Enhanced label noise robustness through early adaptive filtering for the self-supervised speaker verification task

Abderrahim Fathan, Xiaolin Zhu,[*] Jahangir Alam.
Computer Research Institute of Montreal, Montreal (Quebec) H3N 1M3, Canada
`abderrahim.fathan@crim.ca, alice.zhuxl@gmail.com, jahangir.alam@crim.ca`

Abstract

Using clustering-driven annotations to train a neural network can be a tricky task because of label noise. In this paper, we propose a dynamic and adaptive label noise filtering method, called AdaptiveDrop which combines both label noise cleansing and correction simultaneously in cascade to combine their advantages. Contrary to other label noise filtering approaches, our method filters noisy samples on the fly from an early stage of training. We also provide a variant that incorporates sub-centers per each class for enhanced robustness to label noise by continuously tracking the dominant sub-centers via a dictionary table. AdaptiveDrop is a simple general-purpose method, performed end-to-end in only one stage of training, can be integrated with any loss function, and does not require training from scratch on the cleansed dataset. We show through extensive ablation studies for the self-supervised speaker verification task that our method is effective, benefits from long epochs of iterative filtering and provides consistent performance gains across various loss functions and real-world pseudo-labels.

1 Introduction

Label noise is an important problem in machine learning. Indeed, due to the memorization effects of deep models [1], prediction accuracy can drop as incorrect representations are learned, while model complexity and the required number of training samples may increase.

Automatic speaker verification (ASV) as one of the most convenient means of biometric recognition [2], uses the voiceprint of a speaker to verify his identity. Based on known utterances of a speaker, the speaker verification (SV) task aims to identify whether a speaker is a legitimate user or an imposter. With the advent of big data, recently researchers in the domain of ASV start to explore more affordable self-supervised learning (SSL) techniques using large noisy datasets. Indeed, since well-annotated datasets can be expensive to prepare, large-scale datasets are typically collected from the internet within automatic pipelines [3, 4]. Therefore, having a reliable selection of pseudo-labels (PLs) [5] and an effective mechanism to curate these noisy annotations becomes even more crucial.

Many cleansing methods exist in the label noise and data pruning literature [6, 7, 8, 9]. They rely in general on one or multiple stages of label noise cleansing followed by a supervised training stage that uses the curated training set. Besides, iterative approaches to mitigate label noise by refining PLs such as [10, 11] can be intimidating and cumbersome as they require more memory and computations. Contrary to most of these methods that only adopt one of the correction and filtering modes [12] or alternate between them [12] to build robust models, in this paper we propose a method called AdaptiveDrop to integrate both mechanisms in a single framework throughout the whole training to boost the effectiveness of each other for even enhanced robustness against noisy labels. As a

[*]Independent Researcher

consequence, this allows us to combine the advantages of both label correction and filtering where filtering boosts label correction by assuring that the corrected labels lead to a high cosine similarity with class centers which we add as a constraint to ensure higher quality of the corrected labels.

Our approach is performed in one stage of end-to-end training without the need to generate all sample embeddings of the whole dataset, thus avoiding additional memory requirements. Furthermore, we avoid to retrain the model from scratch on the final cleansed dataset as done by most other approaches [13, 14]. In fact, this can alter the performance of the model and can also negatively impact the generalization of the model as it is possible that filtering dramatically reduces the dataset. Besides, since we cannot know with certainty which samples are wrongly labeled, we believe it is important to have an adaptive and dynamic strategy to keep adjusting our filtering decisions throughout training: reprocess cleansed samples to recover them in later epochs in case of wrong drop out decisions or to remove inaccurate labels that were mistakenly included in training.

Finally, AdaptiveDrop is implemented as a general plug-and-play loss module where the backbone loss function can be simply replaced to adapt to one's desired configuration. It is agnostic to the training data size, simple and highly scalable as no training data information is stored, which makes it very suitable for online learning, and can speed up training considerably thanks to filtering. Moreover, our method is versatile and can be effectively applied to a wide range of problems and domains beyond speech or SV.

The contributions of this paper are as follows:

- We propose AdaptiveDrop, a novel general-purpose label noise filtering and correction method that can be used as a plug-and-play module with any loss function.
- A significant number of experimental results demonstrate the effectiveness and robustness of our method under various real-world noisy labels and loss functions.
- Contrary to wide practice, our ablations show that filtering label noise from an early stage of training is very important to avoid overfitting noisy labels.
- AdaptiveDrop outperformed numerous self-supervised SV baselines and achieved high SV performance.

