OPTIMIZING NEURAL NETWORK EMBEDDINGS USING A PAIR-WISE LOSS FOR TEXT-INDEPENDENT SPEAKER VERIFICATION

Hira Dhamyal, Tianyan Zhou, Bhiksha Raj, Rita Singh

Language Technology Institute
Carnegie Mellon University
Pittsburgh, United States

ABSTRACT

This paper proposes a new loss function called the "quartet" loss for the better optimization of the neural networks for matching tasks. For such tasks, where neural network embeddings are the key component, the optimization of the network for better embeddings is critical. The embeddings are required to be class discriminative, resulting in minimal inter-class variation and maximal intra-class variation even for unseen classes for better generalization of the network. The quartet loss explicitly computes the distance metric between pairs of inputs and increases the gap between the similarity score distributions between the same class pairs and the different class pairs. We evaluate on the speaker verification task and demonstrate the performance of the loss on our proposed neural network.

Index Terms— quartet loss, embeddings, neural-networks, speaker verification

1. INTRODUCTION

In "matching" tasks like text-independent speaker verification, the objective is to determine if two given inputs belong to the same class or not without explicitly identifying the class of the inputs. A common approach is to extract class-discriminative *embeddings* – representative feature vectors – from the inputs, which are designed to have the property that the vectors derived from data instances of the same class cluster closer together than those from data from different classes. The match/mismatch decision may then be performed simply by comparing these vectors using an appropriate metric.

The accuracy of the system depends on the quality of the embeddings; how well they manage to cluster instances of a class, while separating instances from different classes. Traditional methods to derive class-discriminative embeddings, such as linear or non-linear discriminant analysis [1, 2, 3] require supervision through exact specification of the classes that must be distinguished. Such supervision is not available in the verification setting since the class of the instances being compared is not known, and may not have previously been encountered. The function that derives the embeddings must be learned without such explicit supervision.

The most successful methods to derive high-performance embeddings for verification tasks utilize neural networks. In the simplest framework, a network employing a linear-classifier output layer such as a multi-class logistic (a.k.a "softmax") is trained with supervision to classify between a large number of known classes. The representation derived by the network in its penultimate layer, prior to the application of the final classification layer, is treated as the embedding derived from the input [4, 5, 6, 7, 8, 9]. For a well-trained network with high accuracy, the embeddings for each of its classes will largely lie within convex regions that are separated from one another. The expectation is that if a network learns to classify a sufficiently large number of classes, the discriminativeness of the embeddings it derives will naturally generalize to other unseen classes as well, such that the embeddings of data from any class will fall within a (possibly) convex region that is distinct from regions occupied by other classes. Recent work using numerous variants of this approach show state of the art results for tasks like speaker verification, e.g. [10, 11].

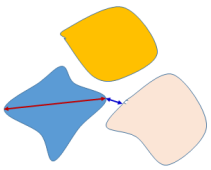

Fig. 1: The clusters represent speakers. While the clusters are well separable by a linear classifier, the distance between two farther instances within the blue cluster (red arrow) is much greater than the distance between two close instances from different clusters (blue arrow).

Regardless of their success, naïvely trained neural network feature extractors using the aforementioned approach demonstrably do not result in the best embeddings for verification tasks. The reason is that while the approach emphasizes the separation of the *distributions* of embeddings for

different classes, it does not directly optimize for the *direct comparison* of instances to determine if they belong to the same class. Thus, instances from adjacent classes may be closer than instances from the same class, even though the classes are separable, as illustrated in Figure 1. The distance between the farthest points within a class may be larger than that between vectors of different adjacent classes, in spite of their separability.

One identifiable reason for this behavior of the network is that the multi-class network is generally trained by minimizing a cross-entropy loss, which only aims to (linearly) separate the classes in the embedding space; any further separation of the classes is only a byproduct of the training process. In order to address this issue, modified loss functions such as center loss [12] and angular loss [13] have been proposed which additionally also try to explicitly increase the separation between classes. While effective at increasing inter-class spacing, these too are founded on the expectation that the statistical characteristics obtained for the embeddings of the known classes that provide supervision during training will generalize to novel, previously unseen classes in the test data, an expectation that does not always hold up.

