DropClass and DropAdapt: Dropping classes for deep speaker representation learning

Odyssey 2020 The Speaker and Language Recognition Workshop
1-5 November 2020, Tokyo, Japan

DropClass and DropAdapt: Dropping classes for deep speaker representation learning

Chau Luu, Peter Bell, Steve Renals

Centre for Speech Technology Research
University of Edinburgh, UK
{chau.luu, peter.bell, s.renals}@ed.ac.uk

Abstract

Many recent works on deep speaker embeddings train their feature extraction networks on large classification tasks, distinguishing between all speakers in a training set. Empirically, this has been shown to produce speaker-discriminative embeddings, even for unseen speakers. However, it is not clear that this is the optimal means of training embeddings that generalize well. This work proposes two approaches to learning embeddings, based on the notion of dropping classes during training. We demonstrate that both approaches can yield performance gains in speaker verification tasks. The first proposed method, DropClass, works via periodically dropping a random subset of classes from the training data and the output layer throughout training, resulting in a feature extractor trained on many different classification tasks. Combined with an additive angular margin loss, this method can yield a 7.9% relative improvement in equal error rate (EER) over a strong baseline on VoxCeleb. The second proposed method, DropAdapt, is a means of adapting a trained model to a set of enrolment speakers in an unsupervised manner. This is performed by fine-tuning a model on only those classes which produce high probability predictions when the enrolment speakers are used as input, again also dropping the relevant rows from the output layer. This method yields a large 13.2% relative improvement in EER on VoxCeleb. The code for this paper has been made publicly available.

1. Introduction

Deep speaker embeddings have become the state of the art technique in learning speaker representations [1, 2], outperforming the historically successful i-vector technique [3]. These speaker representations are crucial for many tasks related to speaker recognition, such as speaker verification, identification and diarization [4, 5, 6].

The networks used to generate speaker embeddings, such as the popular x-vector architecture [2], are typically trained on a speaker classification task, taking as input the acoustic features of an utterance and predicting which training set speaker produced the input utterance. By taking one of the upper layers of this network as an embedding, a fixed dimensional vector can be extracted for any given input utterance. This vector, due to the training objective that the network was given, is speaker-discriminative. Crucially, it has been found that these embeddings can be used to discriminate between speakers that were not present in the training set.

Although the approach of achieving speaker-discriminative embeddings through training via classification is common [7, 8], there exist other means in which to achieve this. For example, there are several approaches that are variants on triplet loss [9, 10, 11], which explicitly optimizes embeddings to move closer to same-class examples whilst moving further away from out-of-class examples. Another approach is the family of angular penalty loss functions [12, 13, 14, 15], which are similar to the standard softmax loss, but enforce a stricter condition on the decision boundary between classes by adding angular penalty terms for the correct class, thus encouraging larger inter-class distances and more compact intra-class distances.

We propose and make available code[1] for two methods aimed at achieving speaker-discriminative embeddings, both focused around the notion of dropping classes during training. The first technique, referred to as DropClass, continually changes the training objective for deep speaker embedding systems by periodically dropping a random subset of classes during training, such that the network is continually trained on many different classification tasks. This is conceptually similar to applying Dropout [16] on the output layer of a classification network while also disallowing training examples from the dropped classes. We argue that speaker recognition tasks have strong parallels with few-shot learning tasks and thus may benefit from a meta-learning style approach, which is what DropClass provides.

The second method that is proposed in this work, referred to as DropAdapt, can be applied to adapt a fully trained model to a set of enrolment speakers in an unsupervised manner. This is achieved by dropping the training classes which are are predicted by the model to be unlikely in the full set of enrolment utterances, rather than dropping randomly selected subsets of classes. We also show that the predicted distribution of speakers in the training set and test set can be heavily mismatched, which we argue negatively impacts performance. Our experiments show that DropAdapt can mitigate this distribution mismatch and that this correlates with improved speaker verification performance.

2. Dropping Training Classes

This work proposes two functionally similar techniques which revolve around the process of dropping classes during training. Despite this functional similarity, they have distinct applications, justifications, and links to literature which will be detailed in the following sections.

Supported by an EPSRC iCASE studentship collaboration with the BBC.

