CENTROID-BASED DEEP METRIC LEARNING FOR SPEAKER RECOGNITION

Jixuan Wang[1,2], *Kuan-Chieh Wang*[1,2], *Marc T. Law*[1,2], *Frank Rudzicz*[1,2,3,4], *Michael Brudno*[1,2,5]

[1]University of Toronto, Canada [2]Vector Institute, Canada [3]St Michaels Hospital, Canada
[4]Surgical Safety Technologies Inc, Canada [5] Hospital for Sick Children, Canada
{jixuan, wangkua1, law, frank, brudno}@cs.toronto.edu

ABSTRACT

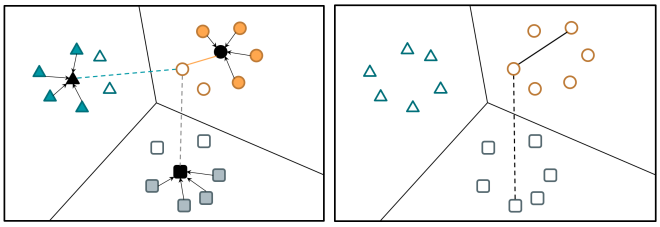
Speaker embedding models that utilize neural networks to map utterances to a space where distances reflect similarity between speakers have driven recent progress in the speaker recognition task. However, there is still a significant performance gap between recognizing speakers in the training set and unseen speakers. The latter case corresponds to the few-shot learning task, where a trained model is evaluated on unseen classes. Here, we optimize a speaker embedding model with *prototypical network loss* (PNL), a state-of-the-art approach for the few-shot image classification task. The resulting embedding model outperforms the state-of-the-art *triplet loss* based models in both speaker verification and identification tasks, for both seen and unseen speakers.

Index Terms— deep metric learning, triplet loss, sequence embedding, speaker recognition

1. INTRODUCTION

The development of embedding models to represent speech features in high-dimensional space has enabled elegant solutions to speaker recognition problems, including speaker verification (SV), and speaker identification (SI). Speaker verification systems verify whether or not a given utterance comes from some claimed speaker, while only having access to a handful of enrolled utterances (i.e. training examples). In the speaker identification task, a trained model is asked to classify among K speakers given small amount of enrollment speech. For both tasks, finding good representation of speech features is essential to recognition of the speaker, especially with small training sets: the variation in phrases needs to be normalized, while variation across speakers must be preserved.

Traditional pipelines that combine i-vector and probabilistic linear discriminant analysis (PLDA) separately train the feature extractor and the final classifier [1]. However, performance of i-vector systems drops for short speech utterances [2]. More robust feature extractors have been proposed, including replacing i-vectors with features extracted from deep networks [3, 4]. Most recent efforts have relied on optimizing an end-to-end deep model with the triplet loss (TL) [5, 6, 7, 8] and the generalized end-to-end (GE2E) loss [9] to build the speaker embeddings .



(a) prototypical network loss (b) triplet loss

Fig. 1: Comparison between the effect of prototypical network loss (PNL) and triplet loss (TL) on the embedding. Dashed lines represent distances encouraged to increase, while solid lines represent distances being decreased. **left**: For PNL, prototypes for different speakers, denoted by black nodes are computed as the mean of the support set (shaded) during training. **right**: For TL, a triplet consists of an anchor, positive, and negative samples, forming the (anchor, positive) and (anchor,negative) pairs. Depending on the sampling strategy not all triplets may be considered.

In this paper we propose the use of prototypical network loss (PNL) to optimize an end-to-end speaker embedding network. PNL was introduced for the few-shot image classification task [10] and is the state-of-the-art approach on several few-shot learning benchmarks. However, to the best of our knowledge, PNL has not been applied to speaker embedding or related problems. Here we show that for SV and SI tasks, a model trained with PNL outperforms an embedding network of the same architecture optimized with TL. We discuss why PNL is a better formulation for learning an embedding model and provide empirical observations as to why it might be easier to use in practice.