An alternate, more appropriate approach is to directly optimize the network for the task at hand, i.e the comparison of inputs without explicit reference to their classes. The resulting networks may be expected to be more optimal than those trained indirectly through a classification task. Such optimization can be achieved by using a family of loss functions which we regard as the "pair-wise" loss functions. The category includes those functions which involve computing similarities between embeddings of pairs of inputs, with the objective of maximizing the similarity of same-class, or "matched" data instances, while minimizing that of "mismatched" instances that belong to different classes. Some examples of such losses are the triplet loss [14], contrastive loss [15, 16], and the quadruplet loss [17].

The ostensible objective of pair-wise losses generally is to maximize the difference between the similarities of matched and mismatched pairs of instances. We note that in reality, however, the actual aim is to minimize the overlap between the *distributions* of similarity scores under match and mismatch (as we explain in Section 2). Directly focusing on minimizing this distributional overlap may hence be expected to result in better generalization of the network than merely enhancing the separation of similarities of individual pairs of matched and mismatched instances. Most already-established loss functions of the "pair-wise" category, however, ignore or only make tangential reference to the true underlying objective, potentially compromising on performance.

In this paper, we propose a new pair-wise loss function called the "Quartet" loss, which embodies the above-mentioned objective. Quartet loss attempts to directly increase the gap between the distributions of similarity scores of matched and mismatched pairs of input. To formulate the optimization we explicitly consider the types of errors in the example problem of speaker verification, i.e., missed detections and false alarms. To make a decision for a given pair of speech recordings, a similarity score computed between their embeddings is compared to a threshold. The choice of the threshold represents a trade-off between missed detection and false alarm rates. Ideally, there would be a threshold where both false alarms and missed detections are zero. This happens when the distribution of similarity scores for matched recordings has no overlap with that of similarity scores for mismatched recordings. More generally, reduced verification errors may be obtained by minimizing the overlap between the distributions. This is the basis of our work.

We demonstrate the effectiveness of the proposed loss on the speaker verification task. We train a neural network for the task and evaluate the performance with and without the quartet loss introduced in the training process. We show that the proposed loss achieves the best verification performance, when compared to a number of baselines.

The rest of the paper is organized as follows: Section 2 outlines the basic problem statement. Section 3 describes the quartet loss function. Section 4 describes network architecture and settings. Finally, in sections 5 and 6, we present our experimental results and conclusions.

2. BACKGROUND AND RELATED WORK

The problem of any matching task can be formulated as follows: given two inputs $\mathbf{x}_1$ and $\mathbf{x}_2$, we must determine if both are from the same class or not. Formally, representing the event that they are from the same class by $\mathcal{H}_s$ and from different classes by $\mathcal{H}_d$, we must determine whether $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{H}_s$ or if $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{H}_d$.

To do so, we derive a fixed-length *feature* $f_{\mathbf{x}}$ from each input and compute a similarity score $S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})$ between them, typically either the log likelihood ratio under the hypotheses $\mathcal{H}_s$ and $\mathcal{H}_d$ [18], or the cosine similarity between $f_{\mathbf{x}_1}$ and $f_{\mathbf{x}_2}$ [19]. If this similarity exceeds a threshold θ, we decide that the two inputs "match," *i.e.,* they belong to the same class ($\mathcal{H}_s$), otherwise we declare a "mismatch" ($\mathcal{H}_d$).