[1] https://github.com/cvqluu/dropclass_speaker

10.21437/Odyssey.2020-50

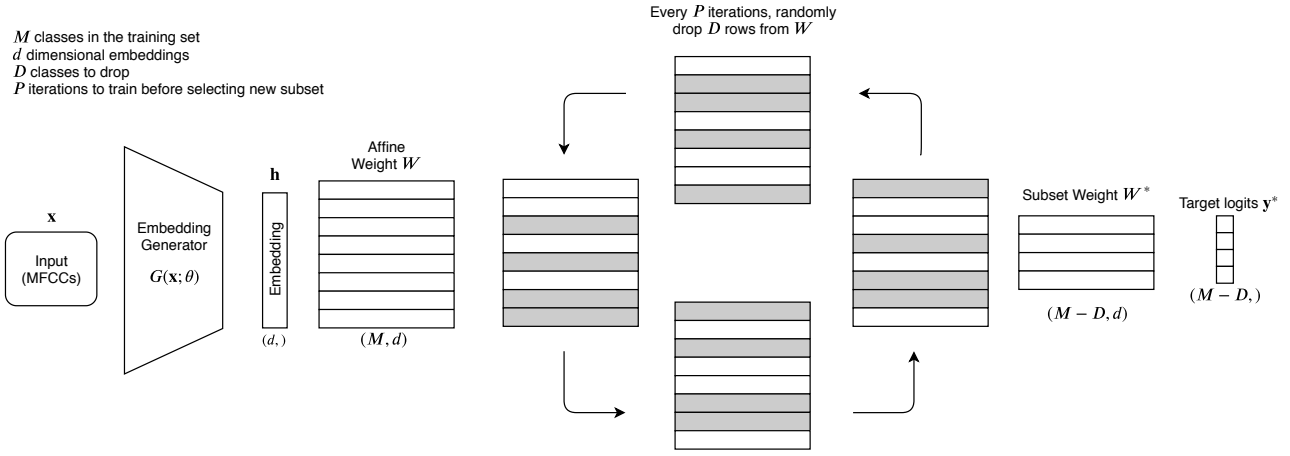

Figure 1: System diagram displaying the process of how classes are dropped throughout training in the proposed DropClass method.

2.1. DropClass

A typical architecture of a speaker embedding network, such as the successful x-vector architecture [2], has the following structure. This network is trained as a whole, but can be split into two components, G and C, which will be detailed below.

From an input $\boldsymbol{x}$ of acoustic features such as MFCCs, the first part of the network G parameterized by θ can produce a d-dimensional embedding $\boldsymbol{h}$.

$$\boldsymbol{h} = G(\boldsymbol{x}; \theta) \tag{1}$$

This is typically trained on a classification task, meaning the classification part of the overall network C, parameterized by ϕ acts on $\boldsymbol{h}$ to produce a prediction $\boldsymbol{y}$ for what class the input $\boldsymbol{x}$ belongs to.

$$\boldsymbol{y} = C(\boldsymbol{h}; \phi) \tag{2}$$

Both C and G are trained as a whole, usually via the standard cross entropy loss against a target one-hot vector $\hat{\boldsymbol{y}}$ which indicates which class out the set of M training classes, $\mathcal{N} : \{i, \ldots, M\}$, $\boldsymbol{x}$ belongs to.

For simplicity, the classification network C may be as rudimentary as an affine transform that projects the input embedding $\boldsymbol{h}$ into the correct number of dimensions, M. In this simplified case, the entirety of ϕ is a weight matrix W with the dimensions (M, d). Without a bias term, this changes Equation 2 to the following:

$$\boldsymbol{y} = \boldsymbol{h}W^T \tag{3}$$

where $\boldsymbol{y}$ contains the logits of the class prediction.

The proposed technique, referred to as DropClass, is detailed in Algorithm 1. When training with DropClass, every P iterations, a random subset of $\mathcal{N}$ is chosen: $\mathcal{R} \subset \mathcal{N}$ with size $M - D$ where D is a variable that determines how many classes should be dropped. P and D are configurable hyperparameters. The set $\mathcal{R}$ defines the permitted classes in the next P iterations. The rows of the weight matrix W which correspond to the subset of classes in $\mathcal{R}$ are selected to make a new matrix, W^*, which has the dimensions $(M - D, d)$, and the output of the resulting modification of Equation 3, $\boldsymbol{y}*$ has dimension $(M - D)$:

$$\boldsymbol{y}^* = \boldsymbol{h}W^{*T} \tag{4}$$

Result: Model trained with DropClass
Given: Feature extractor $G(\boldsymbol{x}; \theta)$, Classification affine matrix W
Set of all training classes $\mathcal{N} : \{i, \ldots, M\}$
D classes to drop per P iterations
Training dataset $\mathcal{D}_{\text{train}}$
while *not done* **do**
  Randomly sample proper subset of size $(M - D)$ from $\mathcal{N}$, $\mathcal{R} : \{j, \ldots, M - D\} \subset \mathcal{N}$
  $W^* \leftarrow W$[Class rows in $\mathcal{R}$]
  $\mathcal{D}_{\text{temp}} \leftarrow \mathcal{D}_{\text{train}}$[Examples from classes in $\mathcal{R}$]
  for P *iterations* **do**
    Train $G(\boldsymbol{x}; \theta)$ and W^* using $\mathcal{D}_{\text{temp}}$
  end
end

Algorithm 1: DropClass approach to training a deep feature extractor.

After P iterations, the process is repeated and a new proper subset is randomly selected, with the process continually repeated until training is completed (Figure 1).

This proposed method can be compared with a number of existing techniques in literature, in particular Dropout [16]. DropClass essentially drops units in the output classification layer and synchronizes this with the data provided to the model, ensuring that no dropped classes are provided while the corresponding classification units are dropped.

The effectiveness of Dropout has been justified by the technique performing a continuous sampling of an exponential number of thinned networks throughout training and then taking an average of these at test time [17, 18]. As a result of this model averaging, Dropout has been shown to reduce overfitting and generally improve performance [16], and has seen widespread adoption in many different applications of neural networks [19, 20, 21]. Similar in its justification, DropClass is continuously sampling from a large number of different classification tasks on which the embedding generator G must perform well, in theory making it agnostic to any one specific task.

