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

MagNetO: X-vector Magnitude Estimation Network plus Offset for Improved Speaker Recognition

Daniel Garcia-Romero, Gregory Sell, and Alan McCree

Human Language Technology Center of Excellence
Johns Hopkins University, Baltimore, MD 21218, USA
{dgromero,gsell2,alan.mccree}@jhu.edu

Abstract

We present a magnitude estimation network that is combined with a modified ResNet x-vector system to generate embeddings whose inner product is able to produce calibrated scores with increased discrimination. A three-step training procedure is used. First, the network is trained using short segments and a multi-class cross-entropy loss with angular margin softmax. During the second step, only a reduced subset of the DNN parameters are refined using full-length recordings. Finally, the magnitude estimation network is trained using a binary cross-entropy loss over pairs of target and non-target trials. The resulting system is evaluated on 4 widely-used benchmarks and provides significant discrimination and calibration gains at multiple operating points.

1. Introduction

The recent NIST SRE-{18,19} [1, 2] evaluations, as well as the VoxSRC [3] challenge, show that the state-of-the-art in speaker recognition is represented by DNNs that produce embeddings using a classification loss (e.g., multiclass cross entropy). In this way, the goal of training the network is to produce embeddings that generalize well to speakers beyond those in the training set.

A successful example of this paradigm is the x-vector system presented in [4, 5], and independently validated in [6, 7]. The x-vector DNN uses a stack of convolutional layers followed by a temporal pooling layer that computes the mean and standard deviation of an input sequence to capture the speaker characteristics over the entire recording. Different topologies of 1D convolutional layers (TDNNs) have been explored [8, 9], and recently, modified ResNets [10, 3] with 2D convolutions have outperformed the TDNN alternatives.

Once the DNN is trained, the embeddings are extracted for each recording and compared using a similarity metric to produce a score. If there is a severe domain-shift between the DNN training data and the deployment environment, a disjoint metric learning process can be used on top of the embeddings to perform domain adaptation. Variants of probabilistic linear discriminant analysis (PLDA) [11, 12, 13, 14] for which well-established domain adaptation methods [15] exist are common. If this is not the case, then the metric used inside the DNN to produce the logits for the softmax classification can generalize well [16, 10, 17] (e.g., cosine distance for DNNs trained with angular softmax [18]).

A principled way to make speaker detection decisions based on these scores is given by Bayes decision theory [19]. If the score is above the Bayes threshold, it is declared a target trial (the speaker is present in the recording), and non-target otherwise. To use this formalism, the system must produce scores that are calibrated log-likelihood-ratios [19]. That is, the scores need to be in units such that the Bayes threshold produces optimal decisions. Unfortunately, this is typically not the case for PLDA scoring, and it is even worse for cosine scoring where the scores are constrained to the interval [-1,1]. Therefore, a calibration mapping (a function that transforms scores into calibrated log-likelihood-ratios) is needed. The most common approach is to use logistic regression to learn a global affine transformation (scale and offset) using a binary cross-entropy loss [20]. This produces a monotonic mapping that preserves the relative ordering of scores (i.e., it does not improve the discrimination ability of the system) and makes the Bayes threshold optimal.

In this work, we propose a magnitude estimation network that can be used with unit-length x-vectors (extracted from a DNN trained with angular softmax [21]) to produce calibrated scores and improved discrimination. Motivated by the success reported in [22], using a refinement stage for a subset of the DNN parameters (freezing all the pre-pooling layer parameters), we follow this approach and train the magnitude estimation as a refinement stage using as input the activations of the pooling layer. This can be understood as a generalization of a global affine transformation, in which each x-vector gets a scale (magnitude) that replaces the global scale factor of the linear calibrator. Note that this results in a non-monotonic mapping that can improve the discrimination ability of the system. The magnitude estimation network and a global offset are trained using a binary cross-entropy loss.

Our experimental setup focuses on wide-band (16KHz sampling rate) datasets. We used the VoxCeleb2-dev set [23] to train all the parameters. We evaluate performance on the speakers in the wild (SITW) core-core task [24], the VoxCeleb1-E (extended test set) [25, 26], the SRE19 [2] audio from video data, and the Chime-5 speaker recognition task [27]. The results show that a single modified ResNet achieves great performance in all the sets. Additionally, a refinement stage using full-duration recordings [22] boosts the baseline performance. Finally, the addition of the magnitude estimation network further improves the discrimination, and along with the global offset also improves calibration.

2. X-vector System

The x-vector system is a DNN that computes speaker embeddings from variable-length speech segments. Recently, variants of ResNet-based x-vectors [10, 17] trained with additive margin have obtained top performance in the VoxSRC [3] and NIST SRE-19 [2] evaluations. In this work, we propose a modified version of the ResNet-34 presented in [10] where we have

10.21437/Odyssey.2020-1

Table 1: *Modified ResNet-34 architecture with 15.4 million parameters. Batch-norm and ReLU layers are not shown. The 1×1 convolutions are used to match the dimensions for the residual connections. The dimensions are (Channels×Frequency×Time). The input comprises 80 Mel-filter bank energies from speech segments. During training we use a fixed segment length of $T = 400$.*