The accuracy of this procedure is critically dependent on the features $f_{\mathbf{x}}$ derived from the inputs, which must be class discriminative. The key metric is the similarity computed between inputs. Figure 2 shows probability distributions of similarity scores under match and mismatch, $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_s)$ and $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_d)$ respectively. The overlap between the two represents the "region of confusion" – input pairs whose similarity score falls in this region are likely to be misclassified. The fraction of all tests that result in an erroneous outcome depends directly on the overlap area. Hence, to maximize verification accuracy, the *features* $f_{\mathbf{x}}$ derived from the inputs must be such that the two distributions are well separated.

Ideally, the function that extracts features from inputs

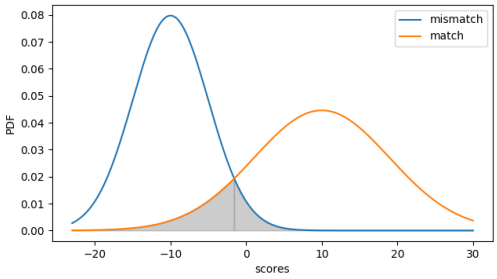

Fig. 2: Probability distributions of similarity scores under match and mismatch.

must aim to separate the distributions of similarity scores under match $\mathcal{H}_s$ and mismatch $\mathcal{H}_d$. Our objective is to develop a neural-network based feature extractor that explicitly attempts to maximize this separation. We do so through the *quartet* loss.

The proposed loss differs from previously proposed pairwise losses as we explain here. The contrastive loss, which is a margin-based loss that only considers pairs of instances at a time, attempts to "push" mismatched pairs ($(x_1, x_2) \in \mathcal{H}_d$) apart until a margin m, effectively ignoring the distribution beyond the margin (which may lie in the overlap region). In spite of its name, it does not explicitly contrast the distributions of similarity scores for matched and mismatched pairs. The triplet loss, which eponymously considers triplets of instances at a time (arguably improving on the contrastive loss in this regard), explicitly contrasts the similarity of the matched pairs to that of mismatched pairs. It maximizes the expected gap between the similarity of individual "anchor" inputs to matched (in class) and mismatched counterparts. While this does improve the separation between the similarity scores for matched and mismatched pairs, it still results in a relatively large intra-class variation, as observed in [20]. In both cases, it can be shown that the losses effectively quantify the gap between the *means* of $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_s)$ and $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_d)$, the distributions of similarity scores for matched and mismatched pairs, and minimizing them attempts to separate the means by the given margin. The same principle is also captured by the loss proposed in [21], which maximizes the gap between the average similarities of matched and mismatched pairs. In all cases, the overlap area of the distributions is only implicitly (and somewhat incidentally) minimized. The quadruplet loss proposed in [17] actually consider sets of *four* inputs at a time. In this regard it is most similar to our proposed quartet loss; however, like the other losses described above it only attempts to maximize the separation between the means of the overlapping regions of the distributions of similarity scores under match and mismatch.

In contrast to the above losses, (minimizing) the Quartet loss minimizes the overlap between $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_s)$ and $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_d)$, or alternately, maximizes the *gap* between the distributions of similarity scores under $\mathcal{H}_s$ and $\mathcal{H}_d$, thereby maximizing the gap between the similarity of matched pairs of inputs for a class and that of *any* mismatched pair of inputs for *any* two classes. In effect, it may be viewed as an even more conservative loss function than other pairwise loss functions.

The exact framework we use is elucidated in the next section.

3. FEATURE LEARNING THROUGH QUARTET LOSS

Let $F(\mathbf{x}; \Phi)$ with parameter Φ be the function that computes the feature $f_{\mathbf{x}}$ from an input $\mathbf{x}$, *i.e.,* $f_{\mathbf{x}} = F(\mathbf{x}; \Phi)$. Our objective is to learn $F(\mathbf{x}; \Phi)$ such that it minimizes the overlap between $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_s)$ and $P(S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})|\mathcal{H}_d)$. Let $S_s \sim P(S|\mathcal{H}_s)$ and $S_d \sim P(S|\mathcal{H}_d)$ represent draws from the probability distributions of similarity scores under match and mismatch. The overlap area is equal to $P(S_s < S_d)$. This area must be minimized.

