UTTERANCE-LEVEL AGGREGATION FOR SPEAKER RECOGNITION IN THE WILD

Weidi Xie[1]*, Arsha Nagrani*[1]*, Joon Son Chung*[1,2] *and Andrew Zisserman*[1]

[1]Visual Geometry Group, Department of Engineering Science, University of Oxford, UK
[2]Naver Corporation, South Korea
{weidi, arsha, joon, az}@robots.ox.ac.uk

ABSTRACT

The objective of this paper is speaker recognition 'in the wild' – where utterances may be of variable length and also contain irrelevant signals. Crucial elements in the design of deep networks for this task are the type of trunk (frame level) network, and the method of temporal aggregation. We propose a powerful speaker recognition deep network, using a 'thin-ResNet' trunk architecture, and a dictionary-based NetVLAD or GhostVLAD layer to aggregate features across time, that can be trained end-to-end. We show that our network achieves state of the art performance by a significant margin on the `VoxCeleb1` test set for speaker recognition, whilst requiring fewer parameters than previous methods. We also investigate the effect of utterance length on performance, and conclude that for 'in the wild' data, a longer length is beneficial.

Index Terms— speaker recognition, speaker verification, speech, deep learning, CNNs

1. INTRODUCTION

Speaker recognition 'in the wild' has received an increasing amount of interest recently due to the availability of free large-scale datasets [1, 2, 3], and the easy accessibility of deep learning frameworks [4, 5, 6]. For speaker recognition, the goal is to condense information into a *single* utterance-level representation, unlike speech recognition where frame-level representations are desired. Obtaining a good utterance level representation becomes particularly important for speech obtained under noisy and unconstrained conditions, where irrelevant parts of the signal must be filtered out. Therefore, a key area of research in deep learning for speaker recognition is to investigate how to effectively aggregate frame-level characteristics into utterance-level speaker representations.

Earlier deep neural network (DNN) based speaker recognition systems have naïvely used pooling methods that have been successful for visual recognition tasks, such as average pooling [2, 3, 7, 8] or fully connected layers [9, 10] to condense frame-level information into utterance-level representations. Although such methods serve the purpose of aggregating frame-level information into a single representation whilst still allowing back-propagation, the aggregation is not content dependent, so they are not able to consider which parts of the input signal contain the most relevant information.

On the other hand, traditional methods for speaker and language identification such as *i-vector* systems have explored the use of statistical or dictionary-based methods for aggregation. A number of recent works have proposed to bring similar methods to deep speaker recognition [11, 12, 13, 14, 15] (described in Sec. 1.1). Based on these works, we propose to marry the best of both Convolutional Neural Networks (henceforth, CNNs) and a dictionary-based NetVLAD [16] layer, where the former is known for capturing local patterns, and the latter can be discriminatively trained for aggregating information into a fixed-sized descriptor from an input of arbitrary size, such that the final representation of the utterance is unaffected by irrelevant information.

We make the following contributions: (i) We propose a powerful speaker recognition deep network, based on a NetVLAD [16] or GhostVLAD [17] layer that is used to aggregate 'thin-ResNet' architecture frame features; (ii) The entire network is trained end-to-end using a large margin softmax loss on the large-scale `VoxCeleb2` [3] dataset, and achieves a significant improvement over the current state-of-the-art verification performance on `VoxCeleb1`, despite using fewer parameters than the current state-of-the-art architectures [3, 13]; and, (iii) We analyse the effect of input segment length on performance, and conclude that for 'in the wild' sequences having longer utterances (4s or more) is a significant improvement over shorter segments.

1.1. Related works

End-to-end deep learning based systems for speaker recognition usually follow a similar three-stage pipeline: (i) frame level feature extraction using a deep neural network (DNN); (ii) temporal aggregation of frame level features; and (iii) optimisation of a classification loss. In the following, we review the three components in turn.

The trunk DNN architecture used is often either a 2D CNN with convolutions in both the time and frequency domain [2, 3, 15, 18, 19, 20], or a 1D CNN with convolutions applied only to the time domain [11, 12, 13, 21]. A number

`http://www.robots.ox.ac.uk/~vgg/research/speakerID`

of papers [8, 22] have also used LSTM-based front-end architectures.