2 Related Work and Motivation

One common way to leverage large unlabeled datasets for SV systems is to use clustering models [15, 16, 17], or to employ SSL-based objectives (SimCLR, MoCo) [18] to generate PLs and train the speaker embedding network using these labels in a discriminative fashion [11, 10]. However, clustering accuracy remains a limiting factor that constrains SV performance [10, 19]. Alternatively, iterative clustering-classification methods [10, 20] aim to jointly improve PLs and SV performance, but errors in PLs may still propagate, reducing overall performance. Thus, there is a need for more resilient training approaches to label noise to minimize its impact on generalization.
To tackle label noise, we can employ noise-robust algorithms [21, 22] to learn directly from noisy labels, or use label-cleansing approaches [9, 23] that remove or correct mislabeled samples. For instance, Subcenter-ArcFace [13] introduced a new loss function that relaxes ArcFace's intra-class constraint, improving robustness to label noise by forming a dominant sub-class of majority clean samples and grouping hard or noisy samples into non-dominant sub-classes.
To detect mislabeled instances, one can simply employ ad-hoc anomaly measures to filter out low-quality instances or correct those that are above a certain threshold [8, 9] (e.g. low cosine similarity, exceptionally high training loss value). Subcenter-ArcFace [13] adopts a filtering strategy to tackle label noise; as a first stage, it directly drops non-dominant sub-centers after the network is fully trained and has enough discriminative power, and introduces a constant angle threshold to drop high-confident noisy samples. In a subsequent stage, the model is retrained from scratch on the cleansed dataset. As a major difference with our AdaptiveDrop version that employs sub-centers, we dynamically track dominant sub-centers for each class from the beginning of training via a dictionary table, and perform a dynamic drop out of noisy samples at each training step from an early stage based on the latter table in order to learn progressively cleaner dominant sub-centers instead of waiting till training finishes. In our case, all training samples are reconsidered at each epoch to provide the opportunity to rectify our previous filtering mistakes. Besides, we are able to achieve considerable performance improvements without having to retrain our model from scratch.
In the context of self-supervised speaker recognition, to mitigate the negative effect of noise present

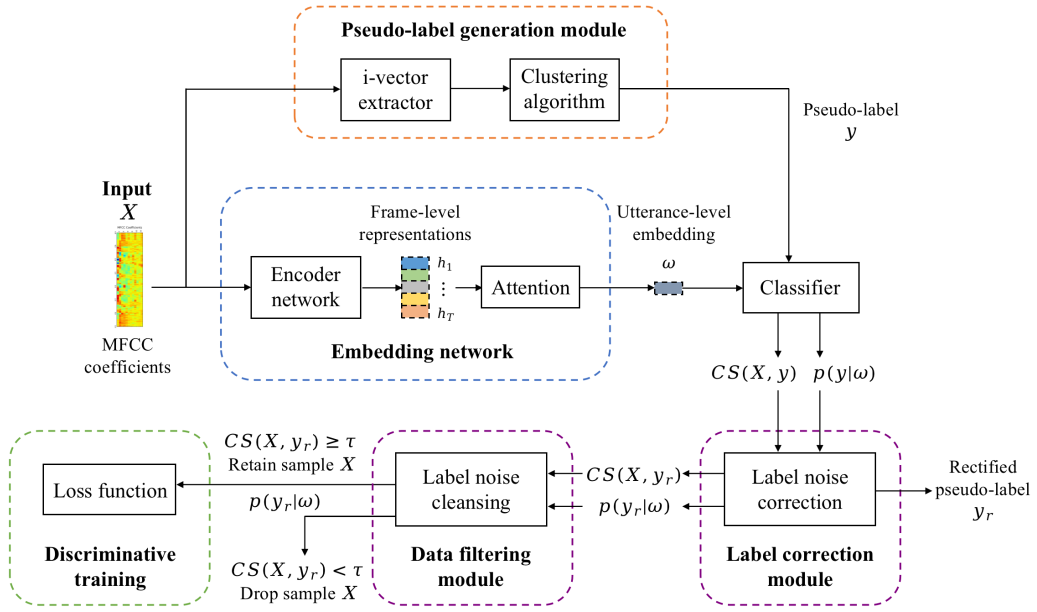


Figure 1: General process for training our self-supervised speaker embedding network using our proposed AdaptiveDrop method. The pipeline depicts the data flow and the label correction module employed for correcting noisy wrong labels, followed by the data filtering module used to remove the remaining misclassified samples. CS denotes the cosine similarity between the generated sample embedding ω and the class identity center embedding ω_y.

in PLs, [14] proposed an audio-visual two-step label noise detection and filtering method by dividing data into easy and peculiar (hard or noisy). [10] also proposed a two-stage iterative loss-gated learning strategy where clustering of generated embeddings is applied to generate PLs, and samples with a large training loss are dropped out. On the contrary, our AdaptiveDrop is simpler as it only employs cosine similarity between samples and their class centers. Additionally, our approach consists of only one stage of training in an end-to-end fashion which makes it easily generalizeable, lightweight, and does not rely on complementary information between several modalities such as speech and audio. In contrast to [10, 14], AdaptiveDrop does not store embeddings or cosine similarities for all samples and does not employ additional objectives such as contrastive losses which require large batch sizes.

3 Proposed method: AdaptiveDrop

A detailed schematic diagram of the AdaptiveDrop filtering scheme is presented in Figure 1 which depicts both our label filtering and label correction modules to mitigate label noise in clustering-driven pseudo-labels.

Given a training batch X and its corresponding noisy PLs Y, our approach employs noisy label correction and filtering simultaneously to enhance the label correction process and improve the accuracy of our PLs by avoiding wrong corrections. Besides, we alleviate the negative impact of over-cleaning since we first correct suspicious labels before filtering them, which can avoid removing too many samples. To achieve this, we use Cosine Similarity (CS) as a proxy label confidence indicator to detect the mislabeled instances and set a constant CS threshold τ for label filtering to drop high-confident noisy data. First, given each training sample x and its label y, we use the weights of our currently trained embedding network to extract embedding ω of x on the fly at each training step. We then compute cosine similarity $CS(x, y) = \cos(\omega, \omega_y) = \frac{\omega \cdot \omega_y}{\|\omega\| \|\omega_y\|}$ between ω and the current learnt class center/prototype embedding ω_y corresponding to class y. If $CS(x, y) >= \tau$, we retain sample x in the training batch, otherwise it is dropped out from the batch. Indeed, applying filtering after label correction helps improve correction by assuring that the corrected labels also have a high cosine similarity which is a second constraint to ensure higher quality of the corrected labels.