This technique also has some similarity to some techniques in the field of meta-learning for few-shot learning, specifically Model-Agnostic Meta Learning (MAML) [22] and the related

technique Almost No Inner Loop (ANIL) [23]. MAML is a method for tackling few-shot learning problems by utilizing two nested optimization loops. The outer loop finds an initialization for a network which can adapt to new tasks quickly, whilst the inner loop uses the initialization from the outer loop and learns from a small number of examples from each desired task (referred to as the 'support set'), performing a few gradient updates.

Raghu et al [23] found the strength of MAML lay in the quality of the initialization found by the outer loop, with each task specific adaptation in the inner loop mostly reusing features already learned in the outer loop step. They proposed ANIL, which reduces the inner task-specific optimization loop to only optimize the classification layer, or 'head', of a MAML-trained network. Similar to DropClass, ANIL makes a distinction between the part of the overall classification network which generates discriminative features (referred to as the 'body'), and the classification head, which is more task specific. Raghu et al also proposed the No Inner Loop (NIL) method, which uses the cosine similarity between the generated features of an unseen example to the generated features of a small number of known examples to weight the classification prediction. This use of cosine similarity to compare embeddings is extremely commonplace in speaker recognition [24] and in practice, the inference step of the NIL technique is identical to a 1 to N speaker identification set up, if one considers the utterances from the N enrolment speakers to be the small number of labeled examples, the 'support set'.

This similarity of the problems of the few-shot learning and speaker recognition tasks has influenced the proposal of DropClass, both of which aim to produce a 'body' that generates features applicable to a distribution of tasks (sub-set classification) rather than to a single task. However, DropClass does not perform the outer and inner loops found in MAML/ANIL which explicitly optimizes the network to be robust to additional gradient steps per sub-task. Instead, DropClass encourages performance on all tasks by continually randomizing the training objective, implicitly encouraging the generated features to perform well across subtasks. Despite this, exploring ANIL and MAML for speaker representation learning would be a natural extension to this work. This extension would be particularly interesting considering the experiments on the NIL method (cosine similarity scoring) from Raghu et al [23], specifically Table 5. They found that MAML and ANIL trained models significantly outperformed 'multiclass training' models, where all possible classes were trained simultaneously. Considering the 'multiclass training' paradigm is the most common approach to training deep speaker embedding extractors, there could well be gains to be found in adopting a meta-learning approach to training speaker embedding extractors.

2.2. DropAdapt

Deep speaker representations are optimized to distinguish between the training set speakers, which is hoped to generalise to any given set of new unseen speakers. Generally however, the distribution of speakers in a desired held out evaluation set does not exactly match the distribution seen during training; that is, the expected distribution along the manifold of known speakers is often not replicated in the evaluation set. A clearer explanation for this can be seen if we examine what classes are predicted by the whole network when we give as input the utterances in the test set.

Starting from a trained model, for a given dataset $\mathcal{D}$ of N examples, the average probability assigned to each class can be calculated as follows,

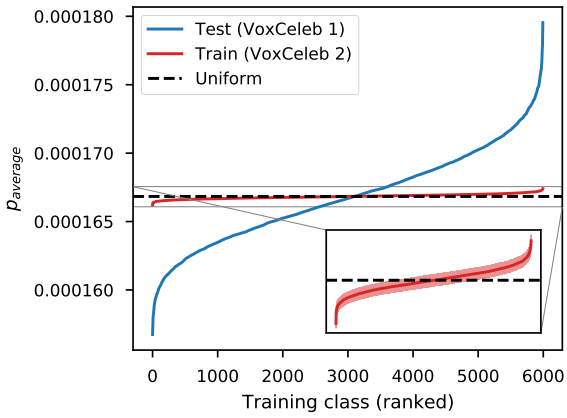

Figure 2: Comparison of the 5994 ranked training class probability predictions from a training set and test set, both provided with 40 unique speakers with 42 examples each. The training set probability has uncertainty bounds for 300 bootstrap sampled variations of speakers and examples.

$$\boldsymbol{p}_{\text{average}} = \frac{1}{N}\sum_{i=1}^{N} \text{softmax}(\boldsymbol{h}_i W^T) \tag{5}$$

where $\boldsymbol{h}_i$ is the embedding extracted from the i^{th} utterance, and W is the final affine weight matrix. The resulting M-dimensional vector $\boldsymbol{p}_{\text{average}}$ is a representation of the mean probability that the model predicts for the presence of each speaker across the N utterances. Provided a uniform distribution of speakers was used to train the model, it would be expected that the model predict a near uniform $\boldsymbol{p}_{\text{average}}$ for an input of training examples with uniform class distribution. This however may not be the case if the model is provided a selection of evaluation utterances with uniform speaker distribution.