2. RELATED WORK

Speaker embedding networks: In traditional i-vector based methods for speaker embedding, a universal background model is first built. Then, a PLDA model is trained to measure the similarity of i-vectors. Replacing traditional i-vectors

The first two authors contributed equally to this work.

with speaker embedding models based on deep neural networks has lead to improvement in SV [4, 3]. Nonetheless, a PLDA classifier is still needed to compare the similarity of embeddings. More recently, end-to-end training of an embedding network that makes decision by comparing distance in the embedding to a cross-validated threshold outperformed traditional methods. For detailed comparison between embedding networks and i-vector based methods, we refer the reader to [6, 4, 3]. Building on top of these studies, our work focuses on the comparison between two different approaches for deep metric learning (TL [5, 6, 7, 8] and PNL [10]) for end-to-end speaker embedding models.

Deep metric learning: End-to-end speaker embedding models can be seen as a form of deep metric learning, which has been widely studied in the machine learning literature. Early examples of metric learning with neural networks include signature [11] and face verification [12]. Both compare pairs of examples with standard similarity functions (e.g. cosine or Euclidean distance) at the final embedding layer of a *siamese* architecture. More complex loss functions involving *triplets* {*anchor*, *positive*, *negative*} were later proposed [13], and shown to perform well on face verification [14].

Few-shot learning: Motivated by the fact that humans can learn new concepts from only a handful of examples, researchers have proposed the challenging task of "few-shot learning" [15, 16]. The test-time task is to classify examples among K new classes (i.e. unseen during training) while only being given a handful of labeled examples from these new classes. The same consideration arises naturally for speaker recognition tasks, as speakers encountered during test time may be different. It is essential to build profiles for previously unseen speakers with limited data. In contrast to previous applications of PNL [10], we use it in conjunction with a recurrent neural network for sequential data (i.e. speech) rather than a convolutional network for images.

3. OPTIMIZATION SCHEMES & MODEL

We now explain the standard triplet loss scheme and compare it to PNL [10]. We then describe the model we optimize using the two schemes for speaker embedding.

3.1. Triplet loss

For triplet-based models, we denote by $S' = \{\mathbf{x}_1, \cdots, \mathbf{x}_{N'}\}$ the examples in one mini-batch of size N', where $\mathbf{x}_i$ is a sequence of speech features. These models sample triplets, which consist of an anchor $\mathbf{x}_a$, a positive sample $\mathbf{x}_p$ with the same speaker label, and a negative sample $\mathbf{x}_n$ with a different speaker label. For each triplet $\tau = (\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n)$, the triplet loss is formulated as:

$$L(\tau) = max(0, d_{a,p} - d_{a,n} + \alpha) \tag{1}$$

where $d_{a,b} = d(f(\mathbf{x}_a), f(\mathbf{x}_b))$, d and f are the distance function (e.g. cosine or squared Euclidean distance) and speaker embedding model, respectively, and $\alpha > 0$ is a margin. Minimizing $L(\mathbf{x}_a, \mathbf{x}_p, \mathbf{x}_n)$ learns representations so that the similar pair $(\mathbf{x}_a, \mathbf{x}_p)$ has smaller distance than the dissimilar pair $(\mathbf{x}_a, \mathbf{x}_n)$, adjusted by the margin. It is worth noting that the triplet loss does not minimize distances between similar pairs (i.e., when $d_{a,p} - d_{a,n} + \alpha < 0$); it only tries to preserve some order between distances. Finally, the loss for a mini-batch is $J_{TL} = \sum_{\tau \in \mathrm{T}} L(\tau)$ where T is the set all of possible triplets in the mini-batch. In the context of very large datasets, creating all the possible triplets (referred as the naïve strategy) is computationally expensive; different triplet sampling strategies have been proposed to ensure fast convergence while avoiding degenerate solutions. For instance, the semi-hard mining strategy [14] samples only one hard negative pair for each positive pair. A triplet is "hard" if $d_{a,p} - d_{a,n} + \alpha > 0$.