The output from the feature extractor is a variable length feature vector, dependant on the length of the input utterance. Average pooling layers have been used in [2, 3, 8] to aggregate frame-level feature vectors to obtain a fixed length utterance-level embedding. [11] introduces an extension of the method in which the standard deviation is used as well as the mean – this method is termed *statistical pooling*, and used by [12, 21]. Unlike these methods which ingest information from all frames with equal weighting, [20, 22] have employed attention models to assign weight to the more discriminative frames. [13] combines the attention models and the statistical model to propose *attentive statistics pooling* – this method holds the current state-of-the-art performance on the `VoxCeleb1` dataset. The final pooling strategy of interest is the Learnable Dictionary Encoding (LDE) proposed by [14, 15]. This method is closely based on the NetVLAD layer [16] designed for image retrieval.

Typically, such systems are trained end-to-end for classification with a softmax loss [13] or one of its variants, such as the angular softmax [15]. In some cases, the network is further trained for verification using the contrastive loss [2, 3, 23] or other metric learning losses such as the triplet loss [7]. Similarity metrics like the cosine similarity or PLDA are often adopted to generate a final pairwise score.

2. METHODS

For speaker recognition, the ideal model should have the following properties: (1) It should ingest arbitrary time lengths as input, and produce a fixed-length utterance-level descriptor. (2) The output descriptor should be *compact* (i.e. low-dimensional), requiring little memory, to facilitate efficient storage and retrieval. (3) The output descriptor should also be *discriminative*, such that the distance between descriptors of different speakers is larger than those of the same speaker.

To satisfy all the aforementioned properties, we use a modified ResNet in a fully convolutional way to encode input 2D spectrograms, followed by a NetVLAD/GhostVLAD layer for feature aggregation along the temporal axis. This produces a fixed-length output descriptor. Intuitively, the VLAD layer can be thought of as trainable discriminative clustering: every frame-level descriptor will be softly assigned to different clusters, and residuals are encoded as the output features. To allow efficient verification (i.e. low memory, fast similarity computation), we further add a fully connected layer for dimensionality reduction. Discriminative representations emerge because the entire network is trained end-to-end for speaker identification. The network is shown in Figure 1 and described in more detail in the following paragraphs.

Feature Extraction. The first stage involves feature extraction from input spectrograms. While any network can be used in our learning framework, we opt for a modified ResNet with 34 layers. Compared to the standard ResNet used before by [3], we cut down the number of channels in each residual block, making it a *thin* ResNet-34 (Table 1).

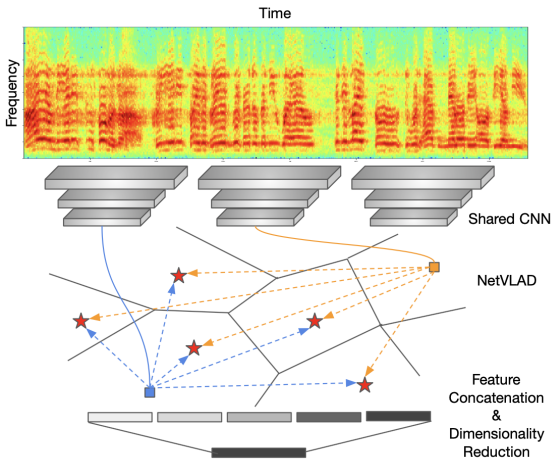

Fig. 1: Network architecture. It consists of two parts: *feature extraction*, where a shared CNN is used to encode the spectrogram and extract frame-level features, and *aggregation*, which aggregates all the local descriptors into a single compact representation of arbitrary length.

