071288), and in part by Shanghai Municipal Science and Technology Major Project (Grant No. 2021SHZDZX0102).

*Corresponding author.

978-1-6654-7189-3/22/$31.00 ©2023 IEEE

channel invariant information and Zhang et al. [11] proposed a channel invariant loss to learn more robust speaker representation. Further, Xia et al. [10] proposed a prototypical memory bank to enlarge the negative sample pool and avoid the latent false negative pairs. Besides, Sang et al. [12] proposed a siamese network and a self-supervised regularization strategy to further improve the contrastive learning performance.

The bootstrap-based method is an emerging technology. Recent works mainly focus on DINO [14] based bootstrap method. Heo et al. [16] found that the DINO training strategy can converge better when there are fewer speakers in the training set and they proposed a curriculum learning strategy to improve the performance. Jung et al. [15] combined the DINO training strategy with a raw waveform-based neural network. However, compared with the contrastive-learning-based method, the bootstrap-based method is less studied in the field of self-supervised speaker representation learning. In this paper, we will give a deep investigation on applying the DINO strategy in self-supervised speaker verification task.

3. METHOD

In this section, we introduce the self-**di**stillation with **no** labels (DINO) based self-supervised training strategy and apply it to learn speaker representation using unlabeled data.

3.1. Self-Distillation with No Labels

As shown in Figure 1, the DINO strategy trains the model in a knowledge distillation (KD) paradigm. The output from the student network is optimized to match the output from the teacher network. Different from the common KD method, the teacher model in DINO is not pre-trained but derived from the student network. Besides, the student and teacher networks share the same architecture but have different parameters. Here, we denote student network as $\mathcal{F}_s$, parameterized by θ_s and denote teacher network as $\mathcal{F}_t$, parameterized by θ_t. The θ_t is exponentially moving averaged (ema) [18] from θ_s in the rule $\theta_t \leftarrow \lambda\theta_t + (1-\lambda)\theta_s$, where λ follows a cosine scheduler from 0.996 to 1 [13] during training process.

Both student and teacher networks contain two modules, a speaker embedding extractor g and a projection head h: $\mathcal{F} = h \circ g$. The speaker embedding extractor can be any kind of structure that can map the variable-length acoustic feature sequence, $\mathbf{x}$, to fix-dimensional speaker embedding, $\mathbf{e} = g(\mathbf{x})$. The projection head consists of some fully connected layers that maps the speaker embedding $\mathbf{e}$ to a K dimensional vector, $\mathbf{q} = h(\mathbf{e})$. For the $\mathbf{q}_s$ from the student network, another softmax function will be applied to map $\mathbf{q}_s$ to a probability distribution $\mathbf{p}_s$:

$$p_s^{(i)} = \frac{\exp\left(q_s^{(i)}/\tau_s\right)}{\sum_{k=1}^{K}\exp\left(q_s^{(k)}/\tau_s\right)} \tag{1}$$

where i is the i-th dimension of $\mathbf{p}_s$ and τ_s is the temperature to control the sharpness of the distribution $\mathbf{p}_s$. The probability distribution $\mathbf{p}_t$ from the teacher branch is calculated similarly but with a different temperature τ_t. In addition, there is a centering operation before softmax. The centering operation is very similar to the batch-norm operation but only does the mean normalization. The centering operation normalizes the teacher output $\mathbf{q}_t$ by a mean statistic $\mathbf{c}$: $\mathbf{q}_t \leftarrow \mathbf{q}_t - \mathbf{c}$. The statistic $\mathbf{c}$ is updated during the training process with a moving average strategy:

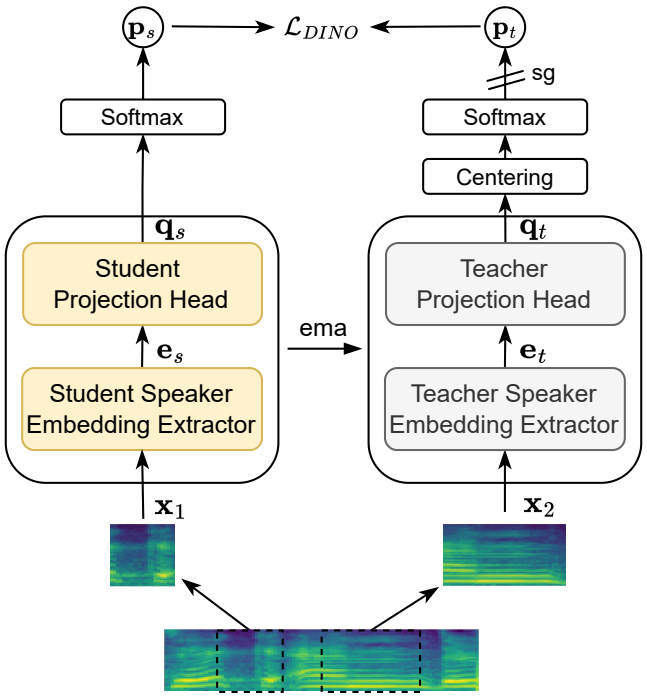

Fig. 1: **Speaker representation learning with DINO.** There could be multiples segments from one utterance fed into the student or teacher network. We only plot one segment in this figure. "sg" denotes the stop gradient operation.

$$\mathbf{c} \leftarrow m\mathbf{c} + (1-m)\frac{1}{B}\sum_{i=1}^{B}\mathbf{q}_t \tag{2}$$

where B is the batch size in one iteration and $0 < m < 1$ is the momentum factor. The student network is optimized to minimize the divergence between distribution $\mathbf{p}_t$ and $\mathbf{p}_s$ measured by cross-entropy loss. Because there is no additional supervisory signal here, the teacher model can easily collapse to the trivial solution that the teacher network always outputs a uniform distribution $\mathbf{p}_t$ or there is a fixed dimension dominating $\mathbf{p}_t$. The centering operation can avoid the fixed dimension dominating problem and the sharpening operation achieved by a smaller temperature τ_t can ease the uniform distribution problem. These two tricks can cooperate to avoid the model collapse problem.

Similar to the implementation in [14], we sampled multiple segments (views) from each utterances. In our experiment, we sample L long segments (global views) and M short segments (local views) from each utterance. All the segments are fed into the student network and only long segments are fed into the teacher segments. The DINO loss for each utterance is calculated as:

$$\mathcal{L}_{DINO} = \frac{1}{L(L+M-1)}\sum_{i=1}^{L}\sum_{\substack{j=1\\ j\neq i}}^{L+M}\text{CrossEntropy}(\mathbf{p}_t^i, \mathbf{p}_s^j) \tag{3}$$

3.2. Data Augmentation for Self-Supervised Learning

Many researchers [9, 12] have shown that the data augmentation strategy is the key to the success of self-supervised speaker representation learning. The self-supervised learning methods always have

the assumption that the speaker information is the only temporally consistent information in a sentence. Thus, when we want to extract some shared information from two segments of the same utterance, the shared information could be speaker information. However, the channel information is also consistent along the whole utterance. To avoid the model learning channel information, different noises or reverberations are added on the different segments from the same utterance to break the channel information consistency across different segments. Moreover, the audio perturbation [19] augmentation, which was mainly evaluated in speaker verification challenge [20, 21, 22], is firstly tried for DINO based self-supervised speaker representation learning in our experiments.