3.2. Prototypical Networks Loss

Prototypical Networks [10] train a neural network episodically; each episode is composed of one mini-batch containing K categories (here, speakers). The mini-batch contains a support set called S and a query set called Q. In our case, the support set $S = \{(\mathbf{x}_i, y_i)\}_{i=1}^{N}$ represents each example as a sequence of speech feature vectors $\mathbf{x}_i$ with corresponding speaker label $y_i \in \{1, \cdots, K\}$. We denote $S_k \subseteq S$ as the set of examples in S of speaker k.

The prototype (or centroid) of each class $\mathbf{c}_k \in \mathbb{R}^M$ is calculated as the mean of embeddings in the support set:

$$\mathbf{c}_k = \frac{1}{|S_k|} \sum_{(\mathbf{x}_i, y_i) \in S_k} f(\mathbf{x}_i) \tag{2}$$

where f is the speaker embedding model which maps sequences of speech features into the M-dimensional embedding space (see details in Section 3.3).

During training, each query example $\{(\mathbf{x}_j, y_j)\} \in Q$ is classified against K speakers based on a softmax over distances to each speaker prototypes:

$$p(y = y_j | \mathbf{x}_j) = \frac{\exp\left(-d(f(\mathbf{x}_j), \mathbf{c}_{y_j})\right)}{\Sigma_{k'} \exp\left(-d(f(\mathbf{x}_j), \mathbf{c}_{k'})\right)} \tag{3}$$

where d is the distance function. The loss function for each mini-batch is $J_{PNL} = \sum_{\{(\mathbf{x}_j, y_j)\} \in Q} - \log p(y = y_j | \mathbf{x}_j)$

3.3. Speech sequence embedding model

In terms of network architecture, we use the same speech sequence embedding model as TristouNet [7]. We use the same model architecture when optimizing with each of the two losses above. As shown in Fig. 2, the sequence of Mel-frequency cepstral coefficients (MFCC) features are collected. They are then fed into bidirectional LSTM [17]. The outputs from the forward and backward LSTMs are first average-pooled over time, concatenated, then processed by a fully connected layer and a normalization layer.

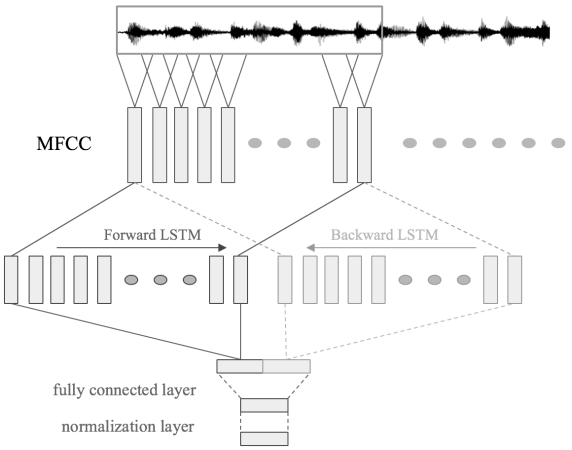


Fig. 2: Speech sequence embedding model

4. EXPERIMENT

In this section, we introduce the experiments conducted for evaluation and comparison of different models.

4.1. Datasets and implementation details

We use the VCTK corpus [18] and a subset of VoxCeleb2 dataset [8]. VCTK corpus contains clearly read speech, while VoxCeleb2 has more background noise and overlapping speech. Speech data of the first 90 speakers in VCTK corpus was divided into training, validation and test sets. Data of the remaining 18 speakers was used as an "unseen" set to evaluate the generalizability of the method. For VoxCeleb2, we selected a subset containing 101 speakers (we refer to this subset as VoxCeleb2 dataset for conciseness) and use data of 71 speakers for training and validation, while the other 30 speakers are used as the "unseen" set.

We use the default implementation of TristouNet for feature extraction and speech sequence embedding models. In the following, we refer to the models by the loss used during optimization (i.e. PNL vs. TL) as the architecture of the embedding network is fixed.

Feature extraction We extracted 19-dimensional MFCCs, their first and second derivatives, along with the first and second derivatives of energy in a 25ms window every 10ms using the *pyannote* multimedia processing toolkit[1] and Yaafe toolkit [19]. This results in 59-dimensional acoustic features. We use 2-second segments for both PNL and TL models to ensure comparability of results and to test performance on shorter utterances.