Formally, let $(\mathbf{x}_1, \mathbf{x}_2) \in \mathcal{H}_s$ and $(\mathbf{y}_1, \mathbf{y}_2) \in \mathcal{H}_d$ be random pairs of matched and mismatched input pairs respectively. Let $S_X = S(f_{\mathbf{x}_1}, f_{\mathbf{x}_2})$ be the similarity computed between the matched pair $\mathbf{x}_1$ and $\mathbf{x}_2$, and similarly let $S_Y = S(f_{\mathbf{y}_1}, f_{\mathbf{y}_2})$ be the similarity computed between the mismatched pair $\mathbf{y}_1$ and $\mathbf{y}_2$. In order to optimize the feature extraction function $F(\mathbf{x}; \Phi)$ we must estimate Φ as

$$\hat{\Phi} = \arg\min_{\Phi} P(S_X < S_Y) \tag{1}$$

$P(S_X < S_Y)$ cannot be known, but an unbiased empirical estimator for it can be computed using the Wilcoxon Mann Whitney (WMW) statistic [22] on collections of randomly drawn pairs of matched and mismatched inputs. Representing the i^{th} randomly drawn matched pair as $X_i = (\mathbf{x}_1^i, \mathbf{x}_2^i)$ and the similarity score computed from it as S_{X_i}, and the j^{th} randomly drawn *mis*matched pair as $Y_j = (\mathbf{y}_1^j, \mathbf{y}_2^j)$ and the corresponding similarity score as S_{Y_j}, the estimate is

$$P(S_s < S_d) \approx \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathcal{I}(S_{X_i} < S_{Y_j})$$

where N and M correspond to the number of matched and mismatched pairs respectively and $\mathcal{I}()$ is the indicator function.

In principle, the estimator

$$\hat{\Phi} = \arg\min_{\Phi} \frac{1}{NM} \sum_{i=1}^{N} \sum_{j=1}^{M} \mathcal{I}(S_{X_i} < S_{Y_j}) \tag{2}$$

will give us a feature extractor $F(\mathbf{x}; \Phi)$ that minimizes the distribution overlap. In practice, the WMW statistic as given above only minimizes the *expected* value of $\mathcal{I}(S_{X_i} < S_{Y_j})$.

To maximize the generalization of the derived features, we must conservatively optimize the *worst* case, rather than the average case, *i.e.,* we must maximize the expected gap between the similarity score for a match and a *worst case* similarity score for mismatches.

We therefore define the following extreme value statistic

$$S^K_{Y,max} = max(S_i \sim P(S|\mathcal{H}_d),\ i = 1, \ldots, K)$$

where S^K_{max} is the maximum of K random draws from $P(S|\mathcal{H}_d)$. We can now define an empirical loss function:

$$L(\Phi) = \frac{1}{N}\sum_{i=1}^{N} \mathcal{I}(S_{X_i} < S^K_{Y_i,max}) \tag{3}$$

where $S^K_{Y_i,max}$ is the largest of the similarity scores obtained from a random draw of K mismatched pairs, that are specific to the i^{th} matched pair. In practice, the indicator function is not differentiable, so we approximate it as a sigmoid. This gives us our final loss function:

$$L(\Phi) = \frac{1}{N}\sum_{i=1}^{N} \sigma(S^K_{Y_i,max} - S_{X_i}) \tag{4}$$

$$\hat{\Phi} = \arg\min_{\Phi} L(\Phi) \tag{5}$$

Note that the sigmoid in Equation 4 can potentially be replaced by other monotonic functions such as a RELU, ELU or leaky RELU in this setting.

4. SYSTEM IMPLEMENTATION

We implement the feature extractor $F(\mathbf{x}; \Phi)$ for the task of speaker verification as a convolutional neural network that operates on spectrograms derived from the audio signal. The similarity score we optimize is the cosine similarity. We present the details below.