Layer name	Structure	Output ($C \times F \times T$)
Input	–	$1 \times 80 \times T$
Conv2D	3×3, stride=1	$128 \times 80 \times T$
ResBlock-1	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, stride=1	$128 \times 80 \times T$
ResBlock-2a	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 128 \end{bmatrix} \times 1$, stride=2	$128 \times 40 \times T/2$
ResBlock-2b	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 3$, stride=1	$128 \times 40 \times T/2$
ResBlock-3a	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 256 \end{bmatrix} \times 1$, stride=2	$256 \times 20 \times T/4$
ResBlock-3b	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 5$, stride=1	$256 \times 20 \times T/4$
ResBlock-4a	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 256 \end{bmatrix} \times 1$, stride=2	$256 \times 10 \times T/8$
ResBlock-4b	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 256 \end{bmatrix} \times 2$, stride=1	$256 \times 10 \times T/8$
Flatten (C, F)	–	$2560 \times T/8$
StatsPooling	–	5120
Dense (Emb.)	–	256
AM-Softmax		Num. spks.

allocated more channels to the early layers and changed their expansion rate at deeper layers. We have found that increasing channels in the early stages improves performance, and keeping them smaller in the deeper layer controls the network parameters with no degradation. Table 1 summarizes the proposed architecture.

3. Angular softmax with additive margin

The use of cosine similarity as the logit input to the softmax layer is referred to in the literature as angular softmax [18]. A number of variants have been proposed [21, 28] to reduce the interclass variance by introducing the notion of a margin penalty to the target class logit. Effective applications for speaker recognition have been presented in [29, 30, 22, 10]. In this work, we use the additive margin variant [21] due to its good performance and ease of implementation. The corresponding loss function is

$$\mathcal{L}_{AM} = -\frac{1}{n}\sum_{i=1}^{n} \log \frac{e^{s(\cos\theta_{y_i} - m)}}{e^{s(\cos\theta_{y_i} - m)} + \sum_{j \neq y_i} e^{s\cos\theta_j}}, \quad (1)$$

where $\cos\theta_{y_i} = \mathbf{w}_{y_i}^T \mathbf{f}_i / \|\mathbf{w}_{y_i}\| \, \|\mathbf{f}_i\|$, $\mathbf{w}_{y_i}$ is the weight vector of class y_i, and $\mathbf{f}_i$ is the input to the layer for example i. Also, s is an adjustable scale factor and m is the penalty margin. To accelerate convergence, we follow the practice of fixing s to a predefined value.

4. Magnitude estimation

The x-vectors extracted from our network have unit-length and have been trained to encode the speaker specific information in their direction. In this way, given two embeddings $\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$, their inner product computes a cosine distance score

$$s_{ij} = \cos\theta_{ij} = \frac{\mathbf{x}_i^T}{\|\mathbf{x}_i\|}\frac{\mathbf{x}_j}{\|\mathbf{x}_j\|} = \tilde{\mathbf{x}}_i^T \tilde{\mathbf{x}}_j. \quad (2)$$

A global affine transformation (scale a and offset b) of the form

$$\ell_{ij} = a\, s_{ij} + b = (\sqrt{a}\,\tilde{\mathbf{x}}_i)^T(\sqrt{a}\,\tilde{\mathbf{x}}_j) + b \quad (3)$$

is typically used to map the scores s_{ij} into calibrated log-likelihood-ratios ℓ_{ij}. In our context, this is akin to estimating a global magnitude ($\sqrt{a}$) for all the embeddings. This can be generalized by allowing each x-vector to have a different magnitude

$$\ell_{ij} = a_i\, a_j\, s_{ij} + b = (a_i\,\tilde{\mathbf{x}}_i)^T(a_j\,\tilde{\mathbf{x}}_j) + b. \quad (4)$$

To accomplish this, we use a magnitude estimation network (MagNet) that maps the DNN post-pooling activations into a scalar for each embedding. Note that this results in a non-monotonic mapping of the scores, and therefore, it can improve the speaker discrimination. To train the network and the global offset, we draw target ($ij \in \mathcal{T}$) and non-target ($ij \in \mathcal{N}$) pairs from our training set, and use a prior-weighted binary cross entropy loss [20]

$$\mathcal{L}_b = \frac{\alpha}{|\mathcal{T}|}\sum_{ij\in\mathcal{T}} \log(1 + e^{-p_{ij}}) + \frac{1-\alpha}{|\mathcal{N}|}\sum_{ij\in\mathcal{N}} \log(1 + e^{p_{ij}}), \quad (5)$$

where α is the prior probability for a target trial, and $p_{ij} = \ell_{ij} + log(\alpha/(1-\alpha))$ is the sum of the log-likelihood-ratio and the log-prior-odds.

5. Training

5.1. Data preparation

We trained our DNNs using the VoxCeleb2-dev [23] data (1,092,009 utterances from 5,994 speakers). We create 2 different augmented versions of it. One for the initial training stage, and another one for refinement. Since the initial training is done using short segments (e.g. 4 seconds), we augment the original utterances to obtain around 6 million utterances. To augment an utterance, we randomly pick from one of the following strategies:

- **Reverb:** Artificially reverberate via convolution with simulated RIRs from the AIR dataset
- **Music:** A single music file (without vocals) is randomly selected from MUSAN, trimmed or repeated as necessary to match duration, and added to the original signal (5-15dB SNR).
- **Noise:** MUSAN noises are added at one second intervals throughout the recording (0-15dB SNR).

