Revisiting SincNet: An Evaluation of Feature and Network Hyperparameters for Speaker Recognition

Dan Oneață, Lucian Georgescu, Horia Cucu, Dragoș Burileanu, Corneliu Burileanu
Speech and Dialogue Research Laboratory
University POLITEHNICA *of Bucharest*
Bucharest, Romania
{lucian.georgescu, dan.oneata}@speed.pub.ro, {horia.cucu, dragos.burileanu, corneliu.burileanu}@upb.ro

***Abstract*—The SincNet architecture [1] was recently introduced as an approach to the speaker recognition task. Its main innovation was the *sinc layer*—an elegant and lightweight way of extracting features from speech. Despite good performance on multiple datasets, little information was provided on the architectural choices. In this work, we aim to shed some light on the importance of the network topology and various hyperparameters. We replace the original network trunk with a lightweight trunk inspired from residual networks (ResNets) and optimize its hyperparameters. Furthermore, we carry an extensive study on the sinc layer's hyperparameters. Our main finding is that the stride and window size of the feature extractor plays a crucial role in obtaining good performance. Further experiments on conventional features, such as MFCCs and FBANKs, yield similar conclusions; in fact, by using optimal values for these two hyperparameters, traditional features are able to match the performance of sinc features. Surprisingly, the best results obtained go against conventional wisdom: an analysis window of only a couple of milliseconds and a stride of only a couple of samples are found to give the best results. Our code is available at https://bitbucket.org/doneata/sincnet.**

***Index Terms*—deep learning, speaker recognition, features, hyperparameter optimization**

I. Introduction

Neural networks have been extensively applied for speaker recognition [1]–[7], but, despite deep learning's promise of learning features from data, most of these approaches still rely on conventional features, such as spectrogram [2]–[4], filter banks (FBANKs) [5], [6] or mel frequency cepstral coefficients (MFCCs) [7]. The SincNet architecture, introduced by Ravanelli and Bengio [1], proposes a novel way of performing speaker recognition in an end-to-end manner, directly from the raw waveform. The main innovation in the SincNet is the *sinc layer*—a bank of convolutional filters which are constrained to take the form of a sinc. Essentially, sinc kernels in the time domain correspond to band-pass filters in the frequency domain. However, differently from traditional feature extraction methods, this approach allows for the cut-off frequencies to be learned during network training.

SincNet has shown good results for speaker verification and identification and, even though it appeared recently, the community has already shown interest in the architecture: Dubey *et al.* [8] successfully transferred the SincNet features to the task of speaker diarisation, Nunes *et al.* [9] improved the speaker recognition results by replacing the cross-entropy loss with the additive margin loss, Ravanelli *et al.* extended the work in multiple directions (task-independent representations [10], visualizations of learned features [11], implementation in the PyTorch-Kaldi toolbox [12]). While promising, we believe the initial work can be further improved.

Ablation study of hyperparameters. The SincNet architecture, as any deep learning architecture, is controlled by a large number of hyperparameters: the sinc features depend on window size, stride, number of filters, while the network trunk depends, among others, on number of layers and number of neurons on each layer. The lack of ablation studies in [1] makes it difficult to tell which hyperparameters are critical for the performance and how they interact with each other. We address this issue by performing *(i)* automatic search across combinations of hyperparameters and *(ii)* an exhaustive search on most important hyperparameters of the sinc features (stride and window size). The first experiment reveals the sensitivity of the network on each of the hyperparameters, while the second one shows interactions of the stride and window size. These experiments were enabled by replacing SincNet's network trunk with a lighter trunk, which also turned out to be more stable and less prone to overfitting.