4.1. Parameterizing the speech signal

The input speech recordings are initially parametrized into a mel-frequency spectrogram, comprising sequences of log mel spectral vectors (melspec) [23]. The parametrization uses a bank of 63 mel-frequency filters, frame length of 25ms and a frame shift of 10ms. Non-speech regions in recordings are excised using an energy-based voice-activity detector [24]. The remaining vectors are mean normalized. During training each recording length is restricted to a randomly selected contiguous block of 16,383 frames, to conform to the logistical needs of batch processing. At testing time, entire recordings are used for the feature extraction.

Table 1: Deep neural network configurations (notation for convolutional layer: (channel, kernel, stride, padding)).

Setting	Detail
ResNet	Conv: (4, 3×3, 2, 0) Residual block: (4, 3×3, 1, 1) Residual block: (4, 3×3, 1, 1) Conv: (16, 3×3, 2, 0) Residual block: (16, 3×3, 1, 1) Residual block: (16, 3×3, 1, 1) Conv: (64, 3×3, 2, 0) Residual block: (64, 3×3, 1, 1) Residual block: (64, 3×3, 1, 1) Conv: (256, 3×3, 2, 0) Conv: (128, 3×3, 2, 0) temporal average pooling: entire feature map
Initialization	FC: classification layer Softmax Cross entropy loss
Fine tuning	quartet loss

4.2. Neural-network feature extractor

Our feature extractor is a convolutional neural network (CNN), with a residual network structure [25]. The network comprises a set of five "naïve" convolutional layers, augmented by several residual blocks. A residual block contains two convolutional layers in our experiments – this was found to be optimal. Table 1 shows the configuration detail of our system.

Result: (match-pairs, mismatch-pairs)
S_i is set of all recordings for speaker i
$S_i = \{x_1^{(i)}, \ldots\}$ *for* $i = 1, \ldots Z$;
S is the set of all speakers
$S = \{S_1, S_2 \cdots, S_Z\}$;
match-pairs $\leftarrow \{\}$;
mismatch-pairs $\leftarrow \{\}$;
for $l \leftarrow 1$ *to* P **do**
  $S_i \leftarrow$ sample-without-replacement(S);
  $x^{(i)} \leftarrow$ sample-with-replacement(S_i);
  match-pairs $\leftarrow$ match-pairs + $\{(x_1^{(i)}, x_2^{(i)})\}$;
  $S_p, S_q \leftarrow$ sample-with-replacement(S);
  $x^{(p)} \leftarrow$ sample-with-replacement(S_p);
  $x^{(q)} \leftarrow$ sample-with-replacement(S_q);
  mismatch-pairs $\leftarrow$ mismatch-pairs + $\{(x_1^{(p)}, x_2^{(q)})\}$;
end

Algorithm 1: Mini-batch creation

Result: loss
loss = 0;
match-pairs = $\{x_1^i, x_2^i\}_{i=1}^P$;
mismatch-pairs = $\{y_1^j, y_2^j\}_{i=1}^P$;
for $i \leftarrow 1$ *to* P **do**
  $S_{X_i} \leftarrow$ similarity-score(x_1^i, x_2^i);
  $S_{Y,max}^K \leftarrow$ max[similarity-score(y_1^i, y_2^i)] $j = 1 \dots K$;
  loss $\leftarrow$ loss + $\sigma(S_{Y,max}^K - S_{X_i})$;
end

Algorithm 2: Loss calculation. similarity-score used in our experiments is cosine similarity and σ for the best setup is the sigmoid function

4.3. Training the network

Initializing the network well is critical. We initialize the network by adding a softmax classification layer at the top to classify between all the speakers in the training set. This is trained for several epochs (70 in our experiments) to minimize the cross-entropy loss. Thereafter we remove the final classification layer and switch to the proposed quartet loss, and continue training for several additional epochs. We use stochastic gradient descent (SGD) with mini-batches of size 128. Momentum and weight decay are set to 0.9 and 0.0001 respectively which was found to be optimal.