This effect can be seen in Figure 2, where a trained model predicts close to the uniform distribution of classes when provided a uniform class distribution of training examples, but predicts a much more skewed $\boldsymbol{p}_{\text{average}}$ on the test set, with some training classes predicted to be much more likely than others. This is perhaps not a surprising result, as in the hypothetical situation of a test set with entirely one gender, a skewed $\boldsymbol{p}_{\text{average}}$ is expected. However, this skew is often not as clearly explainable as the hypothetical one-gender test set, and may have multiple contributing factors. For context, the VoxCeleb 1 test set has a 'good balance of male and female speakers' [25], and the speakers in it were chosen because their names began with the letter 'E'.

This observation can be interpreted in a number of ways. For example, it is well known that class imbalance is a significant impedance to performance in classification tasks [26], especially in cases in which training and inference have significantly different distributions. It is a natural extension to this that the performance of an embedding extracted from a classification network would degrade in performance in the same manner, which has been seen in the work of Huang et al and Khan et al [27, 28]. In these works, they found cost sensitive training and oversampling methods to increase the performance of learned representations.

Result: Model tuned with DropAdapt
Given: Trained feature extractor $G(\boldsymbol{x};\theta)$, trained W
Training set $\mathcal{D}_{\text{train}}$
Unlabeled Test/enrolment utterances $\mathcal{D}_{\text{enrol}}$
while *not done* **do**
  Calculate all $\boldsymbol{p}_{\text{average}}$ from $\mathcal{D}_{\text{enrol}}$ (see Equation 5)
  Rank classes by $\boldsymbol{p}_{\text{average}}$
  Select set of higher probability classes from $\mathcal{N}$, dropping lowest probability D classes $\rightarrow \mathcal{R}$
  if *Combine* **then**
    Assign all examples not in $\mathcal{R}$ same class label in $\mathcal{D}_{\text{train}}$
    $W^* \leftarrow W$[Class rows in $\mathcal{R}$]
  else
    $\mathcal{D}_{\text{train}} \leftarrow \mathcal{D}_{\text{train}}$[Examples from classes in $\mathcal{R}$]
    $W^* \leftarrow W$[Class rows in $\mathcal{R}$]
  end
  $W \leftarrow W^*$
  $\mathcal{N} \leftarrow \mathcal{R}$
  for P *iterations* **do**
    Train $G(\boldsymbol{x};\theta)$ and W using $\mathcal{D}_{\text{train}}$
  end
end

Algorithm 2: DropAdapt method for adapting a deep feature extractor to a chosen dataset

The second closely related interpretation and hypothesis is that the 'low probability' classes predicted by the model are in some way less important to the performance of the embeddings on the test set. These 'low probability' classes are suggested by the model's predictions to be less likely to be present in the test set. This might imply that distinguishing between these specific classes is not crucial to the end task.

Following from these interpretations, this technique, referred to as DropAdapt, works via dropping these low probability classes permanently to fine-tune a fully trained model, adapting the model to a test set and hopefully increasing performance. This is described in Algorithm 2. This method in theory should be applied only to a fully or near fully trained model, as an accurate estimation of the training class occupation must be obtained first.

To ensure an accurate probability estimation of the test set throughout the fine-tuning, this ranking (and dropping) of the least probable classes can be performed periodically, meaning this technique is functionally similar to the DropClass method above, except that classes are removed permanently, and the dropping of classes is determined by the probability criterion $\boldsymbol{p}_{\text{average}}$ instead of randomly.

A slight variation of this method explored in this work is referred to DropAdapt-Combine, in which instead of permanently removing these classes, all the low probability classes are combined into a single new class such that the examples belonging to the removed classes are not completely discarded.

This method can be compared to techniques in the fields of active learning and learning from small amounts of data, such as the Facility-Location and Disparity-Min models [29], which put heavy emphasis on selecting the right subset of examples in order to learn efficiently. These methods are typically used to capture the whole distribution of the desired dataset in as few examples as possible, encouraging a diverse and representative subset of examples. However, it is implied by Figure 2 that in this speaker embedding task, even if the whole training dataset were used, this may not be representative of the distribution found at test time. DropAdapt can be seen as a means of correcting this mismatch through subset selection for fine-tuning.

Buda et al [26] and Huang et al. [27] found oversampling minority classes to be an effective strategy in improving performance for neural networks on imbalanced datasets. Viewing this problem as a dataset imbalance problem, DropAdapt could also be interpreted as a corrective oversampling strategy, training additionally on those classes which are retained to better match the target distribution.

This train-test distribution mismatch is also closely linked to the field of domain adaptation and the domain-shift problem [30]. However, DropAdapt is primarily proposed as a means of adapting to a class distribution mismatch, as it is likely that $\boldsymbol{p}_{\text{average}}$ is less informative the greater the domain mismatch. Combining domain adaptation techniques with DropAdapt could be an interesting extension to this work.

3. Experiments

The following section details the experimental setup and the experiments performed utilizing the proposed methods.

3.1. Experimental Setup

The primary task that these experiments attempted to improve performance on was that of speaker verification, specifically that on VoxCeleb 1 [25] and Speakers In The Wild (SITW) core-core task [31]. Although there exist several metrics to evaluate verification performance, which are typically chosen depending on the desired behaviour of a system, the primary metric explored here was the equal error rate (EER), as that is the primary metric for evaluation on VoxCeleb 1.