Algorithm 1: Pseudo code to construct DINO input

We denote the probability of additive noise (A) and reverberation (R) augmentation as $p_{\text{A+R}}$, the probability of pitch perturbation (P) as p_{P}, and the probability of tempo perturbation (T) as p_{T}, respectively.
For the short segments, the number of segments, segment duration and segment set from each audio are indicated as M, T_{short} and $\mathcal{X}^i_{short}$, and the counterparts of long segments are indicated as L, T_{long} and $\mathcal{X}^i_{long}$.
for *Utterance*$_i \in$ *Train Set* **do**
  Sample q_{P} from uniform distribution $\mathcal{U}(0,1)$
  if $q_P < p_P$ **then**
    Randomly select a pitch shift value α from $\{-200 \text{ cent}, 200 \text{ cent}\}$
    Utterance$_i$ = sox_pitch(Utterance$_i$, α)
  $\mathcal{X}^i_{short}$ = random_segments(Utterance$_i$, M, T_{short})
  $\mathcal{X}^i_{long}$ = random_segments(Utterance$_i$, L, T_{long})
  for *segment* $\in \mathcal{X}_{short} \cup \mathcal{X}_{long}$ **do**
    Sample q_{T} from uniform distribution $\mathcal{U}(0,1)$
    Sample $q_{\text{A+R}}$ from uniform distribution $\mathcal{U}(0,1)$
    if $q_T < p_T$ **then**
      Randomly select a tempo ratio β from $\{0.9, 1.1\}$
      segment = sox_tempo(segment, β)
    if $q_{A+R} < p_{A+R}$ **then**
      Randomly select noise_type from {additive, reverb}
      segment = add_noise(segment, noise_type)

The detailed data augmentation and segment sampling pipeline in DINO is shown in Algorithm 1. In our experiment, we applied the audio perturbation in two different sets up using sox [1] toolkit. The first one is "sox pitch", which will change the pitch of audio by α 'cent' (i.e. 100ths of a semitone). The second one is "sox tempo", which will change the tempo (speed) of the audio but keep the pitch. When the audio pitch is changed, we consider the audio from a new speaker. Thus, in Algorithm 1, pitch perturbation is applied on the whole utterance. After the pitch perturbation augmentation, several long and short segments will be sampled from the utterance. Then, additive noise, reverberation, and tempo perturbation augmentation are applied to each segment independently.

4. EXPERIMENT SETUP

4.1. Dataset

In our experiment, the Voxceleb2 dev set [23] is used as the training set. When we are doing self-supervised training, we do not use any speaker labels. We sample the audios from MUSAN dataset [24] as the additive noise and use the impulse responses in RIR[2] as the reverberation augmentation. We evaluate our system on three different evaluation trials Vox1-O, Vox1-E, and Vox1-H constructed from Voxceleb1 dataset [25].

4.2. System Configuration

In our experiment, we use ECAPA-TDNN [5] as the speaker embedding extractor. Besides, we denote the ECAPA-TDNN with the channel number equal to 512 as the ECAPA-TDNN-S (small) and the ECAPA-TDNN with the channel number equal to 1024 as the ECAPA-TDNN-B (big). Without the specific annotation, the ECAPA-TDNN-S will be used to analyze the characteristics of DINO training strategy.

The projection head in Figure 1 contains three fully connected layers with hidden dimension 2048 followed by ℓ^2 normalization and a weight normalization fully connected layer [26] which maps the final output to K dimension. In our experiment, we set the K to 65536, the same as the best configuration in [14]. The student temperature τ_s in equation 1 is set to 0.1. The teacher temperature τ_t is linearly increased from 0.04 to 0.07 during the first 30 epochs and then fixed. The momentum factor m in equation 2 is set to 0.9. Besides, when sampling the segments from one utterance, the long segment is set to 3s and the short segment is set to 2s. The long segments number and short segments number are set by default to 2 and 4, respectively. Cosine scheduler [13] is used to optimize the system, the initial learning rate and final learning rate are set to 0.2 and 0.00005, respectively. All the systems are trained for 150 epochs without specific annotation. In each epoch, we iterate all the utterances in the training set.

During the evaluation process, speaker embedding is extracted from the student speaker embedding extractor in Figure 1. The cosine score is used as the similarity measurement in our experiment.