Evaluation of low-level features. The original paper compares sinc features with convolutional and hand-crafted features, but the evaluation lacks in several regards. The most important issue resides in comparing systems that differ both in terms of features and network trunks. For example, the sinc features are processed by a convolutional network followed by a multi-layer perceptron (MLP), while MFCCs are used with a different MLP architecture; so we cannot draw certain conclusions about impact of the features. A proper evaluation of low-level features is indeed difficult, but we believe such a study to be of interest to the community, especially given the recent literature, which focuses on the latter stages of the speaker recognition pipeline: normalizations of embeddings [6], pooling methods [4], [5], [7], loss functions [5]. The key challenge when evaluating features is the different resolutions at which the features operate—the stride of the feature extractor affects the number of feature vectors and, consequently, the size of the network. Our solution relies on hyperparameter tuning: we keep the same network structure, but optimize the hyperparameters of both features and network trunk.

The paper is structured as follows. Section II reviews related work on low-level features and hyperparameter search. Section III describes the experimental setup, including datasets, an overview of the pipeline and the approached tasks. Section IV presents the results: an analysis of the entire network and an extensive benchmark over multiple features. Finally, section V draws the main conclusions.

II. Related work

Hyperparameters for feature extraction. Hand-crafted acoustic features, such as filter banks (FBANKs) or mel-frequency cepstral coefficients (MFCCs), are predominantly extracted from windows of 25 ms and using a stride (step size) of 10 ms, *e.g.*, [3], [4]. An exception is the work of Lee *et al.* [13], who conducted experiments on shorter windows (3–30 ms) and smaller strides for FBANKs, and have found better results for the finer models. For deep learning approaches that work on the raw waveform there is yet a consensus regarding

This work was partially supported by a grant of the Romanian Ministry of Research and Innovation, CCCDI – UEFISCDI, project number PN-III-P1-1.2-PCCDI-2017-0818 / 73PCCDI, within PNCDI III.

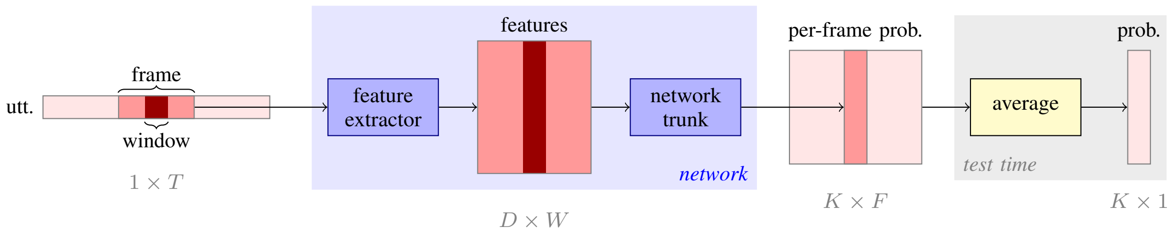

Fig. 1: Overview of the speaker recognition pipeline. Given an input utterance of T samples, the system outputs the probability of it belonging to each of the K speakers. The network operates on 200 ms frames and has two components: *(i)* a feature extractor, which slides a window over the input frame and produces a feature matrix (D features $\times W$ windows); *(ii)* a network trunk, which outputs probabilities based on the features. At test time, the frame is stridden over the entire utterance producing a matrix of probabilities (K speakers $\times F$ frames), which is then averaged over frames (time dimension). Color denotes the temporal resolution: utterance ⬜, frame 🟥, window ■.

the window parameters—some authors resorted to more conventional values [14], [15], while others preferred smaller sizes [13], [16]–[18]. Among the latter, Palaz *et al.* [16], [17] used convolutional neural networks (CNNs) for speech recognition and found the best window size to be around 2–4 ms (50 samples) and the best stride around 0.6 ms (10 samples). But even shorter windows and strides have been tried: the EnvNet [18]—a CNN for environmental sound classification—uses filters of size 10 (samples) and a stride of 1; and the 3^n-SampleCNN [13]—a VGG-inspired architecture for music classification—uses filters of size 3 throughout the entire network. The SincNet architecture extracts sinc features on windows of size 251 (15 ms, which are typical for audio feature extraction) with a stride of 1 sample.