Module	Input Spectrogram ($257 \times T \times 1$)	Output Size
Thin ResNet	conv2d, 7×7, 64	$257 \times T \times 64$
	max pool, 2×2, stride (2, 2)	$128 \times T/2 \times 64$
	$\begin{bmatrix} \text{conv}, 1 \times 1, 48 \\ \text{conv}, 3 \times 3, 48 \\ \text{conv}, 1 \times 1, 96 \end{bmatrix} \times 2$	$128 \times T/2 \times 96$
	$\begin{bmatrix} \text{conv}, 1 \times 1, 96 \\ \text{conv}, 3 \times 3, 96 \\ \text{conv}, 1 \times 1, 128 \end{bmatrix} \times 3$	$64 \times T/4 \times 128$
	$\begin{bmatrix} \text{conv}, 1 \times 1, 128 \\ \text{conv}, 3 \times 3, 128 \\ \text{conv}, 1 \times 1, 256 \end{bmatrix} \times 3$	$32 \times T/8 \times 256$
	$\begin{bmatrix} \text{conv}, 1 \times 1, 256 \\ \text{conv}, 3 \times 3, 256 \\ \text{conv}, 1 \times 1, 512 \end{bmatrix} \times 3$	$16 \times T/16 \times 512$
	max pool, 3×1, stride (2, 2)	$7 \times T/32 \times 512$
	conv2d, 7×1, 512	$1 \times T/32 \times 512$

Table 1: The thin-ResNet used for frame level feature extraction. ReLU and batch-norm layers are not shown. Each row specifies the # of convolutional filters, their sizes, and the # filters. This architecture has only 3 million parameters compared to the standard ResNet-34 (22 million).

NetVLAD. The second part of the network uses NetVLAD [16] to aggregate frame-level descriptors into a single utterance-level vector. Here we provide a brief overview of NetVLAD (for full details please refer to [16]).

The thin ResNet maps the input spectrogram ($R^{257\times T\times 1}$) to frame-level descriptors with size $R^{1\times T/32\times 512}$. The NetVLAD layer then takes dense descriptors as input and produces a single $K \times D$ matrix V, where K refers to the number of chosen cluster, and D refers to the dimensionality

of each cluster. Concretely, the matrix of descriptors V is computed using the following equation:

$$V(k,j) = \sum_{t=1}^{T/32} \frac{e^{w_k x_t + b_k}}{\sum_{k'=1}^{K} e^{w'_k x_t + b_{k'}}} (x_t(j) - c_k(j)) \qquad (1)$$

where $\{w_k\}, \{b_k\}$ and $\{c_k\}$ are trainable parameters, with $k \in [1, 2, ..., K]$. The first term corresponds to the soft-assignment weight of the input vector x_i for cluster k, while the second term computes the residual between the vector and the cluster centre. The final output is obtained by performing $L2$ normalisation and concatenation. To keep computational and memory requirements low, dimensionality reduction is performed via a Fully Connected (FC) layer, where we pick the output dimensionality to be 512. We also experiment with the recently proposed **GhostVLAD** [17] layer, where some of the clusters are not included in the final concatenation, and so do not contribute to the final representation, these are referred to as 'ghost clusters' (we used *two* in our implementation). Therefore, while aggregating the frame-level features, the contribution of the noisy and undesirable sections of a speech segment to normal VLAD clusters is effectively down-weighted, as most of their weights have been assigned to the 'ghost cluster'. For further details, please see [17].

3. EXPERIMENTS

3.1. Datasets

We train our model end-to-end on the `VoxCeleb2` [3] dataset (only on the 'dev' partition, this contains speech from 5,994 speakers) for identification and test on the `VoxCeleb1` verification test sets [3]. Note that the development set of `VoxCeleb2` is completely disjoint from the `VoxCeleb1` dataset (*i.e.* no speakers in common).

3.2. Training Loss

Besides the standard softmax loss, we also experiment with the additive margin softmax (AM-Softmax) classification loss [24] during training. This loss is known to improve verification performance by introducing a margin in the angular space. The loss is given by the following equation:

$$L_i = -log \frac{e^{s(\cos\theta_{y_i} - m)}}{e^{s(\cos\theta_{y_i} - m)} + \sum_{j \neq y_i} e^{s\cos(\theta_j)}} \qquad (2)$$

where L_i refers to cost of assigning the sample to the correct class, $\theta_y = arccos(w^T x)$ refers to the angle between sample features (x) and the decision hyperplane (w), as both vectors have been L2 normalised. The goal is therefore to minimise this angle by making $cos(\theta_{y_i}) - m$ as large as possible, where m refers to the angular margin. The hyper-parameter s controls the "temperature" of the softmax loss, producing higher gradients to the well-separated samples (and further shrinking the intra-class variance). We used the default values $m = 0.4$ and $s = 30$ [24].