Here in our current implementation, we adopt the angular additive margin softmax loss [24], a.k.a ArcFace, to train our models. It is worth mentioning that any other loss can equivalently be used without loss of generality. The ArcFace objective is formulated as follows: $L_{ArcFace} = -\frac{1}{N}\sum_{i=1}^{N} log(\frac{e^{s(cos(\theta_{y_i}+m))}}{K_1})$, where $K_1 = e^{s(cos(\theta_{y_i}+m))} + \sum_{j=1,j\neq i}^{C} e^{scos\theta_j}$, N is the batch size, C is the number of classes, y_i corresponds to label index, θ_{y_i} represents the angle between the column vector of weight matrix $W_{y_i} \in \mathbb{R}^{512\times 1}$ of the class center and the i^{th} feature embedding $x_i \in \mathbb{R}^{512\times 1}$, where both W_{y_i} and x_i are l_2 normalized. $\theta_j = \arccos({W_j}^T x_i)$ is the angle between x_i and the j^{th} class center $W_j \in \mathbb{R}^{512\times 1}$. The scale factor s prevents the gradient from becoming too small during training, and m is a margin that ensures correct classes have higher similarity than incorrect ones.

Inspired by the BoundaryFace [25] loss, we partially adopt its label correction scheme (without its proposed regularization term to emphasize hard samples). For that, before computing the loss of each batch, we perform label correction by deciding whether to correct a label based on whether the sample is distributed within the decision boundary of the nearest negative class: If $\max\{\cos(\theta_k + m), \forall k \neq y_i\} - \cos(\theta_{y_i}) > 0$: $y_i = k$, otherwise label y_i remains the same.

Logically, if one network has been trained on an entire noisy dataset (pushing all sample embeddings to their class centers), using this same network to filter out these same wrongly labeled samples is not a reliable/optimal strategy [14]. On the contrary to the other aforementioned approaches which use fixed network's parameters to extract embeddings, in our case we let our network's parameters evolve after each training step. This provides a diversity in the extracted speaker embeddings for cosine similarity verification which has the potential to improve our label filtering and correction processes by making the filtering process act as an ensemble of models used to smoothly filter out noisy samples. We also perform dropping from a very early stage of training to mitigate overfitting. Therefore, this can induce converging to a better trade-off between the optimized training accuracy and the cosine similarity to class centers.

As samples are dropped out on the fly, and cosine similarities are already available to compute the loss function, AdaptiveDrop can considerably accelerate the training of large noisy datasets without having to decide beforehand which samples are reliably accurate.

Finally, we employ a first very short warm-up stage of training for a few first epochs (typically 5 epochs) to train our embedding network with all the PLs to have meaningful weights when label noise correction and filtering begin. Since neural networks initially focus on memorizing clean labels [1, 26], this ensures mislabeled data won't affect the model's weights before data correction and filtering.

Additionally, inspired from the idea of sub-centers from [13], we try to capture the complexity of the distribution of the whole dataset by employing a max pooling step on the subclass-wise cosine similarity scores ($W^T x_i$) across all our predefined K sub-centers. This helps to identify dominant clean sub-centers to use later as class prototypes for enhanced label noise robustness. Given a weight matrix $W \in \mathbb{R}^{C\times K\times 512}$ of all sub-centers, the Subcenter-ArcFace objective that we now use is formulated as follows: $L_{Subcenter-ArcFace} = -\frac{1}{N}\sum_{i=1}^{N} log(\frac{e^{s(cos(\theta_{i,y_i}+m))}}{K_1})$, where now $K_1 = e^{s(cos(\theta_{i,y_i}+m))} + \sum_{j=1,j\neq i}^{C} e^{scos\theta_{i,j}}$, and $\theta_{i,j} = \arccos(\max_k({W_{jk}}^T x_i))$ for $k \in 1, ..., K$.

Algorithm 1 shows our implementation of AdaptiveDrop using sub-centers. Aa a major difference with the original proposed method in [13], we use those dominant sub-centers from the start to compute cosine similarities instead of simply using the class centers or the corresponding maximum sub-centers of samples which can be noisier.

As a comparison, Subcenter-ArcFace [13] first generates embeddings of the whole dataset offline. These embeddings are used to compute the dominant sub-centers of classes which are then used to filter out noisy samples that have no dominant sub-center as their closest sub-center. This requires considerable additional memory usage and can lead to problems of over-cleaning [7]. To this purpose, we employ a sub-centers dictionary table DSCT $\in \mathbb{N}^{C\times K}$ (initially all entries are set to zero) to adaptively track the dominant ones per class from the beginning of training. As a rule, the ongoing sub-centers with the highest value per class are considered dominant (see Algorithm 1). At each training step, the entry corresponding to the class sub-center assigned to each sample is increased by 1 (except a first short warmup stage). This strategy can be considered as a smooth ensemble majority vote across different previous model weights.