Hyperparameter tuning for neural networks. Neural networks, especially deep ones, come with a daunting number of hyperparameter choices. In such a situation, domain knowledge or intuition brings little help and we need to resort to automatic procedures, as it has been shown that an automatically-tuned architecture for computer vision beats one tuned by a human expert [19]. Another convincing case for hyperparameter tuning was provided recently by Melis *et al.* [20] in the context of language modeling; the authors have shown that baseline architectures can reach state-of-the-art performance if properly tuned. Methods for hyperparameter tuning range from simple ones, such as grid search and random search (which performs surprisingly well), to more sophisticated ones, which try to model the objective function as a function of the hyperparameters using random forests, Gaussian processes or neural networks [19], [21].

III. Experimental Setup

This section describes the experimental setup used throughout the entire analysis. In order to facilitate a direct comparison, we chose a setup (datasets, pipeline and tasks) similar to the one used in [1].

A. Datasets

We used the following two datasets:

TIMIT [22] comprises sentences read by 630 American speakers. The dataset is traditionally used for automatic speech recognition, but, as proposed in [1], we use it for speaker recognition. There are five audio files per speaker in the train set and three in the test set. Differently from [1], we define a validation set by removing one file for each speaker from the training set. The validation set is essential when performing hyperparameter analysis since we do not want to take decisions on the test set.

LibriSpeech [23] is a freely available speech corpus consisting of read audio books in English. The dataset comprises more than 1000 hours of speech from 2484 speakers. Ravanelli and Bengio [1] created a selection of 12–15 seconds per speaker, which was further used in the experiments. The selection was kindly provided, upon request, by the authors.

Both datasets have a sampling rate of 16 kHz and were preprocessed as described in [1] by normalizing the amplitude and removing the silence intervals (both file ends and mid-file).

B. Pipeline

In this subsection we give an overview of the speaker recognition pipeline and introduce the terms that appear later in the paper. For a visual representation of the concepts, please see figure 1.

We take a similar approach to speaker recognition as in [1]. At train time the network learns to associate fixed-size (200 ms) audio *frames* with speakers—for each frame the network outputs the probability of belonging to each speaker. The network has two components which work as follows: *(i)* the *feature extractor* operates on *windows* which are slid with a *stride* over the frame, producing a matrix of feature vectors and *(ii)* the *network trunk* aggregates these features to produce a per-frame vector of probabilities. The training is done at frame-level by optimizing the cross-entropy loss over speakers.

At test time the system receives a variable-length *utterance* and outputs a vector of probabilities over the speakers. The input utterance is processed frame-by-frame (frames are slid in strides of 10 ms over the utterance) and then the per-frame probabilities are averaged over time. Note that since the frame stride is fixed, by *stride* we always refer to the window stride.

C. Features

The sinc layer introduced in [1] is applied on the raw waveform and acts similarly to a feature extractor: it generates 80-dimensional vectors from windows of 251 samples of speech. In this work we refer to the output of the sinc layer with the term *sinc features*.

Besides sinc features, we also use 26-dimensional mel filter banks (FBANK) and 13-dimensional mel-frequency cepstral coefficients (MFCC). Unless otherwise specified, these traditional features are extracted using the default parameters: window size of 25 ms and stride of 10 ms. We also perform experiments without any features at all, by plugging in the raw waveform directly in the network trunk; see figure 2, for a schematic illustration of the architecture.

D. Tasks

We tackle three related tasks:

Frame-level speaker identification is the task for which the SincNet was optimized [1]. The goal is to classify a given speech frame of 200 ms as belonging to one of the speakers. This fine-grained classification could be useful in the context of speaker diarisation (as used for example in [8]). We report the classification error per frame, *i.e.*, frame error rate (FER).