The training data used for all experiments was the VoxCeleb 2 development set [32], which features 5994 unique speakers. This was augmented in the standard Kaldi[2] fashion with noise, music, babble and reverberation. The original x-vector architecture was used with very little modification, using Leaky ReLU instead of ReLU, with 30-dimensional MFCC features as inputs, and 512-dimensional embeddings. The main difference between this implementation and that of Snyder et al [2] was the use of the CosFace [33] angular penalty loss function instead of a traditional cross entropy loss. This classification transform also was applied directly to the embedding layer, unlike the original, which has an additional hidden layer between the embedding layer and the classification layer. This means that the simplified notation for the classifier C following from equation 3 is an accurate representation of our model. All pairs of embeddings were $L2$ normalized and scored using cosine distance.

A batch size of 500 was used, with each example having 350 frames. Each batch had the same number of unique speakers as examples. Models were trained for 120,000 iterations, using SGD with a learning rate of 0.2 and momentum 0.5. The learning rate was halved at 60,000, 80,000, 90,000, and 110,000 steps. For DropAdapt fine-tuning, the learning rate was chosen to be the same as it was at the end of training the original model, and all the enrolment utterances were used to calculate $\boldsymbol{p}_{\text{average}}$.

3.2. DropClass Experiments

Our initial experiments investigated favourable settings of P and D for DropClass, and the results are shown in Figure 3, where the number of classes to drop was fixed at $D = 5000$ and

[2]https://kaldi-asr.org/

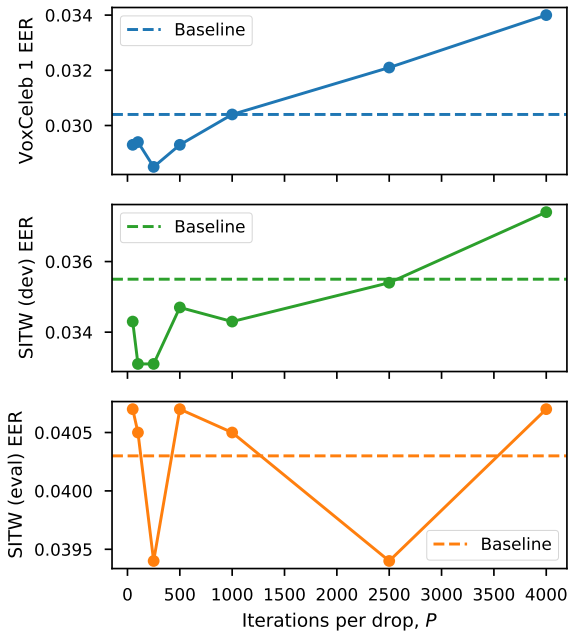

Figure 3: Comparison on the effect on EER of varying the number of iterations to run before re-selecting the class subset, fixed at dropping $5000/5994$ training classes each period.

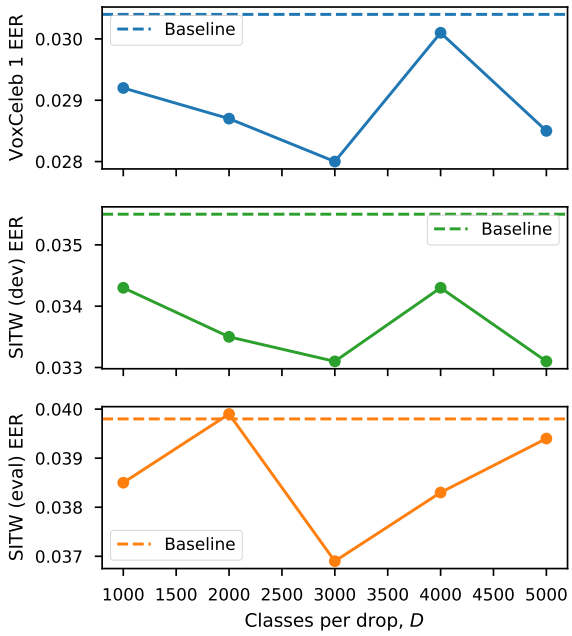

Figure 4: Comparison on the effect on EER of varying the number of classes to cycle out every 250 iterations.

the number of iterations P was varied between 50 and 4,000, the latter being slightly over 1 epoch's worth of data with the chosen batch size. It can be seen that improvements over the baseline are to be found more reliably at lower values of P, with consistent performance improvements when $P < 1000$ for both VoxCeleb and SITW (dev). This is perhaps unsurprising, as a motivating factor for this technique was to train a network on many different permutations for robustness on a variety of tasks, and thus with a training budget for each model of 120,000 iterations, this is not a large number of permutations throughout training. As the value for P increases, this may also increase the risk of incurring the phenomenon of catastrophic forgetting [34, 35], an issue in which networks trained on a new task begin to degrade on the task that they previously were trained on.