Algorithm 1 AdaptiveDrop Algorithm (using Sub-centers)

```
Inputs: Training data $X$, noisy pseudo-labels $Y$, model $M$, loss function $l$, cosine similarity threshold $\tau$, epoch to start correcting noisy labels (ESC), epoch to start dropping out noisy labels (ESD), epoch to start tracking dominant sub-centers (ESTD), dominant sub-centers table DSCT $\in \mathbb{N}^{C\times K}$.
Output: The best model $M$.
Initialize: Table DSCT with zeros
for epoch $= 1$ to max_epochs do
    Extract embeddings $\omega$ for samples of current batch $x$.
    Compute cosine similarities ($CS$) between each sample in $x$ and its corresponding class sub-centers $\omega_{y,j}$ for $j \in \{1,..,K\}$.
    if $epoch \geq$ ESTD then
        #Update dominant sub-centers table (DSCT)
        closest_subcenter $= \arg\max_{j\in\{1,..,K\}}$CS $[y, j]$
        DSCT $[y,$ closest_subcenter$]$ $+= 1$
    end if
    if $epoch \geq$ ESC then
        Correct pseudo-labels $y$ of batch $x$.
    end if
    if $epoch \geq$ ESD then
        #Compute dominant sub-centers
        $K_{\text{dominant}} = \arg\max_{i\in\{1,..,K\}}$DSCT $[y, i]$
        Filter samples based on $CS(x, y) = \cos(\omega, \omega_{y,K_{\text{dominant}}})$.
        Set $(x_{\text{clean}}, y_{\text{clean}}) = \{(x_i, y_i)$ for $x_i, y_i \in (x, y)$ if $CS > \tau\}$
    else
        Set $x_{\text{clean}} = x$ and $y_{\text{clean}} = y$
    end if
    Compute loss $l$ with filtered $x_{\text{clean}}$ and $y_{\text{clean}}$.
    Perform gradient descent to update network parameters of model M.
end for
```

4 Experimental setup

We conducted a set of experiments based on the VoxCeleb2 dataset [4]. To train the embedding networks, we used the development subset of the VoxCeleb2 dataset, which consists of 1,092,009 utterances collected from 5,994 speakers. The evaluation was performed according to the Original (Vox1-O) VoxCeleb1 trials lists [3] which consists of 37,720 trials of 4,874 utterances spoken by 40 speakers. Besides, in Table 4 in Appendix B, we extend the evaluation of our different models to other evaluation sets.

We employ ECAPA-TDNN [27] as our speaker embedding network. As acoustic features for our SV experiments, we used 40-dimensional Mel-frequency cepstral coefficients (MFCCs) extracted at every 10 ms, using a 25 ms Hamming window via Kaldi toolkit [28]. Moreover, we have used waveform-level data augmentations including additive noise and room impulse response (RIR) simulation [29] to follow other SV works. Besides, we have applied augmentation over the extracted MFCCs features, analogous to the specaugment scheme [30]. All SV experiments have been run for 150 epochs using a single A40 GPU (around 2 days training), with a batch size of 200 MFCC samples. Scale factor $s = 30$ and margin $m = 0.2$ were used across all margin-based losses. Besides, we do not use score normalization and CS was used as a backend for verification scoring between enrollment and test embeddings. We refer to our loss implementation using label correction with BoundaryFace and refer to sub-centers combined with label correction with subcenter-BoundaryFace.

Finally, we use a default $K = 3$ sub-centers when sub-centers are employed and unless specified otherwise, a default CS threshold of $\tau = 0.423$. For AdaptiveDrop, we use ESD $= 5$, ESC $= 7$, ESTD $= 3$. Besides, to avoid training unstability especially for highly noisy thresholds, we only drop out a maximum of 50% of samples from each training batch. We use the CAMSAT clustering algorithm [31] to generate PLs using predefined numbers of clusters in {5000, 5994, 10000}. Code of our experiments is available at `https://github.com/fathana/AdaptiveDrop`

Table 1: Speaker verification performance in terms of EER (%) on the Vox1-O test set, using CAMSAT-based PLs and AdaptiveDrop with different losses for different starting epochs of label filtering. Results between parentheses refer to our ablation experiments using the maximum class sub-center instead of the dominant one to compute cosine similarities.

Epoch to start label filtering		**0**	**5**	**10**	**15**	**20**	**50**	**100**	**150**
ArcFace	10k clusters	3.314	2.641	2.609	2.763	2.715	2.837	3.059	-
	5994 clusters	3.595	2.651	2.89	2.842	2.821	3.07	3.012	-
	5k clusters	3.765	2.646	2.821	2.869	2.853	2.911	3.065	-
Subcenter-ArcFace	10k clusters	-	2.508 (2.757)	2.683	2.731	2.683	2.985	2.863	-
	5994 clusters	-	2.55 (2.577)	2.662	2.731	2.662	2.863	3.033	-
	5k clusters	-	2.699 (2.757)	2.842	2.757	2.81	2.884	2.842	-
BoundaryFace	10k clusters	-	2.513	2.635	-	2.757	2.831	2.863	2.752
	5994 clusters	-	2.858	2.784	-	2.9	3.006	2.874	3.049
	5k clusters	-	2.853	3.181	-	3.059	3.102	3.112	3.091