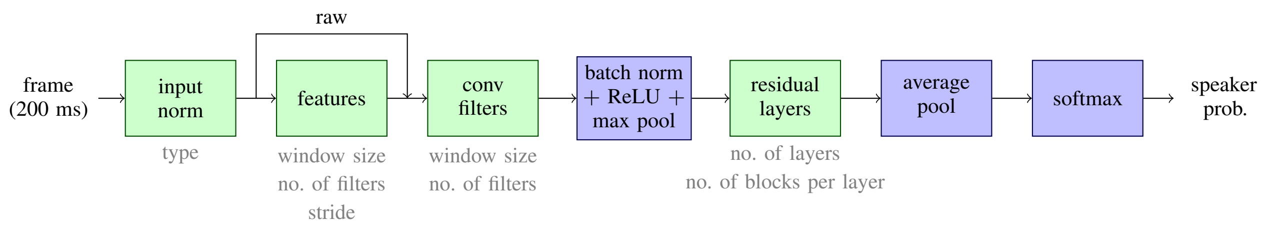

Fig. 2: The network architecture proposed in the paper. The color of each block indicates whether we optimize (green) or not (blue) its corresponding hyperparameters (written in gray underneath). We experiment with various features (sinc, MFCCs and FBANKs) and also with "raw features", which are directly passed to the network trunk.

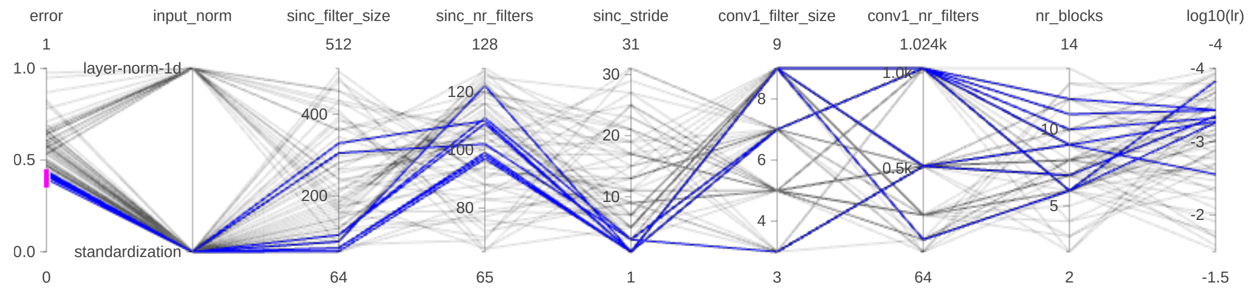

Fig. 3: Results of the hyper-optimization process on TIMIT (validation set). The leftmost axis indicates the frame error rate (FER), while the other axes correspond to hyperparameters. Each line represents a configuration; the highlighted ones achieve FER lower than 45%.

Speaker identification implies classifying a utterance (usually longer than 200 ms) as belonging to one of the speakers. The classification is performed by sliding a 200 ms frame with a stride of 10 ms and then passing it through the network (subsection III-B). The pipeline predicts the speaker with the highest average probability across time. We evaluate speaker identification in terms of the classification error rate (CER) of the utterances.

Speaker verification implies verifying the identity claimed by the speaker who provided the utterance. Again, we run a sliding frame of 200 ms with a stride of 10 ms and feed it to the frame-level network (subsection III-B). The average probability across time for the claimed identity is compared with a threshold. By tuning this threshold one can trade false acceptances for false rejections. The operation point where the two error rates are equal yields the equal error rate (EER), used to report results.

The task of speaker verification usually involves three sets (training, enrollment, testing), with speakers from training differing from those at enrollment and testing time. For our datasets we consider a scenario with only enrollment and testing sets (no training data). This setting is motivated by the fact that both datasets have only a few utterances per speaker (equivalently, tens of seconds per speaker), as it is typically the case at enrollment. In this scenario, the enrollment set is used to *train* the network from scratch. We believe this is a strength of the model since it is not reliant on a large dataset. On the other hand, a limitation of the current methodology is the enrolment of new speakers—adding new speakers requires re-training the network. This drawback is caused by the fixed size of the output layer and it could be mitigated by using, for example, instance-based learning; however, such an approach is out of the scope of this paper.