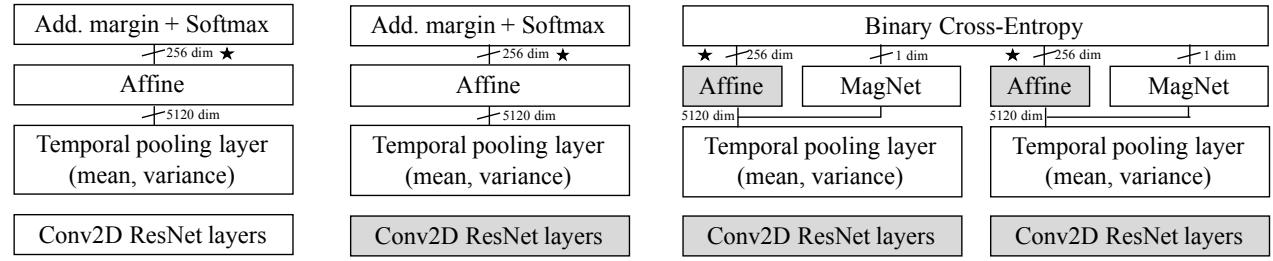

Figure 1: Block diagram of the 3 stages used to train the networks. During the full-length refinement and magnitude estimation only a subset of the parameters are updated. The filled boxes indicate the frozen parameters. For the magnitude estimation, we depict a Siamese structure to indicate that the parameters are trained using pairs of embeddings with a binary cross-entropy loss.

- **Babble:** Three to seven speakers are randomly picked from MX6-micphn, summed together, then added to the original signal (13-20dB SNR).

For the refinement stage, we combine the utterances of each recording together (resulting in 144,789 recordings). The augmentation process was devised in a similar fashion to the network training data, but designed to be more severe and include several new alterations:

- **Reverb**: simulated RIRs drawn from the same dataset as in network training, but from a separate portion with longer decay times.
- **Added noises**: Noise, music, and babble speech were added similarly to the network training, but with lower SNRs ranging 0-12 dB. Additionally, randomly-generated modulated noise was included in the list of possible noises, as well as environmental recordings from the 2013 DCASE Task 1 dataset [1] and the DEMAND dataset [2]
- **Filtering** Input audio was randomly bandpass or bandstop filtered, with bandpass responses randomly chosen of width 2000-3500 Hz and bandstop responses randomly chosen with width 200-500 Hz.
- **Compression** The waveform of a signal was compressed, with the knee located between 6 and 3 dB below the maximum energy of the file, and the slope of the compression function above the knee chosen between 0 and 0.5 (a slope of 0 results in a limiter).
- **Encoding** Input audio was encoded and decoded according to either the GSM-AMR or MPEG standard, with bitrates randomly chosen as 4.75, 6.7, or 12.2 for GSM-AMR or 4, 8, 16, or 32 for MPEG.

In order to further distinguish the data for this second stage from the network training data, augmentations were also performed in series, with up to three occurring for a particular segment. Augmentations were categorized as noises (music, noise, babble, random noise, environment), effects (bandpass/bandstop filter, compress), or channels (reverb, GSM-AMR, MPEG) and up to one of each category of augmentation was selected for each file in any order. In total, the original dataset was augmented 10 unique times, resulting in 1,592,679 total unique training examples.

The audio was processed at 16 KHz sampling rate. We used 80 Mel filter bank log-energies with a 25 ms window every 10 ms over the spectral band of 20–7600 Hz. Mean normalization was applied using a moving window of 3 seconds. A small DNN was used for speech activity detection.

5.2. DNN training

Figure 1 shows the 3 steps used to train the x-vector extractor as well as the magnitude estimation network. Following the strategy in [22], we first train the x-vector network with short segment as an initialization, and then perform a full-recording refinement of a subset of the network parameters. Freezing the DNN components that are responsible for most of the memory (i.e., pre-pooling layers) facilitates the use of longer sequences, and also controls the network capacity so that it does not overfit. Unlike in the first 2 steps, where multi-class cross-entropy is the loss function, the magnitude estimation parameters are trained using pairs of embeddings with a binary cross-entropy loss.

5.2.1. Short segment training

Although we would ideally seek to minimize the mismatch between training and inference, in practice, there are two main reasons why we restrict training to short segments. First, longer batch sizes become feasible in limited GPU memory. Second, longer sequences are easier to classify and the network tends to overfit to them.

The x-vector extractor was trained using PyTorch with data parallelism over 4 Nvidia Titan RTX GPUs. We used SGD with momentum (set to 0.9) with a batch size of 256. To construct a batch, we uniformly sample (without replacement) a subset of speakers. Then, an audio segment from each speaker is sampled and a chunk of 4 seconds is selected using a random offset over the audio segment. We discard audio segments shorter than 4 seconds for the training process. Once all the classes are sampled, we repeat the same process until the end of training. Note that this results in a training set balanced over the speakers. 150K training steps were sufficient to train the networks. The learning rate scheduler starts with a learning rate of 0.2, keeps it constant for 50K steps, and then it applies an exponential decay every 10K steps with a rate of 1/2. The period of constant learning rate is important for training with margin penalty. The margin value was set to $m = 0.3$ and we used a scale of 40.

[1] http://dcase.community/challenge2013/download

[2] https://zenodo.org/record/1227121#.XjRVRS2ZPOQ

Table 2: Comparison of our baseline x-vector system with top performing prior work and also after applying the full-length refinement.