5. RESULTS AND ANALYSIS

5.1. Augmentation Methods

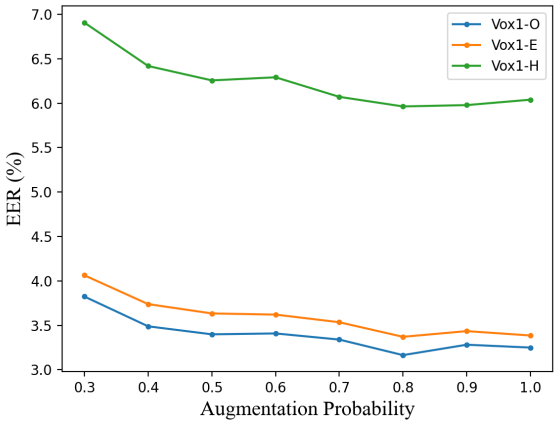

Fig. 2: **Changes of EER (%) with the probabilities of data augmentation.** Here, only additive noise and reverberation are applied. ECAPA-TDNN-S is used as the embedding extractor.

[1] http://sox.sourceforge.net/

[2] https://www.openslr.org/28/

In this section, we analyze the effect of different data augmentation methods in DINO training. We first verify their effectiveness by varying the p_{A+R} in Algorithm 1 and plot the result curve in Figure 2. From the figure, we find that extensive augmentation is very important in DINO training. DINO aims to model consistent information within an utterance. This consistent information can be speaker information or channel information. Additive noise and reverberation can break the channel information consistency across different segments from the same utterance and help the model focus on learning speaker information.

Table 1: **EER (%) results with different augmentation methods.** The abbreviations for augmentation methods: A (additive noise), R (reverberation), T (tempo perturbation) and P (pitch perturbation). Different from P shifting the pitch by 200 or -200 cent in Algorithm 1, for augmentation $\hat{\text{P}}$, the pitch is only shifted by 200 cent. The probability for each kind of augmentation method are set to: $p_{\text{A}} = 1.0$, $p_{\text{R}} = 1.0$, $p_{\text{T}} = 0.5$, $p_{\text{P}} = 0.5$ and $p_{\hat{\text{P}}} = 0.5$. The iteration N is the setup introduced in section 4.2.

Embedding Extractor	Iter #	Aug Type	Vox1-O	Vox1-E	Vox1-H
ECAPA-TDNN-S	N	-	15.22	15.28	19.95
		A	4.276	4.499	7.875
		R	5.212	5.626	9.100
		T	14.79	15.00	19.69
		P	18.49	18.30	23.53
		A + R	3.250	3.386	6.039
		A + R + T	**3.212**	**3.355**	**5.969**
		A + R + P	3.856	4.003	7.535
		A + R + $\hat{\text{P}}$	3.712	3.722	6.816
	2N	A + R	3.271	3.172	5.773
		A + R + T	3.165	3.190	5.778
		A + R + P	3.127	3.293	6.137
		A + R + $\hat{\text{P}}$	2.989	**3.052**	**5.642**
		A + R + T + P	3.085	3.248	6.149
		A + R + T + $\hat{\text{P}}$	**2.968**	**3.052**	5.652
ECAPA-TDNN-B	N	A + R	**2.957**	3.141	5.693
		A + R + T	2.962	**3.044**	**5.580**
		A + R + P	3.202	3.468	6.637
		A + R + $\hat{\text{P}}$	3.064	3.124	5.768
	2N	A + R	2.755	2.861	5.19
		A + R + T	2.718	2.781	5.141
		A + R + P	2.819	2.787	5.433
		A + R + $\hat{\text{P}}$	**2.505**	**2.473**	**4.791**

To further explore the effect of each data augmentation method, we do an ablation study by adding only one kind of augmentation at a time. Besides, we also explore the audio perturbation augmentation which has been introduced in section 3.2. We show the results of the ablation study of each augmentation method in Table 1. The observations from the first attempt by using N steps of iterations in the training process are 1) the ECAPA-TDNN-S model trained without data augmentation performs much worse than those with the data augmented by additive noise or reverberation; 2) audio tempo perturbation can improve EER but marginally; 3) audio pitch perturbation degrades the performance. The above observations, i.e. 2) and 3), are the same even when we combine the audio perturbation with the additive noise and reverberation. Such a phenomenon is inconsistent with the results in supervised training [22].