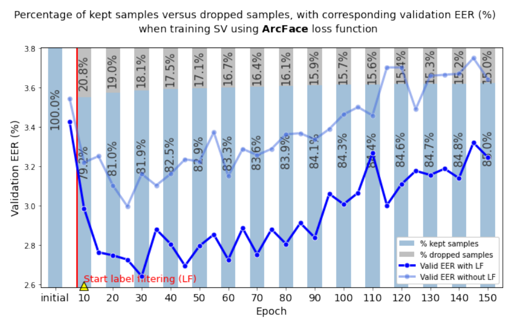
5 Results and Discussion

First of all, to confirm our hypothesis that early label noise filtering performs better, in Table 1 we perform label filtering starting from different epochs of training using ArcFace and Subcenter-ArcFace as backbone losses of AdaptiveDrop and BoundaryFace to integrate label correction into cleansing. As results demonstrate, we find that the best strategy for label filtering is to perform it as early as possible during training (after only 5 epochs of warmup in our case). We also find that a short warmup training is also required to produce reliable enough embeddings and cosine similarities (see case of starting from epoch 0). Indeed, contrary to current practice, we observe that the more we wait to start dropping out noisy labels, the worse is the downstream EER SV performance, which can be attributed to the capacity of the network to memorize the wrong labels over time.

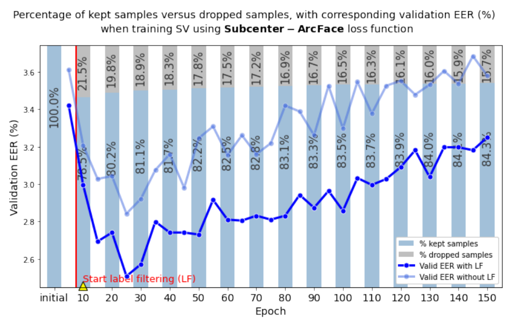
We can also see that employing sub-centers induces better robustness and generalization performance across the three numbers of clusters (e.g. 2.508% EER compared to 2.641% without sub-centers). We believe this is thanks to the cleaner learnt sub-centers of classes which provide more reliable cosine similarities to filter misclassified samples. Therefore, learning better and more generalizeable speaker embeddings. Additionally, we find that our choice of using the dominant class sub-center to compute cosine similarities instead of the current sub-center of maximum cosine similarity at each training step is relevant. Indeed, this provided better performance across the 3 PLs. Notably, we also observe that label correction with BoundaryFace does in general further improve SV performance, reaching an impressive 2.513 % EER performance on Vox1-O without sub-centers.

Finally, figure 2 shows the evolution of several important metrics over time. Importantly, we can see that the two losses achieve better EER performance with AdaptiveDrop and behave much better in terms of overfitting over epochs. Interestingly, we can notice that models using AdaptiveDrop can keep improving performance progressively for at least the first 25 epochs, benefiting from iterative filtering to keep refine the cleansed dataset. We also find that our variant incorporating label correction does not suffer from degradation of EER performance (almost no overfitting). It can also be seen that AdaptiveDrop leads to slightly better final label accuracy over epochs and that we do not suffer from problems of over-cleaning of training samples. Indeed, the percentage of cleansed samples ranges between 15.9% and 21.5% which we find reasonable given that label accuracy of the employed CAMSAT-based PLs is 70.9%. We can also observe a phenomenon where the percentage of removed samples decreases slightly over epochs, which can be attributed to the overall cosine similarities increasing over time (see Figure 3 in Appendix A) and to label correction (see percentage of dropped samples with label correction in (c) and (d) versus without label correction in (a) and (b)).

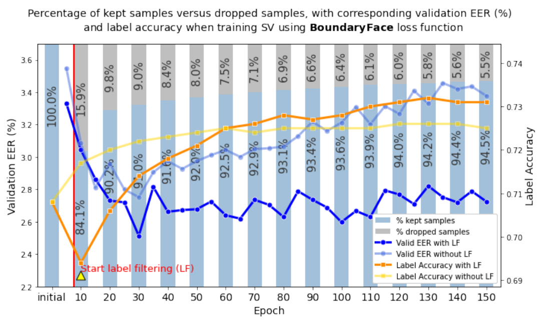
To assess the generalizability of AdaptiveDrop to other types of loss objectives, in Table 2 we train our embedding network with various losses with and without the proposed label noise filtering module. The relative improvements in percentage show that our method is able to considerably improve performance across all losses, up to more than 20% relative improvement. Besides, Figure 3 in Appendix A compares the behavior of our systems trained with and without AdaptiveDrop. According to the curves of validation EER performance, we can observe two phases: a first phase of 20 to 30 epochs where the model fits the clean labels and their simple patterns, therefore generalization



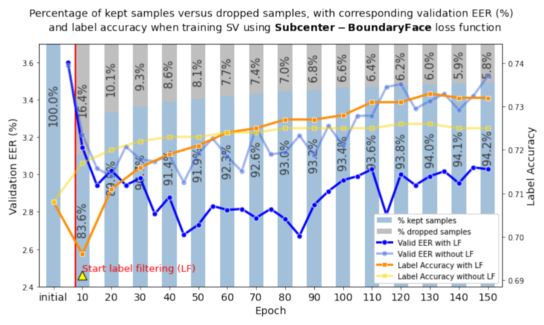
(a)



(b)



(c)



(d)

Figure 2: Validation EER performance over time of our SV system trained under different losses with AdaptiveDrop. We show the percentage of filtered samples versus retained ones over epochs and label accuracy when label correction is applied with and without our label filtering module.

keeps improving. And a second phase where EER performance keeps degrading because of the memorization of wrong labels and memorization of noisy patterns. From the loss curves, we can also see that AdaptiveDrop results in lower training loss (a sign of a cleaner filtered dataset) and also less overfitting to the noisy PLs over time compared to when AdaptiveDrop is not employed. Notably, we find the increasing cosine similarities over epochs confirm our hypothesis that employing cosine similarity later in training is less reliable because of the memorization effects and that label noise

Table 2: A comparison study of different losses with and without AdaptiveDrop, with the relative improvement. SV performance is reported in terms of EER (%) on the Vox1-O test set using CAMSAT-based PLs.