Eval set	BASELINE			REFINEMENT [22]			PRIOR WORK			
	EER	Min C_1	Min C_2	EER	Min C_1	Min C_2	Ref.	EER	Min C_1	Min C_2
VOX1-E	1.05	0.067	0.116	1.17	0.067	0.112	[10]	1.35	–	–
SITW	1.05	0.059	0.095	0.88	0.051	0.085	[8]	1.70	–	0.200
SRE19	1.53	0.083	0.128	1.58	0.063	0.094	[17, 31]	1.76	0.065	–
C5-BIN	0.44	0.023	0.034	0.38	0.022	0.031	[27]	1.20	–	0.130
C5-U01	11.1	0.449	0.574	10.48	0.406	0.517	[27]	13.70	–	0.770
C5-U04	6.86	0.316	0.425	6.47	0.282	0.387	[27]	9.90	–	0.680

Table 3: Topology of three magnitude estimation networks indicating the input and output sizes of the affine layers, as well as the total number of parameters of the network.

Layer	MagNetO-1 (1.5 M)	MagNetO-2 (2.8 M)	MagNetO-3 (20 M)
Affine+ReLU	5120, 256	5120, 512	5120, 2560
Affine+ReLU	256, 256	512, 512	2560, 2560
Affine+ReLU	256, 256	512, 1	2560, 1
Affine+ReLU	256, 1	–	–

5.2.2. Full recording refinement

For the refinement, we freeze all the pre-pooling layers and use the augmented full-recording data described in Section 5.1. Note that unlike in [22], the network is already trained with angular softmax so we do not need to alter its topology. We used the same batch size and sampling strategy as in the initial stage. However, we modified the learning rate scheduler and the parameters of the angular margin. The learning rate was initialized to 0.1 with an exponential decay every 2K steps with a rate of 1/2. Convergence was obtained after 20K steps. The margin value was increased to $m = 0.5$ (to make it harder for training with the longer sequences) and the scale was reduced to 30.

5.2.3. Magnitude and offset estimation

After the full recording refinement, we discard the angular softmax classification head and switch the loss function to use binary cross-entropy over pairs of target and non-target trials. Also, we freeze the post-pooling affine layer (bottleneck to obtain the x-vector) and introduce a sub-network whose purpose is to estimate a magnitude for each x-vector. Additionally, a global offset parameter is added as indicated in Eq. 4. We explored multiple architectures for the magnitude network (MagNet) and observed that it does not need to be very large (see Section 7 for details). The global offset was initialized to the value produced by a global calibrator. For all the MagNet architectures, the weights of the last affine layer were initialized to zero and the bias to the square root of the scale provided by a global calibrator. A ReLU non-linearity was used to ensure non-negative magnitudes. In this way, the first forward pass of the process produced constant magnitude estimates for all x-vectors equal to a global calibrator.

The target and non-target trials were created by constructing batches of 1000 recordings (10 recordings from 100 speakers). The sampling process was uniform across speakers. All unique pairs (upper triangular part of a full square matrix of scores) of scores were computed. As a result of this process, each batch yielded around 5K target trials and 500K non-target trials. To speed up computation, only the 40% highest non-target scores were used to evaluate the cost function.

As in the other steps, we used SGD with momentum (set to 0.9). The learning rate was initialized to 0.01 with an exponential decay every 6K steps with a rate of 1/2. We trained for a total of 30K steps. The target prior α of Eq. 5 was set to 0.01 (matching the empirical prior of the sampled trials in a batch).

6. Experimental setup

We evaluate performance on the speakers in the wild (SITW) core-core task [24], the VoxCeleb1-E (cleaned version) [25, 26], the SRE19 [2] audio from video data, and the Chime-5 (C5) speaker recognition task [27]. For the Chime-5 evaluation we use the core-core task (i.e., test segments are diarized using the manual annotations), and report on 3 conditions that increase in difficulty. The enrollment data is always from the close-talking microphone. The test data varies from close-talking (C5-BIN), single microphone from distant array U04 (C5-U04), and single microphone from array U01 that is further from the speakers than U04 for most of the test segments (C5-U01). Note that the content of the test segments is the same for the three conditions (simultaneous recording) and the only change is in the microphone used to capture the audio. The test segments of SRE19 contain multiple speakers. The score for a trial is obtained by taking the max [32] over all the pairs of scores between the enrollment and the test segment clusters given by the diarization algorithm in [33].

We report results in terms of equal error-rate (EER) and normalized detection cost (min and/or actual) at two operating points with C_1 using $P_{\text{target}} = 0.05$ and C_2 using $P_{\text{Target}} = 0.01$. In both cases the cost of false-alarm and miss-detection is set to 1. The choice of operating points was done to facilitate comparisons with prior results.

7. Results

The performance of the baseline (proposed ResNet) x-vector system is shown in Table 2. We include the best results we have found in prior work to provide a context (single system with no fusion). The baseline produces a strong performance for all the benchmarks, specially for the challenging conditions of C5. We use a simple pipeline that consists of extracting x-vectors from speech segments and directly computing their cosine distance to score a trial. As previously mentioned, the SRE19 test segments are diarized, so we take the max over the clusters to score a trial. Other than that, there is no customization per dataset.