3.3. Training Details

During training, we use a fixed size spectrogram corresponding to a 2.5 second temporal segment, extracted randomly from each utterance. Spectrograms are generated in a sliding window fashion using a hamming window of width 25ms and step 10ms. We use a 512 point FFT, giving us 256 frequency components, which together with the DC component of each frame gives a short-time Fourier transform (STFT) of size 257×250 (frequency $\times$temporal) out of every 2.5 second crop. The spectrogram is normalised by subtracting the mean and dividing by the standard deviation of all frequency components in a single time step. No voice activity detection (VAD), or automatic silence removal is applied. We use the Adam optimiser with an initial learning rate of $1e-3$, and decrease the learning rate by 10 after every 36 epochs until convergence.

4. RESULTS

In this section we first compare the performance of our NetVLAD and GhostVLAD architectures trained using different losses to the state of the art, and then investigate how performance varies with utterance length.

4.1. Verification on VoxCeleb1

The trained network is evaluated on three different test lists from the `VoxCeleb1` dataset: (1) the original `VoxCeleb1` test list with 40 speakers; (2) the extended `VoxCeleb1-E` list that uses the entire `VoxCeleb1` (train and test splits) for evaluation; and (3) the challenging `VoxCeleb1-H` list where the test pairs are drawn from identities with the same gender and nationality. In addition, we find that there are a small number of errors in the `VoxCeleb1-E` and `VoxCeleb1-H` lists, and hence we evaluate on a cleaned up version of both lists as well, which we release publically. The network is tested on the full length of the test segment. We do not use any test time augmentation, which could potentially lead to slight performance gains.

Table 2 compares the performance of our models to the current state-of-the-art on the original `VoxCeleb1` test set. Our model outperforms all previous methods. With standard softmax loss and a NetVLAD aggregation layer, it outperforms the original ResNet-based architecture [3] by a significant margin (EER of 3.57% vs 4.19%) whilst requiring far fewer parameters (10 vs 26 million). By replacing the standard softmax with the additive margin softmax (AM-Softmax), a further performance gain is achieved (3.32% EER). The GhostVLAD layer, which excludes irrelevant information from the aggregation, additionally makes a modest contribution to performance (3.22% EER). On the challenging VoxCeleb1-H test set, we outperform the original ResNet-based architecture [3] (EER of 5.17% vs 7.33%), which is by a larger margin than on the original VoxCeleb1 test set. The most similar architecture to ours is the dictionary based method of Cai *et al.* [15], which we also outperform