According to our analysis (the details will be shown in section 5.4), more speakers in the training set will make the training difficult and thus degrade the performance. The pitch perturbation will change the characteristics of speaker, which is equivalent to generating data with new speakers. To reduce this side effect, we first change the setup of pitch perturbation from P to $\hat{\text{P}}$, i.e., generating data with fewer new speakers. As expected, the side effect of pitch perturbation augmentation is mitigated a bit indicated by the changes of EER in the Table 1. Considering from another angle, the audio perturbation augmentation can generate more training samples and more training steps are required to make the model converge. So we double the training iterations. The EERs with $2N$ iterations in the Table 1 show that the side effect of pitch perturbation augmentation is further mitigated and the augmentation setup of "A+R+$\hat{\text{P}}$" outperforms that of "A+R".

The EER improvements from pitch perturbation are much more significant on the big model, ECAPA-TDNN-B, than those on the small model, ECAPA-TDNN-S. We didn't observe a similar phenomenon in the supervised scenario [3], where the small and large model always has similar benefits from the pitch perturbation augmentation. We conjecture that self-supervised approach and large model size can be complementary to each other for speaker representation learning. Self-supervised learning is a more difficult task due to the weaker supervision. Larger models can use the redundant parameters to store more information from training data. Self-supervised learning enables the model size growth to improve the model performance.

5.2. Comparing with Other Self-Supervised Learning Methods

In this section, we compared our methods with other self-supervised speaker representation learning methods. From the upper part of Table 2, we notice that the DINO strategy performs the best among all the self-supervised training strategies in speaker verification task. Enhanced with our proposed audio perturbation augmentation strategy, our systems outperform other systems with a very large improvement.

Table 2: **Performance comparison of different self-supervised training methods.** All the results are evaluated on the Vox1-O evaluation trial. In this table, our systems are trained with A+R+$\hat{\text{P}}$ augmentation and the iteration number is doubled. For fair comparison with other works, p_{target} in minDCF is set to 0.05 here.

Methods	Embedding Extractor	EER (%)	minDCF
AP+AAT [9]	Fast ResNet34	8.65	0.4540
MOCO+ProtoNCE [10]	TDNN	8.23	0.5900
CEL [27]	Fast ResNet34	8.01	N/A
Contrastive [28]	ECAPA-TDNN-S	7.36	N/A
SSReg [12]	Fast ResNet34	6.99	0.4340
Raw wave DINO [15]	RawNet3	5.40	0.3396
DINO + CL [16]	ECAPA-TDNN	4.47	0.3057
Ours	ECAPA-TDNN-S	2.989	0.2113
	ECAPA-TDNN-B	**2.505**	**0.1626**

5.3. Fine-tuning on Small Amount of Labeled Data

In many scenarios, the training process by leveraging a massive amount of unlabeled data is always considered as a pre-training stage and is usually followed by a fine-tuning stage where a small amount of labeled data from a specific domain is used for continuous training. In this section, we test how our system performs when we consider it as a pre-training model. We fine-tune our systems on the Voxceleb1 dev set and show the result on trial Vox1-O in Table 3. From the table, we find that our best system outperforms most of the other systems. Besides, our best system has a comparable performance with Unispeech-SAT Large model [29], which has been trained on tens of thousands of hours audio data and has a very large model size.

[3] In our supervised training experiment, after pitch perturbation augmentation, the EER (%) improvement on Vox-H is 0.125 (2.31 to 2.185) for ECAPA-TDNN-B and 0.16 (2.391 to 2.231) for ECAPA-TDNN-S.

Table 3: **EER (%) results after fine-tuned on Voxceleb1 dev set.** The model is fine-tuned on Voxceleb1 dev set supervised by AAM loss [7]. Data augmentation with additive noise and reverberation is applied in the fine-tuning. All the results are evaluated on Vox1-O evaluation trial.