Loss function	**Without**	**With**	**Relative improvement (%)**
BoundaryFace [25]	2.752	2.513	8.7 ↑
CosFace [32]	2.863	2.577	10.0 ↑
MV-Arc-Softmax [33]	2.884	2.519	12.7 ↑
Subcenter-ArcFace [13]	2.943	2.508	14.8 ↑
Subcenter-ArcFace (batch size = 50)	-	2.407	-
AMSoftmax [34]	2.959	2.609	11.8 ↑
Subcenter-BoundaryFace	2.959	2.672	9.7 ↑
OCSoftmax [35]	2.969	2.678	9.8 ↑
AdaFace [36]	3.059	2.688	12.1 ↑
ArcFace [24]	3.134	2.641	15.7 ↑
CurricularFace [37]	3.192	2.529	20.8 ↑
DropMax [38]	8.006	6.702	16.3 ↑

Table 3: Some recent SOTA self-supervised SV approaches in EER (%) compared to AdaptiveDrop on the Vox1-O evaluation set.

SSL Objective	EER (%)
MoBY [18]	8.2
InfoNCE [10]	7.36
MoCo [39]	7.3
ProtoNCE [18]	7.21
PCL [18]	7.11
CA-DINO [40]	3.585
i-mix [41]	3.478
l-mix [41]	3.377
Iterative clustering [10]	3.09
CAMSAT [31]	3.065
Subcenter-BoundaryFace [42]	2.752
AdaptiveDrop (ours)	**2.407**

filtering should be performed early. This is to prevent overfitting the wrong labels from inflating the cosine similarities of misclassified samples which can make the use of cosine similarities as an indicator of wrong labels less effective in later usage of the model's weights.

Finally, Table 3 shows a comparison of our AdaptiveDrop approach using our best-performing configuration with Subcenter-ArcFace and CAMSAT-based PLs, compared to recent SOTA self-supervised SV approaches employing diverse SSL objectives with the same ECAPA-TDNN model encoder, on Vox1-O. The results show clearly that our adaptive filtering strategy provides large performance gains compared to all baselines, while being simple, fast and performed end-to-end in a single stage of training.

6 Conclusion

In this work, we proposed a novel and general-purpose label noise filtering method to improve robustness of self-supervised training in the presence of noisy labels. We proposed a framework with several variants encompassing label correction and/or sub-centers of classes to integrate with any type of loss objective. Our single-stage method is highly scalable, and adds no to very negligible memory and computational resources. Besides, our extensive experiments show that our method performs consistently well across all studied losses and lead to considerable improvements in self-supervised speaker verification.

7 Acknowledgment

The authors wish to acknowledge the funding from the Government of Canada's New Frontiers in Research Fund (NFRF) through grant NFRFR-2021-00338 and Natural Sciences and Engineering Research Council of Canada (NSERC) through grant RGPIN-2019-05381. Any opinions, findings, conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the NFRF & NSERC.

References

[1] N. B. D. K. D. Arpit, S. Jastrzębski *et al.*, "A closer look at memorization in deep networks," in *International conference on machine learning*. PMLR, 2017, pp. 233–242.

[2] J. H. Hansen and T. Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Processing Magazine*, 2015.

[3] A. Nagrani, J. S. Chung *et al.*, "Voxceleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[4] J. S. Chung, A. Nagrani, and A. Zisserman, "Voxceleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[5] M. N. Rizve, K. Duarte, Y. S. Rawat, and M. Shah, "In defense of pseudo-labeling: An uncertainty-aware pseudo-label selection framework for semi-supervised learning," *arXiv preprint arXiv:2101.06329*, 2021.

[6] B. Frénay, A. Kabán *et al.*, "A comprehensive introduction to label noise." in *ESANN*. Citeseer, 2014.

[7] I. Guyon, N. Matic, V. Vapnik *et al.*, "Discovering informative patterns and data cleaning." 1996.

[8] J.-w. Sun, F.-y. Zhao, C.-j. Wang, and S.-f. Chen, "Identifying and correcting mislabeled training instances," in *Future generation communication and networking (FGCN 2007)*, vol. 1. IEEE, 2007, pp. 244–250.

[9] C. E. Brodley and M. A. Friedl, "Identifying mislabeled training data," *Journal of artificial intelligence research*, 1999.

[10] R. Tao, K. A. Lee, R. K. Das, V. Hautamäki, and H. Li, "Self-supervised speaker recognition with loss-gated learning," in *ICASSP*. IEEE, 2022.

[11] J. Peng, C. Zhang, J. H. Cernockỳ, and D. Yu, "Progressive Contrastive Learning for Self-Supervised Text-Independent Speaker Verification," in *Proc. of Odyssey Workshop*, 2022.

[12] G. Jiang, J. Zhang, X. Bai, W. Wang, and D. Meng, "Which is more effective in label noise cleaning, correction or filtering?" in *Proceedings of the AAAI Conference*, vol. 38, no. 11, 2024, pp. 12 866–12 873.