Refining the baseline with full recordings, also shown in Table 2, results in a positive trend across datasets. The SITW benchmark shows the largest improvement, while the

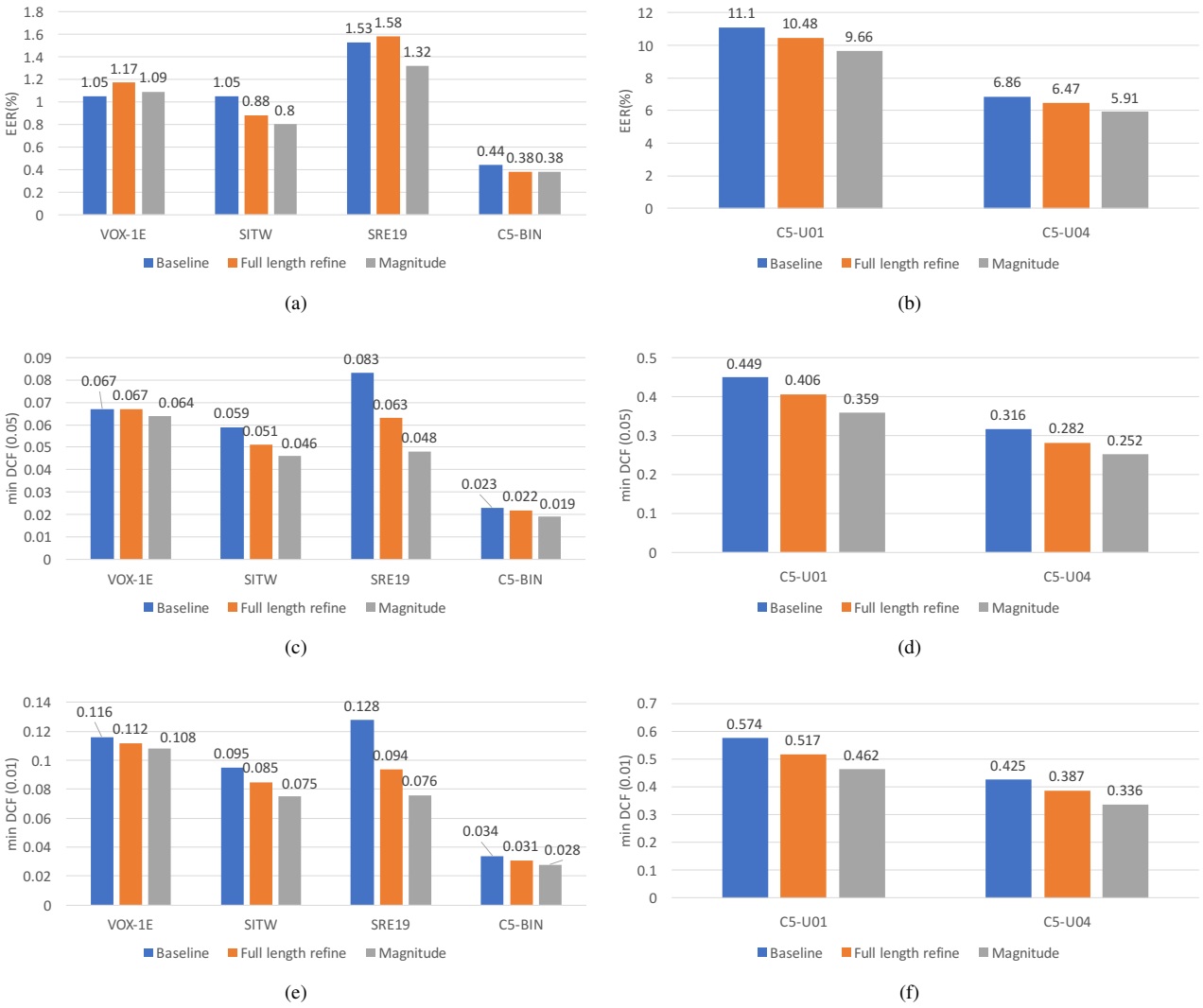

Figure 2: Bar plots that summarize the overall discrimination performance in terms of EER and min C_1 and C_2 costs. The more challenging C5 tasks are split into their own graph to facilitate visualization.

Table 4: Performance of three magnitude estimation networks.

Eval set	MagNetO-1 (1.5 M)			MagNetO-2 (2.8 M)			MagNetO-3 (20 M)		
	EER	Min C_1	Min C_2	EER	Min C_1	Min C_2	EER	Min C_1	Min C_2
VOX1-E	1.10	0.064	0.108	1.09	0.064	0.108	1.09	0.063	0.108
SITW	0.85	0.047	0.078	0.80	0.046	0.075	0.82	0.048	0.076
SRE19	1.35	0.050	0.077	1.32	0.048	0.076	1.14	0.048	0.081
C5-BIN	0.41	0.019	0.031	0.38	0.019	0.028	0.37	0.018	0.026
C5-U01	9.56	0.355	0.459	9.66	0.359	0.462	9.61	0.354	0.455
C5-U04	5.89	0.251	0.336	5.91	0.252	0.336	5.87	0.247	0.332

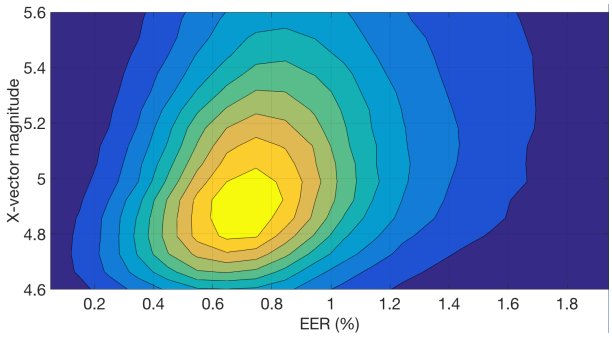

Figure 3: Surface plot (high values in yellow and low values in dark-blue) of a 2D histogram of pairs (EER, x-vector magnitude) on the VoxCeleb2-dev set using MagNetO-2 (see end of Section 7 for details).