IV. Analysis

A. Network trunk

The network trunk proposed in the original SincNet paper [1] totals over 22M parameters. Because of its large capacity we found the training unstable and experienced overfitting (end-to-end models for speaker recognition have been found to be sensitive to overfitting [24]). We had trouble replicating the original results (with the code provided by the authors), obtaining a worse performance than what was originally reported—we achieved a frame error rate (FER) of 41.9% on TIMIT, while the paper reported 33.0% FER. Our results are in line with the work of Nunes *et al.* [9], which obtain a FER of 44.64% using the original SincNet architecture.

We observed that the computational bottleneck lies in the fully connected layers following the initial convolutional layers, so we decided to replace those layers with a trunk inspired by the residual networks (ResNet). ResNets have been shown to be among the most efficient architectures in terms of performance per number of parameters. Our decision is also motivated by their state-of-the-art performance on speaker recognition [2], [4]. The proposed architecture is illustrated in figure 2.

An important change to the original ResNet was to perform all operations (*e.g.*, convolution, normalization and pooling) across a single dimension—the temporal dimension. We made this modification because the input signal is one dimensional and we do not expect the activations of nearby filters to be correlated. Prior work on speaker recognition have employed both 2D [3] and 1D convolutions [4], [7], but the trend seems to be towards the latter.

We have also made some updates in terms of training compared to the original paper: *(i)* we used early stopping as opposed to a fixed number of epochs to terminate the optimization procedure; *(ii)* we used a learning rate scheduler to decrease the learning rate if the error did not improve for a number of training steps. Both decisions were taken by monitoring the FER on the validation set.

After carefully choosing the networks' hyperparameters using an automated procedure (see section IV-B), we obtained a FER of 29.5%. Moreover, the network is almost ten times smaller (2.8M parameters) and more stable to train. Having a lighter network trunk allows us to carry an extensive evaluation.

B. Hyperparameters

In order to select a network configuration we run automated hyper-parameter optimization using the tree of Parzen estimators method proposed by Bergstra *et al.* [21]. We use their `hyperopt` package, which allows us to run computations distributed across multiple

		hyperparameters: features				hyperparameters: trunk			FER (%)	
	feature type	choice	window (ms)	stride (ms)	no. filters	choice	res. blocks	no. params	TIMIT	Libri
1	sinc [1]	default	15.7	0.06	80	default	—	22.0M	33.0	—
2	sinc	optim.	5.6	0.06	122	optim.	$[3, 2, 1, 1]$	2.8M	29.5	13.4
3	raw	—	—	—	—	as sinc	$[3, 2, 1, 1]$	2.8M	30.8	13.8
4	FBANK	default	25	10	26	as sinc	$[3, 2, 1, 1]$	2.8M	53.9	36.0
5	MFCC	default	25	10	13	as sinc	$[3, 2, 1, 1]$	2.8M	65.2	46.1
6	raw	—	—	—	—	optim.	$[3, 3, 3, 3]$	6.2M	**25.8**	**12.1**
7	FBANK	optim.	3.1	0.31	*26*	optim.	$[3, 3, 3, 2]$	4.6M	29.7	19.1
8	MFCC	optim.	1.6	0.31	*13*	optim.	$[3, 2, 3, 0]$	1.5M	37.4	34.0

TABLE I: Frame error rate (FER) for speaker identification for multiple features. We show results for *(i)* sinc features (rows 1–2), *(ii)* hand-crafted features using the same network trunk as the sinc features (rows 3–5), *(iii)* hyper-optimized features and architecture (rows 6–8). We did not optimize over the values in italics. The values marked with *optim.* were selected on TIMIT and re-used for Libri(Speech).