[13] J. Deng, J. Guo, T. Liu, M. Gong, and S. Zafeiriou, "Sub-center arcface: Boosting face recognition by large-scale noisy web faces," in *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, Proceedings, Part XI 16*. Springer, 2020, pp. 741–757.

[14] R. Tao, K. A. Lee, Z. Shi, and H. Li, "Speaker recognition with two-step multi-modal deep cleansing," in *ICASSP*. IEEE, 2023, pp. 1–5.

[15] A. Fathan, J. Alam, and W. Kang, "On the impact of the quality of pseudo-labels on the self-supervised speaker verification task," in *NeurIPS 2022 Second ENLSP Workshop*, 2022. [Online]. Available: https://neurips2022-enlsp.github.io/papers/paper_51.pdf

[16] W. H. Kang, J. Alam, and A. Fathan, "l-mix: a latent-level instance mixup regularization for robust self-supervised speaker representation learning," *IEEE Journal of Selected Topics in Signal Processing*, 2022.

[17] ——, "An analytic study on clustering-based pseudo-labels for self-supervised deep speaker verification," in *SPECOM*, 2022.

[18] W. Xia, C. Zhang, C. Weng, M. Yu, and D. Yu, "Self-supervised text-independent speaker verification using prototypical momentum contrastive learning," in *ICASSP*. IEEE, 2021.

[19] B. Han, Z. Chen, and Y. Qian, "Self-supervised speaker verification using dynamic loss-gate and label correction," *arXiv preprint arXiv:2208.01928*, 2022.

[20] Y. Li, P. Hu, Z. Liu, D. Peng, J. T. Zhou, and X. Peng, "Contrastive clustering," in *AAAI*, 2021.

[21] M. Guan, V. Gulshan, A. Dai, and G. Hinton, "Who said what: Modeling individual labelers improves classification," in *Proc. of the AAAI conference on artificial intelligence*, vol. 32, no. 1, 2018.

[22] D. Rolnick, A. Veit *et al.*, "Deep learning is robust to massive label noise," *ICLR*, 2018.

[23] D. T. Nguyen, C. K. Mummadi, T. P. N. Ngo, T. H. P. Nguyen, L. Beggel *et al.*, "Self: Learning to filter noisy labels with self-ensembling," *arXiv preprint arXiv:1910.01842*, 2019.

[24] J. Deng, J. Guo, N. Xue, and S. Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," *IEEE TPAMI*, 2021.

[25] S. Wu and X. Gong, "Boundaryface: A mining framework with noise label self-correction for face recognition," in *European Conference on Computer Vision*. Springer, 2022, pp. 91–106.

[26] A. Fathan and J. Alam, "An analytic study on clustering driven self-supervised speaker verification," *PRL*, 2024.

[27] B. Desplanques, J. Thienpondt, and K. Demuynck, "ECAPA-TDNN: emphasized channel attention, propagation and aggregation in TDNN based speaker verification," in *Interspeech 2020*. ISCA.

[28] D. Povey, A. Ghoshal, G. Boulianne, L. Burget, O. Glembek, N. Goel *et al.*, "The kaldi speech recognition toolkit," in *In IEEE 2011 workshop*, 2011.

[29] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *Proc. of IEEE ICASSP*, 2018, pp. 5329–5333.

[30] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "Specaugment: A simple data augmentation method for automatic speech recognition." in *Interspeech*, 2019, pp. 2613–2617.

[31] A. Fathan and J. Alam, "Camsat: Augmentation mix and self-augmented training clustering for self-supervised speaker recognition," in *IEEE Automatic Speech Recognition and Understanding (ASRU) Workshop*, 2023.

[32] H. Wang, Y. Wang, Z. Zhou, X. Ji, D. Gong, J. Zhou *et al.*, "Cosface: Large margin cosine loss for deep face recognition," in *Proc. of the IEEE conference on CVPR*, 2018, pp. 5265–5274.

[33] X. Wang, S. Zhang, S. Wang, T. Fu, H. Shi, and T. Mei, "Mis-classified vector guided softmax loss for face recognition," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 241–12 248.

[34] F. Wang, J. Cheng, W. Liu, and H. Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, no. 7, pp. 926–930, 2018.

[35] Y. Zhang, F. Jiang, and Z. Duan, "One-class learning towards synthetic voice spoofing detection," *IEEE Signal Processing Letters*, 2021.

[36] M. Kim, A. K. Jain, and X. Liu, "Adaface: Quality adaptive margin for face recognition," in *Proceedings of the IEEE/CVF conference*, 2022, pp. 18 750–18 759.

[37] Y. Huang, Y. Wang, Y. Tai, X. Liu, P. Shen, S. Li, J. Li, and F. Huang, "Curricularface: adaptive curriculum learning loss for deep face recognition," in *proceedings of the IEEE/CVF conference*, 2020, pp. 5901–5910.

[38] H. B. Lee, J. Lee, S. Kim, E. Yang, and S. J. Hwang, "Dropmax: Adaptive variational softmax," *Advances in Neural Information Processing Systems*, 2018.

[39] J. Cho, J. Villalba, and N. Dehak, "The jhu submission to voxsrc-21: Track 3," *arXiv preprint arXiv:2109.13425*, 2021.