VoxCeleb1-E (VOX1-E) benefits the least. This is consistent with the duration distribution of the SITW audio recordings (average 60 seconds) and much shorter for VOX1-E (average of 8 seconds). Overall, the average improvement across all the sets is around 4% in EER and 10% for both C_1 and C_2. These results further validate the findings in [22], and also show that it can produce gains at all operating points.

We did a wide search over feed-forward architectures for the magnitude estimation network. Three representative examples are summarized in Table 3. We explored depth and width of layers resulting in different number of parameters. As shown in Table 4, the performance is quite stable across networks, even when the number of parameters varies greatly (i.e., from 1.5 to 20 million). MagNetO-2 (includes the global offset) strikes a good balance of size and performance and it will be used to draw comparisons with the baseline and its full-recording refined version. Figure 2 presents bar plots that summarize the overall trends. The more challenging C5 tasks are split into their own graph. Overall, the magnitude estimation network produces gains on top of the already improved results from the full-recording refinement stage. The average improvement across all the sets is around 8% in EER and 12% for both C_1 and C_2. Additionally, comparing the final network (refine+magnitude) to the baseline we get gains of around 12% in EER and 21% for both C_1 and C_2. These are substantial improvements on top of a very strong baseline system.

Up until now we have reported results using calibration insensitive metrics to highlight the improved discrimination of the refinement and magnitude estimation stages. Table 5 includes the actual C_1 and C_2 costs for a global linear calibrator on top of the full-recording refined system versus the magnitude estimation using MagNetO-2. The calibration parameters and x-vector magnitudes were learned on the full-length VoxCeleb2-dev set. Looking at the global calibrator, we note that this training set is a good match for the VOX1-E and SITW sets, but does not match well the SRE19 and C5 tasks. Some potential factors contributing to this mismatch could be that SRE19 involves taking the max over diarized test segments, and the C5 tasks contain overlapped speech (and severe reverberation for C5-U01 and C5-U04). The MagNetO-2 system also exhibits this behavior but it decreases the calibration loss for the mismatched sets. Also, its overall actual costs are smaller for all sets.

To gain some insights into the x-vector magnitude estimation, Figure 3 shows a surface plot of a 2D histogram of pairs (EER, x-vector magnitude) on the augmented full-recording

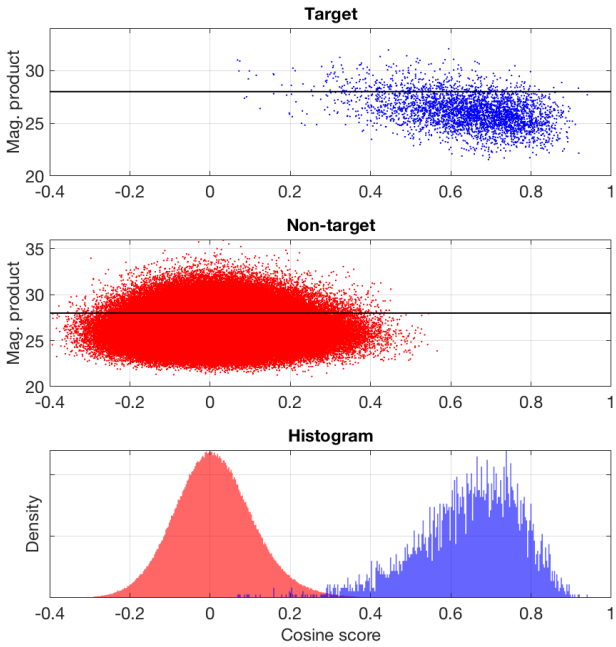

Figure 4: The top two panels show the scatter plot of cosine score (i.e. unit-length x-vectors) vs the product of the magnitudes of the embedding pairs in the SITW trials (MagNetO-2 system). The horizontal line is the constant scale assigned by a global calibrator. The bottom panel show a histogram of the score distributions.

VoxCeleb2-dev set. In particular, for each of the 1.5 million recordings we evaluate the EER by computing all pairs of target and non-target trials with the other recordings. The EER and the estimated magnitude are used to construct the histogram. We can see that most of the density is around the 0.5 to 1% EER. More interestingly, we observe a positive correlation between the x-vector magnitude (and variance) and the EER. If we focus on the high density EER region (which has lower variance) we see that smaller magnitudes are assigned to embedding for which the EER is smaller. This serves the purpose of attenuating the high-scoring non-target trials (which have a cost higher than the false-alarms for both C_1 and C_2) at the expense of also decreasing the target scores (which probably do not affect the cost much as there is good separation between the score distributions). For the situations with higher EER, there is less separation and the trade-off results in less attenuation (higher magnitude).