4.4. Sampling for mini-batches

We organize the sampling of mini-batches as follows. Within each minibatch, we first choose P *matched* pairs of recordings, from P randomly chosen speakers. Subsequently, for each of the P pairs, we randomly draw K *mismatched* pairs of recordings. In our experiments we set $K = 40$. From these K recordings, we select the pair with the highest similarity score. The matched and mismatched scores together then feed into our loss. Note that the result is that for each entry into the loss function, *four* speech signals are considered. In our experiments, P was set to 32, resulting in 128 speech recordings per minibatch. Speakers and matched pairs are sampled without replacement. All other sampling is with replacement. Note that the computation only requires a single forward and backward propagation of individual training instances within each batch. Since this is the primary computational expense during training, the computational expense of optimizing the quartet loss is comparable to that of other losses within each minibatch. The pseudo-code for forming a minibatch is explained in Algorithm 1.

5. EXPERIMENTS

5.1. Data description

For training, we use the dataset published by the NIST speaker recognition evaluations (SRE). Our training set consists of several previous SRE sets from year 2004, 2005, 2006 and 2008, with a total of 36612 utterances and 3805 speakers.

Our evaluation set is SRE10 consisting of 7169 match pairs and 408950 many mismatch pairs, with a total of 416,119 trials. Our experimental results are reported on these trials.

5.2. Results

As noted in Section 4, the loss function in Equation 4 can employ a variety of approximations for the indicator function. We test different approximation functions in our loss including the Sigmoid, ELU and the LeakyReLU. All networks are identically initialized. While the actual loss is optimized using the cosine similarity metric, final verification employs PLDA [26, 27, 18, 28] scoring. The lowest EERs for the Sigmoid, ELU and LeakyRelu activations are 2.85%, 2.86% and 3.13% respectively. The EER for Sigmoid and ELU are the same on the last iteration, however based on the lowest EER achieved, we refer to the one with Sigmoid as our best system.

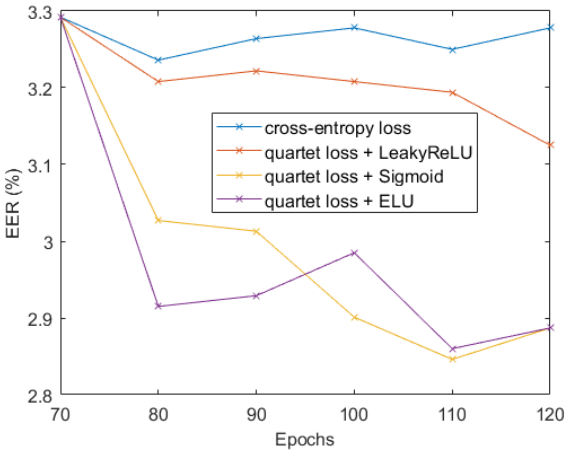

Fig. 3: EER for cross-entropy loss and quartet loss.

We also compare multiple approaches. Firstly, we continue training the initial CNN with the cross-entropy loss (with the final classification softmax layer) for an additional 50 epochs. This corresponds to the traditional approach of using CNNs for feature extraction [29]. In the comparators, we switch to optimizing using the proposed quartet loss (after removing the softmax layer) after the preliminary 70 initial iterations with the cross-entropy loss. Figure 3 demonstrates the EER trend for cross-entropy loss and quartet loss. We observe that cross entropy doesn't improve performance and

even seems to result in overfitting. By contrast, our loss decreases EER, especially for the ELU and Sigmoid approximations.

Secondly, we compare quartet loss with triplet loss. The network architecture and initialization are identical to those used for the quartet loss. The CNN is also optimized on the triplet loss using the best performing margin value. An EER of 3.32% is achieved with the margin of 0.2. As Figure 4 shows, the quartet loss, with its EER of 2.846% outperforms triplet loss.