From the previous experiment, choosing the best performing value of $P = 250$ across each dataset, the number of classes to drop D was then varied from 1000 to 5000, shown in Figure 4. From this, we can see that for nearly all configurations of D, performance was improved on all datasets using DropClass over the baseline. Dropping approximately half the classes at D=3000 appeared to produce the best performance, although a more thorough exploration with different training data is likely required to ascertain if any heuristic exists for the selection of this value. However, from the previous experiments, it can be seen that for a suitably low value of P, DropClass can convey improvements over the baseline.

It is however important to note that a crucial component of this method is the use of the CosFace [33] angular penalty loss, with Table 1 showing a comparison of the effect that changing the loss function had on the improvement that DropClass produced on VoxCeleb. A more in-depth analysis on how each loss function changes with the permutations of each subset of classes is required.

3.3. DropAdapt Experiments

Table 2 displays the relative improvement in EER from utilizing the DropAdapt and DropAdapt-Combine method, using either the enrolment speakers from VoxCeleb 1 or SITW (dev) to choose which speakers to drop. The starting point was a standard classification trained baseline. Models were trained on a budget of 30,000 iterations, and one configuration for D and P was tested. Also compared were the following control experiments: The baseline but trained additionally for the same number of iterations as DropAdapt, Drop-Random, which drops random classes permanently, ignoring the $\boldsymbol{p}_{\text{average}}$ score, and Drop Only Data, which removes the low probability classes from the training data, but does not remove the relevant rows in the final weight matrix, bypassing the use of W^* in Equation 4.

Compared to the baseline and control experiments, both DropAdapt and DropAdapt-Combine show strong performance gains on VoxCeleb. The 2.63% EER on VoxCeleb is particularly impressive when compared to other works which use similar or larger network architectures and more training data and achieve $> 3\%$ EER [36, 7]. The improvements over the baseline on SITW however are more modest, with DropClass trained models and 'Drop Only Data' outperforming the DropAdapt models.

An interesting observation is the fact that dropping only the data improved performance on VoxCeleb, but not as much as the DropAdapt methods. As discussed in section 2.2, DropAdapt can be viewed as a form of corrective oversampling of targeted classes, with oversampling techniques having been shown to improve performance in imbalanced data scenarios [27, 28]. From this, we can see that for the within-domain data, some of the benefit of DropClass is gained from only fine-tuning via oversampling, but this benefit is increased further by also dropping the classes from the output layer. Conversely, for the out-of-domain SITW dataset, dropping only the classes from the

	EER (VoxCeleb)	
	Baseline	DropClass
Softmax	5.89%	6.25%
CosFace [33]	**3.04%**	**2.80%**
SphereFace [14]	3.92%	4.76%
ArcFace [13]	3.19%	3.08%
AdaCos [15]	3.24%	3.81%

Table 1: EER values on VoxCeleb 1 for using DropClass (P=250, D=3000) or not with different angular penalty loss functions, all with the paper recommended settings of hyper-parameters.

	EER	Rel Impr
Baseline (VoxCeleb)	3.04%	-
Baseline (More iterations)	3.06%	-0.7%
Drop Random (D=500, P=5000)	3.08%	-1.3%
Drop Only Data (D=500, P=5000)	2.86%	5.9%
DropAdapt (D=500, P=5000)	2.68%	11.8%
DropAdapt-C (D=500, P=5000)	**2.64%**	**13.2%**
Baseline (SITW)	3.55%	-
Baseline (More iterations)	3.61%	-1.7%
Drop-Random (D=500, P=5000)	3.73%	-5.1%
Drop Only Data (D=500, P=5000)	**3.31%**	**6.7%**
DropAdapt (D=500, P=5000)	3.47%	2.3%
DropAdapt-C (D=500, P=5000)	3.39%	4.5%

Table 2: Relative improvment in EER from using DropAdapt and DropAdapt-Combine (DropAdapt-C) on the VoxCeleb 1 and SITW datasets on a budget of $30,000$ iterations

data performed the best. We hypothesize that the reduced effectiveness of DropAdapt in this case may be due to the technique having to adapt to not only a new speaker distribution, but also a new domain. Further exploration combining DropAdapt with traditional domain adaptation techniques is left for future work.

In addition, more experimentation on the configurations of P and D could be explored, as it may be possible for example that the iterative dropping of classes is not necessary, and that the initial probability estimation is suitable. Furthermore, the most obvious extension left for future work is to use both DropClass and DropAdapt in conjunction, as both have been shown to provide performance increases in parallel.

Following up on the hypothesis presented in section 2.2 that the imbalanced distribution of $\boldsymbol{p}_{\text{average}}$ on the test set may be an indicator of train-test mismatch and thus incurring performance loss, Figure 5 shows the EER and the KL divergence ($D_{KL}(p||U)$) from the VoxCeleb test set $\boldsymbol{p}_{\text{average}}$ to the uniform distribution as the DropAdapt-Combine model is trained. As we can see from the figure, while the EER decreases, the distribution of $\boldsymbol{p}_{\text{average}}$ also gets closer to the uniform distribution. Whilst there appears to be a correlation, this is likely not a strongly linked pair of observations, in that we can easily break this relationship by training only the final affine matrix W and freezing the embedding extractor to provide more favourable class weightings for $\boldsymbol{p}_{\text{average}}$. However, in the case of DropAdapt, the decreasing $D_{KL}(p||U)$ may indicate that a favourable change in the extracted representations is occurring. This could be useful as a stopping criterion for cases in which adaptation data has no labels at all.