	Methods (Model Param #)	Amount of Pre-training Data	Vox-O
Chen et al. [29]	UniSpeech-SAT Base ($\sim$ 100M)	94k hrs	*1.611
	UniSpeech-SAT Large ($\sim$ 320M)	94k hrs	*1.218
Zhang et al. [11]	AP + Channel-Invariant (1.4M)	2.36k hrs	3.88
Jung et al. [15]	Raw wave DINO (16.3M)	2.36k hrs	2.18
Heo et al. [16]	DINO + CL	2.36k hrs	1.84
Ours	ECAPA-TDNN-S (6.2M)	No Pre-training	2.484 (*2.234)
	ECAPA-TDNN-B (14.7M)	No Pre-training	2.356 (*2.180)
	ECAPA-TDNN-S (6.2M)	2.36k hrs	1.702 (*1.505)
	ECAPA-TDNN-B (14.7M)	2.36k hrs	1.335 (*1.228)

*: doing the adaptive score normalization in the scoring process

5.4. The number of Speakers and Utterances in the Training Corpus

In this section, we explore the impact of the number of speakers and utterances in the training set on the DINO self-supervised training strategy. Here, we randomly sample a small set from the original Voxceleb2 dev set in two different ways. In the first way, we sample the utterances according to the speaker identity. In the second way, we just randomly sample a small set from the original training set. Although we have no way to know how many people are in the unlabeled audio data, such an analysis can help us better understand the self-supervised training algorithm and guide how we collect data.

We list the corresponding results in Table 4. In the table, we take a half or a quarter subset from the original training data. As said above, we sample the subset according to speaker identity or directly randomly sampled each utterance. Interestingly, for two subsets with the same amount of utterances, the subset with more speakers performs worse. This phenomenon seems counter-intuitive and supervised training system shows the opposite trend. We speculate that DINO needs to traverse enough audios belonging to a particular person to model that person. For two subsets with the same number of voices, the model may not be able to model each speaker well when there are fewer utterances belonging to each speaker. In contrast, supervised training may focus on discriminating more speakers. Besides, Heo et al. [16] also found a similar phenomenon that reducing the speaker number in the DINO training can make the task easier and they proposed a curriculum learning strategy to improve the self-supervised learning strategy.

Table 4: **EER (%) results of systems trained on the dataset with different speakers and utterances.** N_{spk} and N_{utt} denote the total speaker number and total utterance number in Voxceleb2 development set. The ECAPA-TDNN-S is used as the embedding extractor. Additive noise and reverberation augmentation are applied in the training process with $p_{A+R} = 1.0$. We put the results of the supervised system in the bracket. The supervised system is trained with AAM loss [7] and we apply the additive noise and reverberation augmentation in the training process.

Speaker #	Utt #	Vox1-O	Vox1-E	Vox1-H
N_{spk}	N_{utt}	3.250 (1.117)	3.386	6.039
$N_{spk}/2$	$N_{utt}/2$	3.516 (1.744)	3.598	6.425
N_{spk}	$N_{utt}/2$	4.526 (1.510)	4.697	8.022
$N_{spk}/4$	$N_{utt}/4$	4.643 (2.803)	4.696	7.968
N_{spk}	$N_{utt}/4$	4.936 (2.026)	5.338	8.979

5.5. Probing the Characteristics of DINO Output Distribution

As introduced in section 3.1, the DINO is trained in a teacher-student paradigm. The model is optimized to match the output distribution from student and teacher networks. In our experiment, we find that there is one dimension dominating the teacher output distribution, which is very similar to a one-hot vector used in the classification-based cross-entropy loss. We're wondering if it is possible to consider the dimension index of the largest value in the distribution as some kind of pseudo speaker label. In Table 5, we calculate the normalized mutual information (NMI) [30] between true speaker labels and estimated pseudo labels on the Voxceleb2 dev set. We consider the index of the largest value or the indexes combination of the top two values in the teacher output distribution as the pseudo label. From the results, we find the pseudo label has a strong correlation with the true label. Besides, the index of the second largest value couldn't further improve the NMI.

Table 5: **Normalized mutual information (NMI) between estimated speaker label from DINO output distribution and true speaker labels.** The NMI value ranges from 0 to 1 and larger value represent stronger correlation. Speaker # denotes the pseudo speaker label number. In this table, the systems are trained with A+R+$\hat{\text{P}}$ augmentation and the iteration number is doubled.