machines. The algorithm is set to try 64 parameter combinations and, in order to speed-up the process, we do not train until convergence, but stop after 20 training steps. The selected hyperparameters are:

input normalization The first layer of the network is a pre-processing layer, whose goal is to scale the input data. We tried two variants: *(i)* one-dimensional layer normalization (across the temporal dimension) and *(ii)* standardization, *i.e.*, make the data zero mean and unit variance.

sinc hyperparameters For the sinc layer we optimized the window size, number of filters, and stride. We have included these hyperparameters in the optimization process to obtain a strong initial configuration, but afterwards we analyze in detail the impact of these values.

conv hyperparameters The sinc layer is followed by a convolutional layer for which we tune the window size and the number of filters. The stride is set to be proportional to the window size.

number of residual blocks The bulk of the network consists of residual layers. We use at most four residual layers, each of them consisting of multiple residual blocks. Each residual layer has a fixed number of channels and reduces the temporal dimensionality by half, using strides of length two. The number of channels increases with the layer: 64, 128, 256, and 512. We optimized the number of residual layers and the number of residual blocks (from one to at most four) in each of them.

learning rate The initial learning rate for the optimization algorithm.

Figure 3 shows the results of the optimization procedure: for each combination of hyperparameter we train for 20 steps and monitor the FER. The importance of a careful selection of hyperparameters is emphasized by the error's large variability, between 39.6% and 97.5%. Moreover, the network is more sensitive to some parameters than others—e.g., it is more sensitive to the stride and window size of sinc and less sensitive to window size of the convolutional layer.

Table I lists on row 2 the hyperparameter values obtained after the optimization process and compares them with the sinc layer hyperparameters proposed in [1] (row 1). Our best results were obtained for a bank of shorter, but more numerous sinc filters. Regarding the filter stride, our initial hypothesis was that a stride of 1 sample (0.06 ms) is too small for the 251 samples sinc filter and leads to unnecessary computations. However, the optimization experiments revealed that the stride of 1 sample proposed in [1] is actually the best choice.

Motivated by these results, we perform an exhaustive evaluation of the sinc window size and stride, while keeping the network trunk fixed (figure 4, left). First, we observe that the lower the stride, the better the performance. Second, the network appears to be less sensitive to the window size; while there seems to be an optimum around 4–8 ms, the differences in FER are small. By reducing the sinc window size from 251 samples, as originally proposed, to 31 samples (2 ms), we can improve the computational cost without losing performance.

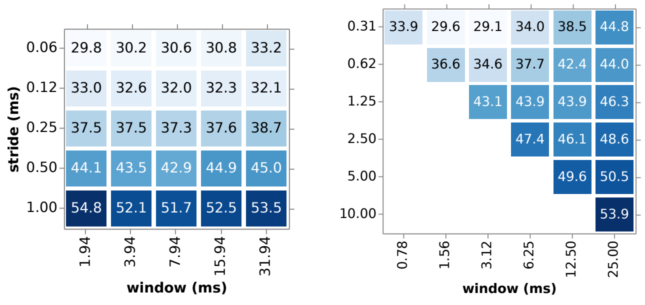

Fig. 4: Frame error rate (FER) on TIMIT (test set) as a function of the stride and window size for sinc (left) and FBANK (right).

C. Features

Ravanelli and Bengio [1] provided a comparison of sinc features with other features, such as FBANKs, MFCCs and raw waveform. Because these features operate at different temporal resolutions, using a general-purpose DNN on all features is inherently difficult. In [1] this issue was overcome by using a different architecture for each feature, but this renders the comparison less conclusive. Our goal was to compare sinc features with other features in a fair setting, by using the same network trunk and changing only the features.

We first combined the conventional features with the network trunk that was optimized for the sinc features. As shown in table I, rows 3–5, the performance for these features is worse than for sinc. However, if we optimize the hyperparameters of both features and network trunk, we obtain visible improvements (rows 6–8 in table I). The results show comparable performances for the feature-based DNNs (25.8–37.4% FER) to the sinc DNN (29.5% FER). Remarkably, the raw features (*i.e.*, no features) show top performance; their optimized version performs best, although the learned network is larger than the others. This result suggests that a powerful network trunk can compensate for the lack of feature extractor.