Training We use PyTorch [20] for implementation of both losses[2]. The output dimension is 16, and the $tanh$ activation function is used for the fully connected layer. Squared Euclidean distance is used as the distance function. The margin α used in our implementation of TL is 0.2 following [7]. For both models, the Adam optimizer [21] is used with 10^{-3} learning rate. Both models are trained for 100 epochs.

For each mini-batch, we randomly choose 15 speakers without replacement on VCTK (10 speakers on VoxCeleb2). We enforce equal mini-batch size between the two formulations (i.e., $|S| + |Q| = N'$). TL models are trained with Euclidian distance or cosine distance using naïve or semi-hard strategy. PNL models are trained with different size of support and query sets. To avoid confusion in the following, "TL (s, d)" denotes sampling strategy s with distance metric d, and "PNL (x_y, d)" denotes PNL using x-shot with query set of size y training episodes.

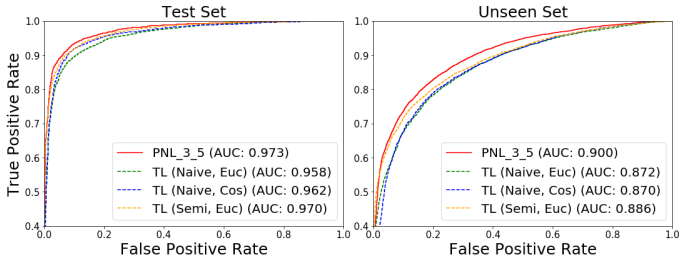


Fig. 3: "*same/different*" experiments on test and unseen sets.

4.2. "*Same/Different*" experiments

As with TristouNet [7], we first conduct "*same/different*" experiments on the VCTK corpus. The same number of positive (same speaker) and negative (different speakers) pairs are randomly selected. For each pair of segments, we calculate the distance between their embeddings and compare it with a threshold to predict whether they are from the same or different speakers.

We report the receiver operating characteristic (ROC) curve for each model, shown in Fig. 3. All models perform comparably well on seen data set. However, on the unseen data PNL model outperforms TL models.

4.3. Speaker Identification

To evaluate SI performance, we simulate each test *task* with a batch of K speakers, each with S enrollment samples, and Q query samples. Classification of a query is done by finding the closest prototype based on some metric d. Results are shown in Table 1, from which one can observe that the 3-shot PNL-based model outperforms the TL (Semi, Euc) model on all tasks, especially on the more challenging 18-way SI task (6% and 19% relative improvement on test and unseen data set, respectively). Interestingly, the one-shot PNL based model performs better than triplet loss based model with naïve sampling strategy while it "sees" many fewer positive and negative pairs for each batch. It also performs nearly as well as TL that uses the more complicated semi-hard sampling strategy.

[1] http://pyannote.github.io/

[2] Our implementation of PNL is based on that of the original authors: https://github.com/jakesnell/prototypical-networks [10]

Table 1: SI accuracy on test and unseen sets of VCTK. Under the Model column are the training configurations. The top row 'S:N_S, Q:N_Q, K-way' denotes the *task* configurations.

Model	S: 5, Q: 5, 6-way		S: 10, Q: 10, 18-way	
	Test	Unseen	Test	Unseen
TL (Naive, Euc)	91.96%	77.80%	81.25%	56.37%
TL (Naive, Cos)	92.37%	77.69%	83.51%	58.51%
TL (Semi, Euc)	93.33%	79.69%	85.38%	58.49%
TL (Semi, Cos)	92.13%	73.94%	83.99%	52.97%
PNL (1_5, Euc)	93.24%	77.90%	83.71%	56.52%
PNL (3_5, Euc)	**95.63%**	84.81%	**90.53%**	**69.64%**
PNL (5_5, Euc)	95.47%	83.69%	89.85%	68.55%
PNL (10_10, Euc)	94.38%	**85.00%**	88.43%	66.63%

Table 2: SI accuracy on VoxCeleb2.