Backbone	Argmax Top 1		Argmax Top 2	
	NMI	Pseudo Speaker #	NMI	Pseudo Speaker #
ECAPA-TDNN-S	0.816	2595	0.806	24725
ECAPA-TDNN-B	0.790	1319	0.785	17900

5.6. The Number of Segments

In section 3.1, we have introduced the multi-segment strategy, where multiple long and short segments are sampled from the same utterance. Here, we explore the importance of this strategy. The corresponding results are listed in the Table 6. From the results, we find that increasing the number of long and short segments can consistently improve performance. Because the average utterance duration of Voxceleb2 is 8s and more segments will cost more computation, we only sample at most two long segments and four short segments.

Table 6: **EER (%) results for different segmentation number setups.** The ECAPA-TDNN-S is used as the embedding extractor. Additive noise and reverberation augmentation are applied in the training process with $p_{A+R} = 1.0$.

Long Seg #	Short Seg #	Vox1-O	Vox1-E	Vox1-H
1	1	4.074	4.588	8.665
1	2	3.516	3.730	6.827
2	4	**3.250**	**3.386**	**6.039**

6. CONCLUSION

In this paper, we presented a DINO based self-supervised framework for speaker representation learning. The speaker pseudo labels predicted from the output distributions of our self-supervised model have a strong correlation with the true labels. Our ablation studies also indicate that data augmentation with the proposed audio perturbation strategy can further improve the model performance in addition to additive noise and reverberation. In the future, we plan to increase the data size by leveraging the unlabeled data in the wild, in which we haven't achieved a better performance by using the current settings. We believe we also need to scale up the model size accordingly.

7. REFERENCES

[1] Yuan Liu, Yanmin Qian, Nanxin Chen, Tianfan Fu, Ya Zhang, and Kai Yu, "Deep feature for text-dependent speaker verification," *Speech Communication*, vol. 73, pp. 1–13, 2015.

[2] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep neural network embeddings for text-independent speaker verification.," in *Interspeech*, 2017, pp. 999–1003.

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

[4] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matějka, and Oldřich Plchot, "But system description to voxceleb speaker recognition challenge 2019," *arXiv preprint arXiv:1910.12592*, 2019.

[5] Brecht Desplanques, Jenthe Thienpondt, and Kris Demuynck, "ECAPA-TDNN: Emphasized Channel Attention, Propagation and Aggregation in TDNN Based Speaker Verification," in *Proc. Interspeech 2020*, 2020, pp. 3830–3834.

[6] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *Proc. IEEE ICASSP 2018*. IEEE, 2018, pp. 4879–4883.

[7] Jiankang Deng, Jia Guo, Niannan Xue, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *Proc. CVPR*, 2019, pp. 4690–4699.

[8] Xu Xiang, Shuai Wang, Houjun Huang, Yanmin Qian, and Kai Yu, "Margin matters: Towards more discriminative deep neural network embeddings for speaker recognition," in *Proc. APSIPA ASC 2019*. IEEE, 2019, pp. 1652–1656.

[9] Jaesung Huh, Hee Soo Heo, Jingu Kang, Shinji Watanabe, and Joon Son Chung, "Augmentation adversarial training for unsupervised speaker recognition," in *Workshop on Self-Supervised Learning for Speech and Audio Processing, NeurIPS*, 2020.

[10] Wei Xia, Chunlei Zhang, Chao Weng, Meng Yu, and Dong Yu, "Self-supervised text-independent speaker verification using prototypical momentum contrastive learning," in *Proc. IEEE ICASSP 2021*. IEEE, 2021, pp. 6723–6727.

[11] Haoran Zhang, Yuexian Zou, and Helin Wang, "Contrastive self-supervised learning for text-independent speaker verification," in *Proc. IEEE ICASSP 2021*. IEEE, 2021, pp. 6713–6717.

[12] Mufan Sang, Haoqi Li, Fang Liu, Andrew O Arnold, and Li Wan, "Self-supervised speaker verification with simple siamese network and self-supervised regularization," in *Proc. IEEE ICASSP 2022*. IEEE, 2022, pp. 6127–6131.