For classical features (FBANKs and MFCCs) the best results are obtained for windows of 25–50 samples (1.5–3 ms) and strides of 5 samples. We further analyzed the stride and window size for the FBANK features, by looking at multiple combinations while keeping the network trunk fixed (the configuration on row 7 from table I). The results (figure 4, right), indicate the same conclusion as the one we drew for sinc features: smaller windows and strides for classical features yield better results. In order to extract features from smaller windows, we kept the FFT size set at 512 points and zero-padded the

		TIMIT			LibriSpeech		
	features	FER	CER	EER	FER	CER	EER
1	sinc [1]	33.0	0.85	—	—	0.96	0.320
2	sinc	29.5	1.08	0.168	13.4	0.95	0.265
3	raw	**25.8**	0.65	**0.072**	**12.1**	0.68	0.281
4	FBANK	29.7	**0.29**	**0.072**	19.1	0.60	0.281
5	MFCC	37.4	0.65	0.173	34.0	**0.43**	**0.100**

TABLE II: Results of the optimized networks on the three tasks we considered. We report frame (FER), classification (CER), equal (EER) error rates. All figures represent percentages.

windows. Presumably, this generated smoother spectra or, equivalently, feature vectors (FBANKs) with lower frequency resolution.

While rather unexpected, similar conclusions were reported by Lee *et al.* [13] in the context of sound classification. They found better results using finer strides and shorter windows for FBANK features. We speculate that the network performs better with input features at higher temporal resolution (smaller stride) in exchange for a lower frequency resolution. We plan to further investigate these results on other tasks and corpora.

Utterance-level speaker identification and verification. In table II we report results for the optimized networks on the other two tasks, utterance-level identification and verification. Our results on both TIMIT and LibriSpeech improve over what was previously reported [1], but we observe that the CER and EER metrics do not correlate with the FER: *e.g.*, the MFCC-based network obtains better results in terms of CER and EER than the one using sinc features, but performs worse in terms of FER. Recall that the network is trained at frame-level; we would expect better CER and EER performance if the network was to be trained at utterance-level.

V. Conclusions and future work

This paper presented an in-depth analysis of SincNet [1], a new deep architecture for speaker recognition. First, we investigated a ResNet-inspired trunk instead of the original multi-layer perceptron. The results showed that the our trunk is not only lighter (2.8M parameters instead of 22M), but it also improves the performance, from 33.0% to 29.5% FER.

Second, we provided a detailed analysis on the sinc's layer hyperparameters, its stride and window size. Somewhat surprisingly, the finest stride (1 sample or 0.06 ms) was the best choice, while the window size was not critical for good performance—decreasing the window size by a factor of four yields a similar performance, while reducing the computation cost. We also demonstrated the sensitivity of the network to these values and the importance of properly tuning the hyperparameters.

Third, we showed that conventional features fare well when properly optimized. The key components were again the stride and window size, whose optimal values were shorter than what is traditionally used. We believe that understanding the choice of these hyperparameters deserves closer investigation and that the potential findings can impact other speech processing tasks.

Working on TIMIT and LibriSpeech facilitated direct comparison with previous work and rapid experimentation given the datasets' size. We believe a next step forward is extending the study of feature analysis on large-scale datasets, such as VoxCeleb2 [3] or NIST SRE.

References

[1] Mirco Ravanelli and Yoshua Bengio, "Speaker recognition from raw waveform with SincNet," in *IEEE Spoken Language Technology Workshop*, Dec 2018, pp. 1021–1028.

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

[3] Joon Son Chung, Arsha Nagrani, and Andrew Zisserman, "VoxCeleb2: Deep speaker recognition," *arXiv preprint arXiv:1806.05622*, 2018.