Model	15-way			
	S: 10, Q: -		S: 30, Q: -	
	Test	Unseen	Test	Unseen
TL (Semi, Cos)	74.74%	53.92%	75.18%	59.61%
TL (Semi, Euc)	71.78%	51.74%	72.02%	56.79%
PNL (5_5, Euc)	**78.38%**	**59.44%**	**79.23%**	**66.63%**

Table 3: EER of SV on both data sets. "2s" (2 seconds) refers the duration of speech we used for enrollment.

(a) EER on VCTK

Model	Test	Unseen	
	60s	60s	10s
TL (Semi, Cos)	5.43(±0.16)	13.87(±0.37)	16.19(±0.86)
TL (Semi, Euc)	5.05(±0.09)	12.26(±0.69)	13.44(±0.91)
PNL (5_5, Euc)	**4.08(±0.13)**	**10.77(±0.58)**	**12.00(±0.76)**

(b) EER on VoxCeleb2

Model	Test	Unseen	
	60s	60s	10s
TL (Semi, Cos)	9.23(±0.13)	14.62(±0.35)	16.93(±0.45)
TL (Semi, Euc)	9.90(±0.11)	15.92(±0.32)	17.61(±0.51)
PNL (5_5, Euc)	**8.29(±0.12)**	**13.68(±0.26)**	**15.67(±0.56)**

4.4. Speaker verification

To evaluate SV performance, we randomly select some speech segments for enrollment. Then, 200 (resp. 100) segments of each speaker are selected as positive samples on VoxCeleb2 (resp. VCTK). Equal number of negative samples are selected from different speakers. During enrollment phase, speaker prototypes are computed from the enrollment set. For verification, the decision is made by comparing the distance between the embedding of the query segment and the speaker prototype to a threshold. Performance is evaluated by equal error rate (EER). Results are shown in Table 3.

Results obtained by repeating the experiments 10 times (i.e. mean and standard deviation) on VCTK and VoxCeleb2 are shown in Table 3a and Table 3b, respectively. EER of PNL model is significantly lower than that of TL models ($p \ll 0.001$ using t-test) across all tasks. As expected, both models perform reasonably well on test set of VCTK, while a little worse on test set of VoxCeleb2. For unseen set, EERs of all models decrease for longer duration of speech data for enrollment. Although TL (Semi, Cos) outperforms TL (Semi, Euc) on more noisy data set, PNL still achieves the lowest EER.

4.5. Analysis

The fact that generalization improves as the number of data points per category increases for the PNL model may be explained by the fact that PNL is a specific (supervised) formulation of clustering with Bregman divergences [22]. The prototype of each category approximates the point that minimizes the loss in Bregman information [22] for that category. Bregman information is related to Shannon's rate distortion theory, it corresponds to the optimal distortion-rate to encode a category when the distortion is measured by d (i.e., the squared Euclidean distance). The point that minimizes the loss in Bregman information [22] for the category is the mean vector of all the examples that belongs to the category. Therefore, the larger the number of 'shots', the better the approximation of the mean vector of the examples in the category. However, a larger number of shots does not necessarily lead to better performance as discussed in [10]. Statistical guarantees of (a generalization of) PNL are studied in [23]. In practice, PNL is quite robust to the choice of number of shots. We find that anywhere between 3 to 10 shots work well.

The reason why SI accuracy of TL models drops dramatically for experiments with more "ways" might be due to limitation of TL, which has been extensively studied in the literature in different contexts [24, 25]. One main limitation is that TL does not necessarily group each category into a single cluster even when the global optimum is reached [24].

In our experience, PNL is practically easier to use than TL. PNL is not dependent on a triplet sampling strategy, that can impact performance, and do not require the margin parameter α. PNL models are also $\sim$3x faster to train (wallclock time) than TL models with same mini-batch size because TL requires more pairwise comparisons for batches of same size.