To observe how this generalizes beyond the training set, Figure 4 shows scatter plots of cosine score vs the product of the magnitudes of the embedding pairs in the SITW trials, as well as a histogram of the score distributions. The horizontal line in the scatter plots is the constant scale assigned by a global calibrator. Since MagNetO-2 improves speaker discrimination, it is expected that a large proportion of the target scores should have products of magnitudes larger than the global scale, and the opposite for the non-target scores. Additionally, this property is more relevant for low-scoring targets and high-scoring non-target (since they are the ones that dominate to the loss function). We can observe that this trend is present in the plots. The magnitude estimation network is able to train a function that is predictive of when a segment needs to be attenuated/amplified beyond the training set. This generalization ability is responsi-

Table 5: Comparison of a global scale and offset (linear) calibrator on top of the full-recording refined system versus the magnitude estimation using MagNetO-2. The calibration parameters and x-vector magnitudes are learned on the full-length VoxCeleb2-dev set.

Eval set	Global Linear Calibrator				Magnitude Estimation (MagNetO-2)			
	Min C_1	Act C_1	Min C_2	Act C_2	Min C_1	Act C_1	Min C_2	Act C_2
VOX1-E	0.067	0.069	0.112	0.114	0.064	0.067	0.108	0.109
SITW	0.051	0.070	0.085	0.100	0.046	0.051	0.075	0.079
SRE19	0.063	0.574	0.094	1.072	0.048	0.432	0.076	0.671
C5-BIN	0.022	0.491	0.031	0.770	0.019	0.362	0.028	0.451
C5-U01	0.406	0.680	0.517	1.053	0.359	0.544	0.462	0.724
C5-U04	0.282	0.596	0.387	0.940	0.252	0.463	0.336	0.619

ble for the performance gains previously reported.

8. Discussion

The ability of MagNetO-2 to produce improved discrimination and calibration across a wide set of tasks makes it a very attractive approach. Additionally, it results in an elegant pipeline at test time. That is, for each input audio, the network produces an embedding that can be directly compared using an inner product (scaling the unit-length x-vector with its magnitude and including the offset as an additional dimension of the embedding). The improved discrimination indicates that the magnitude can further boost the speaker information contained in the direction of the embedding (recall that angular-softmax based embeddings can only encode information in the direction). Also, learning the parameters of the magnitude network as a refinement stage using full-recordings and binary cross-entropy produces gains on top of a state-of-the-art system. To this date, our attempts to use a single stage approach (i.e., end-to-end using binary cross-entropy) have resulted in inferior results compared to the baseline.

Moving from a global calibrator (2 parameters) to a network that estimates a magnitude per embedding has the potential to not generalize well (as the capacity of the network can overfit to the training set). This is the reason why, to date, the most common approach to achieve calibration is to train a linear calibration on a training set that matches the deployment environment. However, alternatives with more capacity have been explored in the past [34] and are currently receiving more attention [35]. Of particular relevance to this discussion is the work introduced in [36]. Although this work is formulated in the context of end-to-end systems that use modified PLDA backends, they also introduce calibration parameters that depend on the signals conditions (represented by metadata vectors). Their results also support the idea that it is possible to train components with more capacity and still generalize well. Additionally, a recent extension presented in [37] shows that trial-based calibration without an explicit condition detector is also very effective (while avoiding the need for condition labels in the training set).

9. Conclusion

For our baseline, we have proposed a modified ResNet-34 x-vector system that generates embeddings that can be directly scored using cosine distance. Compared with prior work, it produces very strong results. Additionally, we have further validated the idea that a full-recording refinement of a subset of the network parameters (post-pooling) can produce significant gains. Mostly by addressing the duration mismatch induced between the training and test phases. Finally, a magnitude estimation network trained using a binary cross-entropy loss over pairs of target and non-target trials is able produce calibrated scores with increased discrimination over the four evaluation benchmarks used in this work. The resulting system provides an elegant solution where the embeddings can be directly compared using an inner product.

10. References

[1] Seyed Omid Sadjadi, Craig Greenberg, Elliot Singer, Douglas Reynolds, Lisa Mason, and Jaime Hernandez-Cordero, "The 2018 NIST Speaker Recognition Evaluation," in *Interspeech*, 2019.

[2] "NIST 2019 Speaker Recognition Evaluation Plan," 2019.

[3] Joon Son Chung, Arsha Nagrani, Ernesto Coto, Weidi Xie, Mitchell McLaren, Douglas A Reynolds, and Andrew Zisserman, "VoxSRC 2019: The First VoxCeleb Speaker Recognition Challenge," in *VoxSRC Challenge workshop*, 2019.

[4] D. Snyder, D. Garcia-Romero, D. Povey, and S. Khudanpur, "Deep neural network embeddings for text-independent speaker verification," in *Interspeech*, 2017.

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

[6] Ondrej Novotný, Oldrich Plchot, Pavel Matejka, Ladislav Mosner, and Ondrej Glembek, "On the use of x-vectors for robust speaker recognition," in *Odyssey*, 2018.

[7] Mitchell McLaren, Diego Castán, Mahesh Kumar Nandwana, Luciana Ferrer, and Emre Ylmaz, "How to train your speaker embeddings extractor," in *Odyssey*, 2018.

[8] David Snyder, Daniel Garcia-Romero, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker Recognition for Multi-Speaker Conversations Using X-Vectors," in *ICASSP*, 2019.

[9] Jesus Villalba, Nanxin Chen, David Snyder, Daniel Garcia-Romero, Alan McCree, Gregory Sell, Jonas Borgstrom, Fred Richardson, Suwon Shon, Francois Grondin, Reda Dehak, L. Paola García-Perera Paola Garcia Perera, Dan Povey, Pedro Torres-Carrasquillo, Sanjeev Khudanpur, and Najim Dehak, "State-of-the-art Speaker Recognition for Telephone and Video Speech: the JHU-MIT Submission for NIST SRE18," in *Interspeech*, 2019.