[13] Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Guo, Mohammad Gheshlaghi Azar, et al., "Bootstrap your own latent-a new approach to self-supervised learning," *NeurIPS*, vol. 33, pp. 21271–21284, 2020.

[14] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin, "Emerging properties in self-supervised vision transformers," in *Proc. IEEE ICCV 2021*, 2021, pp. 9650–9660.

[15] Jee-weon Jung, You Jin Kim, Hee-Soo Heo, Bong-Jin Lee, Youngki Kwon, and Joon Son Chung, "Pushing the limits of raw waveform speaker recognition," *arXiv preprint*, vol. 2203, 2022.

[16] Hee-Soo Heo, Jee-weon Jung, Jingu Kang, Youngki Kwon, You Jin Kim, and Bong-Jin Lee abd Joon Son Chung, "Self-supervised curriculum learning for speaker verification," *arXiv preprint arXiv:2203.14525*, 2022.

[17] Themos Stafylakis, Johan Rohdin, Oldrich Plchot, Petr Mizera, and Lukás Burget, "Self-supervised speaker embeddings," in *Interspeech*, Gernot Kubin and Zdravko Kacic, Eds., 2019, pp. 2863–2867.

[18] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick, "Momentum contrast for unsupervised visual representation learning," in *Proc. IEEE CVPR 2022*, 2020, pp. 9729–9738.

[19] Hitoshi Yamamoto, Kong Aik Lee, Koji Okabe, and Takafumi Koshinaka, "Speaker augmentation and bandwidth extension for deep speaker embedding," in *Interspeech*, Gernot Kubin and Zdravko Kacic, Eds., 2019, pp. 406–410.

[20] Weiqing Wang, Danwei Cai, Xiaoyi Qin, and Ming Li, "The dku-dukeece systems for voxceleb speaker recognition challenge 2020," *arXiv preprint arXiv:2010.12731*, 2020.

[21] Miao Zhao, Yufeng Ma, Min Liu, and Minqiang Xu, "The speakin system for voxceleb speaker recognition challange 2021," *arXiv preprint arXiv:2109.01989*, 2021.

[22] Zhengyang Chen, Bei Liu, Bing Han, Leying Zhang, and Yanmin Qian, "The sjtu x-lance lab system for cnsrc 2022," *arXiv e-prints*, pp. arXiv–2206, 2022.

[23] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Interspeech*, B. Yegnanarayana, Ed., 2018, pp. 1086–1090.

[24] David Snyder, Guoguo Chen, and Daniel Povey, "MUSAN: A Music, Speech, and Noise Corpus," 2015, arXiv:1510.08484v1.

[25] Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Voxceleb: A large-scale speaker identification dataset," in *Interspeech*, Francisco Lacerda, Ed., 2017, pp. 2616–2620.

[26] Tim Salimans and Durk P Kingma, "Weight normalization: A simple reparameterization to accelerate training of deep neural networks," *NIPS*, vol. 29, 2016.

[27] Sung Hwan Mun, Woo Hyun Kang, Min Hyun Han, and Nam Soo Kim, "Unsupervised representation learning for speaker recognition via contrastive equilibrium learning," *arXiv preprint arXiv:2010.11433*, 2020.

[28] Ruijie Tao, Kong Aik Lee, Rohan Kumar Das, Ville Hautamäki, and Haizhou Li, "Self-supervised speaker recognition with loss-gated learning," in *Proc. IEEE ICASSP 2022*. IEEE, 2022, pp. 6142–6146.

[29] Zhengyang Chen, Sanyuan Chen, Yu Wu, Yao Qian, Chengyi Wang, Shujie Liu, Yanmin Qian, and Michael Zeng, "Large-scale self-supervised speech representation learning for automatic speaker verification," in *in Proc. IEEE ICASSP 2022*. IEEE, 2022, pp. 6147–6151.

[30] Roger Guimera, Marta Sales-Pardo, and Luís A Nunes Amaral, "Module identification in bipartite and directed networks," *Physical Review E*, vol. 76, no. 3, pp. 036102, 2007.