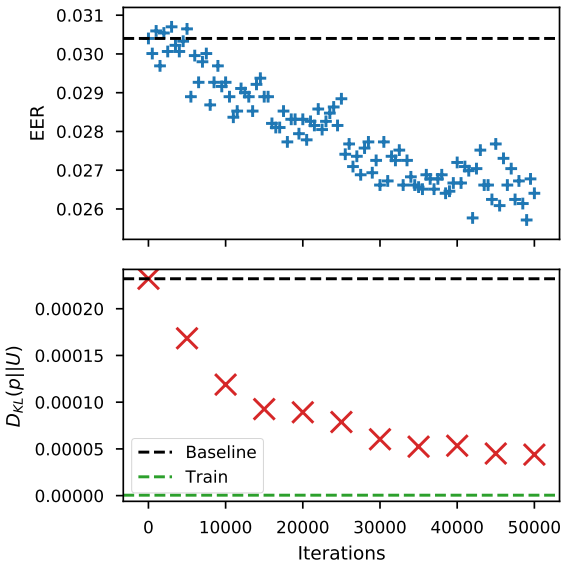

Figure 5: Plot of the evolving EER on VoxCeleb as classes are dropped with DropAdapt-Combine (P=5000, D=500) along with the KL divergence from $\boldsymbol{p}_{\text{average}}$ on the test set to the uniform distribution.

4. Conclusion

In this work we presented the DropClass and DropAdapt methods for training and fine-tuning deep speaker embeddings. Both methods are based around the notion of dropping classes from the final classification output layer while also withholding examples belonging to those same classes. Drawing inspiration from Dropout and meta-learning, DropClass is a method that drops classes randomly and periodically throughout training such that a model is trained on a large number of different classification objectives for subsets of the training classes as opposed to classifying on the full set of classes. We show that in conjunction with the CosFace [33] loss function, DropClass can improve verification performance on the VoxCeleb and SITW core-core tasks.

We present the mismatch in class imbalance between train and test as a potential reason for reduced performance in verification, and propose DropAdapt as a means of alleviating this. DropAdapt is a method which can adapt a trained model to a target dataset with unknown speakers in an unsupervised manner. This is achieved by calculating the average predicted probability of each training class with the adaptation data as input. From these predictions, the model is fine-tuned by dropping the low probability classes and training for more iterations, focusing on the classes which the model has predicted to be presented in the adaptation dataset. This is not unlike traditional oversampling techniques. Applying DropAdapt to VoxCeleb leads to a large improvement over the baseline, with DropAdapt also outperforming simply oversampling the same classes, suggesting it may be an effective strategy in adapting to a different class distribution than what was seen during training. We also show empirically that as the class distribution mismatch is corrected during DropAdapt, so too does the verification performance increase.

5. References

[1] David Snyder, Daniel Garcia-Romero, Daniel Povey, and Sanjeev Khudanpur, "Deep Neural Network Embeddings for Text-Independent Speaker Verification," in *Interspeech*, ISCA, 8 2017, pp. 999–1003, ISCA.

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

[3] Najim Dehak, Patrick J. Kenny, Réda Dehak, Pierre Dumouchel, and Pierre Ouellet, "Front end factor analysis for speaker verification," *IEEE Trans. Audio. Speech. Lang. Processing*, vol. 19, no. 4, pp. 788–798, 2011.

[4] Gregory Sell, David Snyder, Alan McCree, Daniel Garcia-Romero, Jesús Villalba, Matthew Maciejewski, Vimal Manohar, Najim Dehak, Daniel Povey, Shinji Watanabe, and Sanjeev Khudanpur, "Diarization is Hard: Some Experiences and Lessons Learned for the JHU Team in the Inaugural DIHARD Challenge," in *Interspeech*. 09 2018, pp. 2808–2812, ISCA.

[5] Mireia Diez, Lukas Burget, Shuai Wang, Johan Rohdin, and Honza Cernocký, "Bayesian HMM based x-vector clustering for Speaker Diarization," *Interspeech*, pp. 346–350, 2019.

[6] Jesús Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Leibny Paola García-Perera, Fred Richardson, Réda Dehak, Pedro A. Torres-Carrasquillo, and Najim Dehak, "State-of-the-art speaker recognition with neural network embeddings in NIST SRE18 and Speakers in the Wild evaluations," *Computer Speech and Language*, vol. 60, 2020.

[7] W. Xie, A. Nagrani, J. S. Chung, and A. Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *IEEE ICASSP*, May 2019, pp. 5791–5795.

[8] Yun Tang, Guo-Hong Ding, Jing Huang, Xiaodong He, and Bowen Zhou, "Deep speaker embedding learning with multi-level pooling for text-independent speaker verification," *IEEE ICASSP*, pp. 6116–6120, 2019.