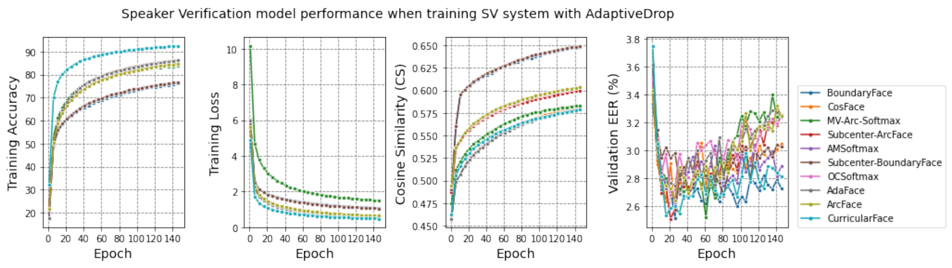
[40] B. Han, Z. Chen, and Y. Qian, "Self-supervised learning with cluster-aware-dino for high-performance robust speaker verification," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 32, pp. 529–541, 2023.

[41] A. Fathan and J. Alam, "On the influence of the quality of pseudo-labels on the self-supervised speaker verification task: a thorough analysis," in *IWBF*. IEEE, 2023.

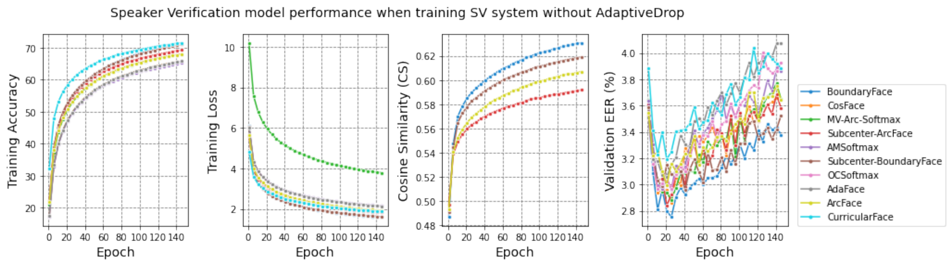
[42] A. Fathan, X. Zhu, and J. Alam, "An investigative study of the effect of several regularization techniques on label noise robustness of self-supervised speaker verification systems." *Odyssey 2024*, 2024.

A Behaviour over time

Figure 3 compares the behavior of our systems trained with and without AdaptiveDrop. According to the curves of validation EER performance, we can observe two phases: a first phase of 20 to 30 epochs where the model fits the clean labels and their simple patterns, therefore generalization keeps improving. And a second phase where EER performance keeps degrading because of the memorization of wrong labels and memorization of noisy patterns. From the loss curves, we can also see that AdaptiveDrop results in lower training loss (a sign of a cleaner filtered dataset) and also less overfitting to the noisy PLs over time compared to when AdaptiveDrop is not employed. Very importantly, we find the increasing cosine similarities over epochs confirm our hypothesis that employing cosine similarity later in training is less reliable because of the memorization effects and that label noise filtering should be performed early in training. This is to prevent overfitting the wrong labels from inflating the cosine similarities of misclassified samples which can make the use of cosine similarities as a proxy indicator of wrong labels less effective in later usage of the model's weights.



(a)



(b)

Figure 3: Training accuracy/loss, cosine similarity, and validation performance over time of our SV system trained under various loss functions, with and without AdaptiveDrop.

B Evaluation on other datasets

Finally, we report the evaluation of the performance of our different models with and without AdaptiveDrop on all VoxCeleb1 trials lists (Vox1-O, Vox1-E, and Vox1-H) in Table 4. Vox1-E and Vox1-H are test pairs drawn from the VoxCeleb1 development set. Vox1-E consists of 581,480 trials of 145,375 utterances spoken by 1251 speakers, and Vox1-H consists of 552,536 trials of 138,137 utterances spoken by 1190 speakers. Vox1-H is composed of identities that share the same gender and nationality, making it more challenging for verification compared to Vox1-E.

First, from the comparison of EER performance using true labels to train our models versus CAMSAT-based PLs, results show that speaker verification can degrade considerably in the presence of noisy training labels. Secondly, results across the different loss objectives also demonstrate that models trained using AdaptiveDrop generalize better across the 3 evaluation sets and benefit from our proposed label filtering method.

Table 4: EER (%) evaluation performance across the three Voxceleb1 test sets. Training is performed with and without AdaptiveDrop across different loss functions using the ground-truth labels or CAMSAT-based pseudo-labels.

Loss function	**AdaptiveDrop**	**Labels**	**Vox1-O**	**Vox1-E**	**Vox1-H**
ArcFace	✗	True labels	1.437	1.623	2.998
Subcenter-ArcFace	✗		1.384	1.563	2.874
ArcFace	✓	CAMSAT-PLs	2.641	3.238	5.578
ArcFace	✗		3.134	3.606	6.471
Subcenter-ArcFace	✓		2.508	**3.203**	5.519
Subcenter-ArcFace (batch size=50)	✓		**2.407**	3.216	5.485
Subcenter-ArcFace	✗		2.943	3.321	5.914
BoundaryFace	✓		2.513	3.216	**5.453**
BoundaryFace	✗		2.752	3.371	5.899
Subcenter-BoundaryFace	✓		2.672	3.331	5.578
Subcenter-BoundaryFace (batch size=50)	✗		2.662	3.223	5.658
Subcenter-BoundaryFace	✗		2.959	3.419	5.927
CosFace	✓		2.577	3.345	5.691
CosFace	✗		2.863	3.454	6.068