[10] Hossein Zeinali, Shuai Wang, Anna Silnova, Pavel Matejka, and Oldrich Plchot, "BUT system description to VoxCeleb speaker recognition challenge 2019," in *VoxSRC Challenge workshop*, 2019.

[11] S. Ioffe, "Probabilistic linear discriminant analysis," in *ECCV*, 2006.

[12] P. Kenny, "Bayesian speaker verification with heavy-tailed priors.," in *Odyssey*, 2010.

[13] D. Garcia-Romero and C. Espy-Wilson, "Analysis of i-vector length normalization in speaker recognition systems.," in *Interspeech*, 2011.

[14] A. Silnova, N. Brümmer, D. Garcia-Romero, D. Snyder, and L. Burget, "Fast Variational Bayes for Heavy-tailed PLDA Applied to i-vectors and x-vectors," in *Interspeech*, 2018.

[15] Daniel Garcia-Romero and Alan McCree, "Supervised domain adaptation for i-vector based speaker recognition," in *ICASSP*, 2014.

[16] Li Wan, Quan Wang, Alan Papir, and Ignacio Lopez Moreno, "Generalized End-to-End Loss for Speaker Verification," in *ICASSP*, 2018.

[17] Daniel Garcia-Romero, Gregory Sell, and Alan McCree, "JHU-HLTCOE system description for NIST SRE19 MULTIMEDIA," in *NIST SRE19 Workshop*, 2019.

[18] Weiyang Liu, Yandong Wen, Zhiding Yu, Ming Li, Bhiksha Raj, and Le Song, "Sphereface: Deep hypersphere embedding for face recognition," in *CVPR*, 2017.

[19] Niko Brümmer and Johan A. du Preez, "Application-independent evaluation of speaker detection," in *Computer Speech & Language*, 2004.

[20] Niko Brümmer, Albert Swart, and David A. van Leeuwen, "A comparison of linear and non-linear calibrations for speaker recognition," in *Odyssey*, 2014.

[21] Feng Wang, Jian Cheng, Weiyang Liu, and Haijun Liu, "Additive margin softmax for face verification," *IEEE Signal Processing Letters*, vol. 25, 2018.

[22] Daniel Garcia-Romero, David Snyder, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "X-vector DNN Refinement with Full-Length Recordings for Speaker Recognition," in *Interspeech*, 2019.

[23] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "Voxceleb2: Deep speaker recognition," in *Interspeech*, 2018.

[24] M. McLaren, Luciana Ferrer, Diego Castan, and Aaron Lawson, "The 2016 speakers in the wild speaker recognition evaluation," in *Interspeech*, 2016.

[25] A. Nagrani, J. S. Chung, and A. Zisserman, "Voxceleb: a large-scale speaker identification dataset," in *Interspeech*, 2017.

[26] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Utterance-level Aggregation For Speaker Recognition In The Wild," in *ICASSP*, 2019.

[27] Daniel Garcia-Romero, David Snyder, Shinji Watanabe, Gregory Sell, Alan McCree, Daniel Povey, and Sanjeev Khudanpur, "Speaker recognition benchmark using the CHiME-5 corpus," in *Interspeech*, 2019.

[28] Jiankang Deng, Jia Guo, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," in *CoRR*, 2018, vol. abs/1801.07698.

[29] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," in *Odyssey*, 2018.

[30] Sergey Novoselov, Andrey Shulipa, Ivan Kremnev, Alexandr Kozlov, and Vadim Shchemelinin, "On deep speaker embeddings for text-independent speaker recognition," in *Odyssey*, 2018.

[31] Jesús Villalba, Daniel Garcia-Romero, Nanxin Chen, Gregory Sell, Jonas Borgstrom, Alan McCree, David Snyder, Saurabh Kataria, Paola García-Perera, Fred Richardson, Pedro A. Torres-Carrasquillo, and Najim Dehak, "The JHU-MIT System Description for NIST SRE19 AV," in *NIST SRE19 Workshop*, 2019.

[32] D. Snyder, D. Garcia-Romero, G. Sell, A. McCree, D. Povey, and S. Khudanpur, "Speaker recognition for multi-speaker conversations using x-vectors," in *ICASSP*, 2019.

[33] Alan McCree, Gregory Sell, and Daniel Garcia-Romero, "Speaker Diarization using Leave-one-out Gaussian PLDA Clustering of DNN Embeddings," in *Interspeech*, 2019.

[34] Yun Lei, Luciana Ferrer, Aaron Lawson, Mitchell McLaren, and Nicolas Scheffer, "Trial-based calibration for speaker recognition in unseen conditions," in *Odyssey*, 2014.

[35] Luciana Ferrer, Mahesh Kumar Nandwana, Mitchell McLaren, Diego Castán, and Aaron Lawson, "Toward Fail-Safe Speaker Recognition: Trial-Based Calibration With a Reject Option," *IEEE TASLP*, 2019.

[36] Luciana Ferrer and Mitchell McLaren, "A discriminative condition-aware backend for speaker verification," *ICASSP*, 2019.

[37] Luciana Ferrer and Mitchell McLaren, "A speaker verification backend for improved calibration performance across varying conditions," *Odyssey*, 2020.