5. CONCLUSION

We have proposed a prototypical network loss-based speaker embedding model, and compared it with the popular triplet loss-based models. With identical speech sequence embedding architectures, PNL outperforms the triplet loss when speakers are seen during training, and by an even larger margin on held-out, unseen speakers for both speaker identification and speaker verification tasks. We also illustrate some of the practical advantages of PNL models over TL. In the future, we would like to explore better architectures of speech sequence embedding models and integrate the few-shot learning based speaker embedding model into a speaker diarization pipeline.

6. REFERENCES

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

[2] Achintya Kumar Sarkar, Driss Matrouf, Pierre Michel Bousquet, and Jean-François Bonastre, "Study of the effect of i-vector modeling on short and mismatch utterance duration for speaker verification," in *Interspeech*, 2012.

[3] David Snyder, Daniel Garcia-Romero, Gregory Sell, Daniel Povey, and Sanjeev Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," *Submitted to ICASSP*, 2018.

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

[5] Chao Li, Xiaokong Ma, Bing Jiang, Xiangang Li, Xuewei Zhang, Xiao Liu, Ying Cao, Ajay Kannan, and Zhenyao Zhu, "Deep speaker: an end-to-end neural speaker embedding system," *arXiv preprint arXiv:1705.02304*, 2017.

[6] Chunlei Zhang and Kazuhito Koishida, "End-to-end text-independent speaker verification with triplet loss on short utterances," in *Proc. of Interspeech*, 2017.

[7] Hervé Bredin, "Tristounet: triplet loss for speaker turn embedding," in *ICASSP*. IEEE, 2017, pp. 5430–5434.

[8] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[9] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized end-to-end loss for speaker verification," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 4879–4883.

[10] Jake Snell, Kevin Swersky, and Richard Zemel, "Prototypical networks for few-shot learning," in *NIPS*, 2017.

[11] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah, "Signature verification using a" siamese" time delay neural network," in *NIPS*, 1994.

[12] Sumit Chopra, Raia Hadsell, and Yann LeCun, "Learning a similarity metric discriminatively, with application to face verification," in *CVPR*, 2005.

[13] Matthew Schultz and Thorsten Joachims, "Learning a distance metric from relative comparisons," in *NIPS*, 2004.

[14] Florian Schroff, Dmitry Kalenichenko, and James Philbin, "Facenet: A unified embedding for face recognition and clustering," in *CVPR*, 2015.

[15] Li Fei-Fei, Rob Fergus, and Pietro Perona, "One-shot learning of object categories," *IEEE TPAMI*, vol. 28, no. 4, pp. 594–611, 2006.

[16] Brenden M Lake, Ruslan Salakhutdinov, and Joshua B Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015.

[17] Sepp Hochreiter and Jürgen Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[18] Christophe Veaux, Junichi Yamagishi, Kirsten MacDonald, et al., "Superseded-cstr vctk corpus: English multi-speaker corpus for cstr voice cloning toolkit," 2016.

[19] Benoit Mathieu, Slim Essid, Thomas Fillon, Jacques Prado, and Gaël Richard, "Yaafe, an easy to use and efficient audio feature extraction software.," in *ISMIR*, 2010, pp. 441–446.

[20] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer, "Automatic differentiation in pytorch," in *NIPS-W*, 2017.

[21] Diederik P Kingma and Jimmy Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[22] Arindam Banerjee, Srujana Merugu, Inderjit S Dhillon, and Joydeep Ghosh, "Clustering with bregman divergences," *JMLR*, vol. 6, no. Oct, pp. 1705–1749, 2005.

[23] Marc T Law, Jake Snell, Amir massoud Farahmand, Raquel Urtasun, and Richard S Zemel, "Dimensionality reduction for representing the knowledge of probabilistic models," in *ICLR*, 2019.

[24] Hyun Oh Song, Stefanie Jegelka, Vivek Rathod, and Kevin Murphy, "Deep metric learning via facility location," in *CVPR*, 2017.

[25] Marc T. Law, Nicolas Thome, and Matthieu Cord, "Learning a distance metric from relative comparisons between quadruplets of images," *IJCV*, pp. 65–94, 2017.