Thirdly, we compare our approach with CLNet architecture proposed in [30], which was shown to achieve the best results for neural-network based feature extraction when used as an ensemble of three nets. As shown in Figure 4, our method greatly outperforms a single CLNet. We believe the gains will carry over to ensembles of quartet loss trained networks as well.

Finally, we implement the *x-vector* architecture as proposed in [10], using the recipe described therein. X-vectors have proven to be successful neural network embeddings for the speaker verification task on short audio files, and are among the current state-of-the-art architectures for the task. The performance shows that the embeddings trained using the quartet loss are able to achieve better on the verification task.

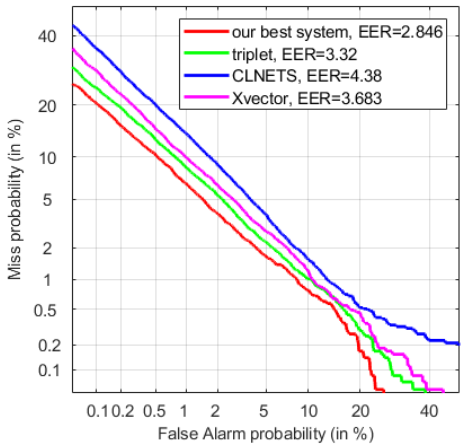

Fig. 4: DET curves of x-vectors, triplet loss(m=0.2), CLNets and our best system.

6. CONCLUSIONS

In this paper, we propose a quartet loss function to derive embeddings for verification tasks. The quartet loss explicitly maximizes the overlap between the similarity-score distributions for matched and mismatched pairs of inputs. In effect this accomplishes the task of increasing the inter-class variation and decreasing the intra-class variation of embeddings. We evaluate the loss on the speaker verification task. Experimental results indicate that our loss function achieves better results than other similar losses in terms of separating score distributions, resulting in improved verification, for minimal additional computational overhead. We note several avenues for future work, including the investigation of different sampling strategies and similarity metrics to optimize the network. [17] has inspired us to look at an adaptive margin approach which we also aim to incorporate in the loss. Specifically, within the verification task, we expect to fine tune the x-vector architecture [10] with quartet loss and exceed the reported performance.

7. REFERENCES

[1] K. Fukunaga, *Introduction to statistical pattern recognition*. Elsevier, 2013.

[2] T. Hastie and R. Tibshirani, "Discriminant analysis by gaussian mixtures," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 155–176, 1996.

[3] W. M. Campbell, D. E. Sturim, and D. A. Reynolds, "Support vector machines using gmm supervectors for speaker verification," *IEEE signal processing letters*, vol. 13, no. 5, pp. 308–311, 2006.

[4] E. Variani, X. Lei, E. McDermott, I. L. Moreno, and J. Gonzalez-Dominguez, "Deep neural networks for small footprint text-dependent speaker verification," in *2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2014, pp. 4052–4056.

[5] D. Snyder, D. Garcia-Romero, and D. Povey, "Time delay deep neural network-based universal background models for speaker recognition," in *Automatic Speech Recognition and Understanding*, 2016, pp. 92–97.

[6] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *INTERSPEECH*, 2017, pp. 999–1003.

[7] L. Li, Y. Chen, Y. Shi, Z. Tang, and D. Wang, "Deep speaker feature learning for text-independent speaker verification," in *INTERSPEECH*, 2017, pp. 1542–1546.

[8] J. Jorrin, P. Garcia, and L. Buera, "Dnn bottleneck features for speaker clustering," in *INTERSPEECH*, 2017, pp. 1024–1028.

[9] L. Li, Z. Tang, D. Wang, and T. F. Zheng, "Full-info training for deep speaker feature learning," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5369–5373.

[10] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 5329–5333.

[11] Y. Zhu, T. Ko, D. Snyder, B. Mak, and D. Povey, "Self-attentive speaker embeddings for text-independent speaker verification." in *Interspeech*, 2018, pp. 3573–3577.