[9] Elad Hoffer and Nir Ailon, "Deep metric learning using triplet network," in *Similarity-Based Pattern Recognition*, Cham, 2015, pp. 84–92, Springer International Publishing.

[10] Georg Heigold, Ignacio Moreno, Samy Bengio, and Noam Shazeer, "End-to-end text-dependent speaker verification," in *IEEE ICASSP*, 2016, vol. 2016-May, pp. 5115–5119.

[11] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized End-to-End Loss for Speaker Verification," *arXiv*, 2017.

[12] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, no. 7, pp. 926–930, 2018.

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

[14] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "SphereFace: Deep hypersphere embedding for face recognition," in *IEEE CVPR*, 2017, pp. 6738–6746.

[15] Xiao Zhang, Rui Zhao, Yu Qiao, Xiaogang Wang, and Hongsheng Li, "AdaCos: Adaptively scaling cosine logits for effectively learning deep face representations," in *IEEE CVPR*, 2019, pp. 10815–10824.

[16] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, pp. 1929–1958, 2014.

[17] Pierre Baldi and Peter Sadowski, "The dropout learning algorithm," *Artif. Intell.*, vol. 210, no. 1, pp. 78–122, 2014.

[18] David Warde-Farley, Ian J. Goodfellow, Aaron Courville, and Yoshua Bengio, "An empirical analysis of dropout in piecewise linear networks," *ICLR*, pp. 1–10, 2014.

[19] George E. Dahl, Tara N. Sainath, and Geoffrey E. Hinton, "Improving deep neural networks for LVCSR using rectified linear units and dropout," in *IEEE ICASSP*, 2013, pp. 8609–8613.

[20] Ehsan Variani, Xin Lei, Erik McDermott, Ignacio Lopez Moreno, and Javier Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *IEEE ICASSP*. 2014, pp. 4052–4056, Institute of Electrical and Electronics Engineers Inc.

[21] Yuxuan Wang, R. J. Skerry-Ryan, Daisy Stanton, Yonghui Wu, Ron J. Weiss, Navdeep Jaitly, Zongheng Yang, Ying Xiao, Zhifeng Chen, Samy Bengio, and Quoc Le, "Tacotron: Towards end-To-end speech synthesis," in *Interspeech*. 2017, vol. 2017-Augus, pp. 4006–4010, International Speech Communication Association.

[22] Chelsea Finn, Pieter Abbeel, and Sergey Levine, "Model-agnostic meta-learning for fast adaptation of deep networks," in *ICML*, 2017, vol. 3, pp. 1856–1868.

[23] Aniruddh Raghu, Maithra Raghu, Samy Bengio, and Oriol Vinyals, "Rapid Learning or Feature Reuse? Towards Understanding the Effectiveness of MAML," 2019.

[24] John H.L. Hansen and Taufiq Hasan, "Speaker recognition by machines and humans: A tutorial review," *IEEE Signal Process. Mag.*, vol. 32, no. 6, pp. 74–99, 2015.

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

[26] Mateusz Buda, Atsuto Maki, and Maciej A. Mazurowski, "A systematic study of the class imbalance problem in convolutional neural networks," *Neural Networks*, vol. 106, pp. 249–259, oct 2018.

[27] Chen Huang, Yining Li, Change Loy Chen, and Xiaoou Tang, "Deep imbalanced learning for face recognition and attribute prediction," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2019.

[28] Salman H. Khan, Munawar Hayat, Mohammed Bennamoun, Ferdous A. Sohel, and Roberto Togneri, "Cost-sensitive learning of deep feature representations from imbalanced data," *IEEE Trans. Neural Networks Learn. Syst.*, vol. 29, no. 8, pp. 3573–3587, 2018.

[29] V. Kaushal, R. Iyer, S. Kothawade, R. Mahadev, K. Doctor, and G. Ramakrishnan, "Learning from less data: A unified data subset selection and active learning framework for computer vision," in *IEEE WACV*, Jan 2019, pp. 1289–1299.

[30] V. M. Patel, R. Gopalan, R. Li, and R. Chellappa, "Visual domain adaptation: A survey of recent advances," *IEEE Signal Processing Magazine*, vol. 32, no. 3, pp. 53–69, May 2015.

[31] Mitchell McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson, "The speakers in the wild (SITW) speaker recognition database," in *Interspeech*, 2016, vol. 08-12-Sept, pp. 818–822.

[32] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *Interspeech*, , no. ii, pp. 1086–1090, 2018.

[33] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Liu, "Cosface: Large margin cosine loss for deep face recognition," *IEEE CVPR*, Jun 2018.

[34] Robert M. French, "Catastrophic forgetting in connectionist networks," in *Trends Cogn. Sci.*, 1999, vol. 3, pp. 128–135.

[35] Ian J. Goodfellow, Mehdi Mirza, Da Xiao, Aaron Courville, and Yoshua Bengio, "An empirical investigation of catastrophic forgetting in gradient-based neural networks," in *ICLR*, 2014.

[36] Koji Okabe, Takafumi Koshinaka, and Koichi Shinoda, "Attentive statistics pooling for deep speaker embedding," in *Interspeech*, 2018, vol. 2018-Septe, pp. 2252–2256.