	Front-end model	Loss	Dims	Aggregation	Training set	EER (%)
VoxCeleb1 test set						
Nagrani *et al.* [2]	I-vectors + PLDA	–	–	–	VoxCeleb1	8.8
Nagrani *et al.* [2]	VGG-M	Softmax	1024	TAP	VoxCeleb1	10.2
Cai *et al.* [15]	ResNet-34	A-Softmax + PLDA	128	TAP	VoxCeleb1	4.46
Cai *et al.* [15]	ResNet-34	A-Softmax + PLDA	128	SAP	VoxCeleb1	4.40
Cai *et al.* [15]	ResNet-34	A-Softmax + PLDA	128	LDE	VoxCeleb1	4.48
Okabe *et al.* [13]	TDNN (x-vector)	Softmax	1500	TAP	VoxCeleb1	4.70
Okabe *et al.* [13]	TDNN (x-vector)	Softmax	1500	SAP	VoxCeleb1	4.19
Okabe *et al.* [13]	TDNN (x-vector)	Softmax	1500	ASP	VoxCeleb1	3.85
Hajibabaei *et al.* [19]	ResNet-20	A-Softmax	128	TAP	VoxCeleb1	4.40
Hajibabaei *et al.* [19]	ResNet-20	AM-Softmax	128	TAP	VoxCeleb1	4.30
Chung *et al.* [3]	ResNet-34	Softmax + Contrastive	512	TAP	VoxCeleb2	5.04
Chung *et al.* [3]	ResNet-50	Softmax + Contrastive	512	TAP	VoxCeleb2	4.19
Ours	Thin ResNet-34	Softmax	512	TAP	VoxCeleb2	10.48
Ours	Thin ResNet-34	Softmax	512	NetVLAD	VoxCeleb2	3.57
Ours	Thin ResNet-34	AM-Softmax	512	NetVLAD	VoxCeleb2	3.32
Ours	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	**3.22**
Ours	Thin ResNet-34	AM-Softmax	512	GhostVLAD	VoxCeleb2	3.23
Ours (cleaned †)	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	3.24
VoxCeleb1-E						
Chung *et al.* [3]	ResNet-50	Softmax + Contrastive	512	TAP	VoxCeleb2	4.42
Ours	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	**3.24**
Ours (cleaned †)	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	**3.13**
VoxCeleb1-H						
Chung *et al.* [3]	ResNet-50	Softmax + Contrastive	512	TAP	VoxCeleb2	7.33
Ours	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	**5.17**
Ours (cleaned †)	Thin ResNet-34	Softmax	512	GhostVLAD	VoxCeleb2	**5.06**

Table 2: Results for verification on the original VoxCeleb1 test set [2] and the extended and hard test sets (VoxCeleb-E and VoxCeleb-H) [3]. [15, 13, 19] do not report results on the VoxCeleb-E and VoxCeleb-H) [3] test sets. TAP: Temporal Average Pooling. SAP: Self-attentive Pooling Layer [15], † Cleaned up versions of the test lists have been released publically. We encourage other researchers to evaluate on these lists.

(EER of 3.22% vs 4.48%). We note that training a softmax loss based on features from temporal average pooling (TAP) yields extremely poor results (EER of 10.48%). We conjecture that the features from TAP are typically good at optimizing the inter-class difference (i.e., separating different speakers), but not good at reducing the intra-class variation (i.e. making features of the same speaker compact). Therefore, contrastive loss with online hard sample mining leads to a significant performance boost, as demonstrated in [3] for TAP. It is possible that this would also give a performance boost for NetVLAD/GhostVLAD pooling.

4.2. Probing verification based on length

Table 3 shows the effect of the length of the test segment on speaker recognition performance. In order to provide a fair comparison on lengths up to 6 seconds, we restricted the testing dataset (VoxCeleb1) to speech segments that were 6 seconds or longer (87,010 segments or 56.7% of the total dataset). To generate verification pairs, for each speaker in the VoxCeleb1 dataset (1251 speakers in total), we randomly sample 100 positive pairs and 100 negative pairs, resulting in 25,020 verification pairs. During testing, segments of length 2s, 3s, 4s, 5s and 6s are randomly cropped from each verification pair. We repeat this process three times, and compute mean and standard deviation.

As shown in Table 3, there is indeed a strong correlation between verification performance and sequence length. As the temporal length increases, there is a higher chance of capturing relevant speech signals from the actual speaker.

Segment length(s)	2	3	4	5	6
EER	7.97±0.06	5.73±0.04	4.70±0.02	4.10±0.02	3.39±0.02

Table 3: Utterance length (in seconds) on performance.

5. CONCLUSION

In this paper, we have proposed a powerful speaker recognition network, using a 'thin-ResNet' trunk architecture, and a dictionary-based NetVLAD and GhostVLAD layers to aggregate features across time that can be trained end-to-end. The network achieves state-of-the-art performance on the popular `VoxCeleb1` test set for speaker recognition, whilst requiring fewer parameters than previous methods. We have also shown the effect of utterance length on performance, and concluded that for 'in the wild' data, a longer length is beneficial.

Acknowledgements. Funding for this research is provided by the EPSRC Programme Grant Seebibyte EP/M013774/1. AN is supported by a Google PhD Fellowship in Machine Perception, Speech Technology and Computer Vision.