[12] Y. Wen, K. Zhang, Z. Li, and Y. Qiao, "A discriminative feature learning approach for deep face recognition," in *ECCV*, 2016, p. 499515.

[13] W. Liu, Y. Wen, Z. Yu, M. Li, B. Raj, and L. Song, "Sphereface: Deep hypersphere embedding for face recognition," *CoRR*, vol. abs/1704.08063, 2017. [Online]. Available: http://arxiv.org/abs/1704.08063

[14] K. Q. Weinberger and L. K. Saul, "Distance metric learning for large margin nearest neighbor classification," *Journal of Machine Learning Research*, vol. 10, pp. 207–244, 2009.

[15] R. Hadsell, S. Chopra, and Y. LeCun, "Dimensionality reduction by learning an invariant mapping," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 2. IEEE, 2006, pp. 1735–1742.

[16] S. Chopra, R. Hadsell, Y. LeCun *et al.*, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR (1)*, 2005, pp. 539–546.

[17] W. Chen, X. Chen, J. Zhang, and K. Huang, "Beyond triplet loss: a deep quadruplet network for person re-identification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2017, pp. 403–412.

[18] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems," in *INTERSPEECH 2011, Conference of the International Speech Communication Association, Florence, Italy, August*, 2011, pp. 249–252.

[19] N. Dehak, R. Dehak, P. Kenny, N. Brummer, P. Ouellet, and P. Dumouchel, "Support vector machines versus fast scoring in the low-dimensional total variability space for speaker verification," in *Tenth Annual conference of the international speech communication association*, 2009, pp. 1559–1562.

[20] D. Cheng, Y. Gong, S. Zhou, J. Wang, and N. Zheng, "Person re-identification by multi-channel parts-based cnn with improved triplet loss function," in *Proceedings of the iEEE conference on computer vision and pattern recognition*, 2016, pp. 1335–1344.

[21] D. Snyder, P. Ghahremani, D. Povey, D. Garcia-Romero, Y. Carmiel, and S. Khudanpur, "Deep neural network-based speaker embeddings for end-to-end speaker verification," in *2016 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2016, pp. 165–170.

[22] H. B. Mann and D. R. Whitney, "On a test of whether one of two random variables is stochastically larger than the other," *Ann. Math. Statist.*, vol. 18, no. 1, pp. 50–60, 03 1947. [Online]. Available: https://doi.org/10.1214/aoms/1177730491

[23] S. B. Davis and P. Mermelstein, "Comparison of parametric representations for monosyllabic word recognition in continuously spoken sentences," in *Readings in speech recognition*. Elsevier, 1990, pp. 65–74.

[24] J. Ramirez, J. M. Górriz, and J. C. Segura, "Voice activity detection. fundamentals and speech recognition system robustness," in *Robust speech recognition and understanding*. InTech, 2007.

[25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," pp. 770–778, 2015.

[26] S. Prince and J. Elder, "Probabilistic linear discriminant analysis for inferences about identity," in *IEEE 11th International Conference on Computer Vision*, 2007, pp. 1–8.

[27] J. A. Villalba and N. Brummer, "Towards fully bayesian speaker recognition: Integrating out the between-speaker covariance," in *INTERSPEECH 2011, Conference of the International Speech Communication Association, Florence, Italy, August*, 2011, pp. 505–508.

[28] D. Garcia-Romero, X. Zhou, and C. Espy-Wilson, "Multicondition training of gaussian plda models in i-vector space for noise and reverberation robust speaker recognition," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2012, pp. 4257–4260.

[29] C. Li, X. Ma, B. Jiang, X. Li, X. Zhang, X. Liu, Y. Cao, A. Kannan, and Z. Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[30] Y. Wen, T. Zhou, R. Singh, and B. Raj, "A corrective learning approach for text-independent speaker verification," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2018.