6. REFERENCES

[1] M. McLaren, L. Ferrer, D. Castan, and A. Lawson, "The speakers in the wild (SITW) speaker recognition database," in *INTERSPEECH*, 2016.

[2] A. Nagrani, J. S. Chung, and A. Zisserman, "VoxCeleb: a large-scale speaker identification dataset," in *INTERSPEECH*, 2017.

[3] J. S. Chung, A. Nagrani, and A. Zisserman, "VoxCeleb2: Deep speaker recognition," in *INTERSPEECH*, 2018.

[4] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, et al., "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint arXiv:1603.04467*, 2016.

[5] A. Paszke, S. Gross, S. Chintala, G. Chanan, E. Yang, Z. DeVito, Z. Lin, A. Desmaison, L. Antiga, and A. Lerer, "Automatic differentiation in pytorch," 2017.

[6] A. Vedaldi and K. Lenc, "Matconvnet: Convolutional neural networks for matlab," in *Proc. ACMM*, 2015.

[7] 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.

[8] L. Wan, Q. Wang, A. Papir, and I.L. 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.

[9] Y. Lukic, C. Vogt, O. Dürr, and T. Stadelmann, "Speaker identification and clustering using convolutional neural networks," in *IEEE 26th International Workshop on Machine Learning for Signal Processing (MLSP)*. IEEE, 2016, pp. 1–6.

[10] I. Lopez-Moreno, J. Gonzalez-Dominguez, O. Plchot, D. Martinez, J. Gonzalez-Rodriguez, and P. Moreno, "Automatic language identification using deep neural networks," in *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on*. IEEE, 2014, pp. 5337–5341.

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

[12] S. Shon, H. Tang, and J. Glass, "Frame-level speaker embeddings for text-independent speaker recognition and analysis of end-to-end model," *arXiv preprint arXiv:1809.04437*, 2018.

[13] K. Okabe, T. Koshinaka, and K. Shinoda, "Attentive statistics pooling for deep speaker embedding," *arXiv preprint arXiv:1803.10963*, 2018.

[14] W. Cai, Z. Cai, X. Zhang, X. Wang, and M. Li, "A novel learnable dictionary encoding layer for end-to-end language identification," *arXiv preprint arXiv:1804.00385*, 2018.

[15] W. Cai, J. Chen, and M. Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

[16] R. Arandjelović, P. Gronat, A. Torii, T. Pajdla, and J. Sivic, "NetVLAD: CNN architecture for weakly supervised place recognition," in *Proc. CVPR*, 2016.

[17] Y. Zhong, R. Arandjelović, and A. Zisserman, "GhostVLAD for set-based face recognition," in *Asian Conference on Computer Vision, ACCV*, 2018.

[18] W. Cai, J. Chen, and M. Li, "Analysis of length normalization in end-to-end speaker verification system," *arXiv preprint arXiv:1806.03209*, 2018.

[19] M. Hajibabaei and D. Dai, "Unified hypersphere embedding for speaker recognition," *arXiv preprint arXiv:1807.08312*, 2018.

[20] G. Bhattacharya, J. Alam, and P. Kenny, "Deep speaker embeddings for short-duration speaker verification," in *Proc. Interspeech*, 2017, pp. 1517–1521.

[21] D. Snyder, D. Garcia-Romero, G. Sell, D. Povey, and S. Khudanpur, "X-vectors: Robust dnn embeddings for speaker recognition," *ICASSP, Calgary*, 2018.

[22] FA Chowdhury, Quan Wang, Ignacio Lopez Moreno, and Li Wan, "Attention-based models for text-dependent speaker verification," *arXiv preprint arXiv:1710.10470*, 2017.

[23] D. Chen, S. Tsai, V. Chandrasekhar, G. Takacs, H. Chen, R. Vedantham, R. Grzeszczuk, and B. Girod, "Residual enhanced visual vectors for on-device image matching," in *Asilomar*, 2011.

[24] F. Wang, W. Liu, H. Liu, and J. Cheng, "Additive margin softmax for face verification," *arXiv preprint arXiv:1801.05599*, 2018.