[4] Weidi Xie, Arsha Nagrani, Joon Son Chung, and Andrew Zisserman, "Utterance-level aggregation for speaker recognition in the wild," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 5791–5795.

[5] Weicheng Cai, Jinkun Chen, and Ming Li, "Exploring the encoding layer and loss function in end-to-end speaker and language recognition system," *arXiv preprint arXiv:1804.05160*, 2018.

[6] Weicheng Cai, Jinkun Chen, and Ming Li, "Analysis of length normalization in end-to-end speaker verification system," in *Interspeech*, 2018, pp. 3618–3622.

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

[8] Harishchandra Dubey, Abhijeet Sangwan, and John HL Hansen, "Transfer learning using raw waveform SincNet for robust speaker diarization," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6296–6300.

[9] Joao Antônio Chagas Nunes, David Macêdo, and Cleber Zanchettin, "Additive margin SincNet for speaker recognition," *arXiv preprint arXiv:1901.10826*, 2019.

[10] Santiago Pascual, Mirco Ravanelli, Joan Serrà, Antonio Bonafonte, and Yoshua Bengio, "Learning problem-agnostic speech representations from multiple self-supervised tasks," *arXiv preprint arXiv:1904.03416*, 2019.

[11] Mirco Ravanelli and Yoshua Bengio, "Interpretable convolutional filters with SincNet," *arXiv preprint arXiv:1811.09725*, 2018.

[12] Mirco Ravanelli, Titouan Parcollet, and Yoshua Bengio, "The PyTorch-Kaldi speech recognition toolkit," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2019, pp. 6465–6469.

[13] Jongpil Lee, Jiyoung Park, Keunhyoung Kim, and Juhan Nam, "SampleCNN: End-to-end deep convolutional neural networks using very small filters for music classification," *Applied Sciences*, vol. 8, pp. 150, 2018.

[14] Yedid Hoshen, Ron Weiss, and Kevin Wilson, "Speech acoustic modeling from raw multichannel waveforms," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4624–4628.

[15] Tara N Sainath, Ron J Weiss, Andrew Senior, Kevin W Wilson, and Oriol Vinyals, "Learning the speech front-end with raw waveform CLDNNs," in *Interspeech*, 2015.

[16] Dimitri Palaz, Mathew Magimai Doss, and Ronan Collobert, "Convolutional neural networks-based continuous speech recognition using raw speech signal," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2015, pp. 4295–4299.

[17] Dimitri Palaz, Mathew Magimai-Doss, and Ronan Collobert, "Analysis of CNN-based speech recognition system using raw speech as input," in *Interspeech*, 2015.

[18] Yuji Tokozume and Tatsuya Harada, "Learning environmental sounds with end-to-end CNN," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, 2017, pp. 2721–2725.

[19] Jasper Snoek, Hugo Larochelle, and Ryan P Adams, "Practical Bayesian optimization of machine learning algorithms," in *Advances in Neural Information Processing Systems*, 2012, pp. 2951–2959.

[20] Gábor Melis, Chris Dyer, and Phil Blunsom, "On the state of the art of evaluation in neural language models," in *International Conference on Learning Representations*, 2018.

[21] James Bergstra, Daniel Yamins, and David Daniel Cox, "Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures," *Journal of Machine Learning Research*, 2013.

[22] John S Garofolo, Lori F Lamel, William M Fisher, Jonathan G Fiscus, and David S Pallett, "DARPA TIMIT acoustic phonetic continuous speech corpus," 1993.

[23] Vassil Panayotov, Guoguo Chen, Daniel Povey, and Sanjeev Khudanpur, "Librispeech: An ASR corpus based on public domain audio books," in *IEEE International Conference on Acoustics, Speech and Signal Processing*, April 2015, pp. 5206–5210.

[24] Jee-Weon Jung, Hee-Soo Heo, IL-Ho Yang, Hye-Jin Shim, and Ha-Jin Yu, "Avoiding speaker overfitting in end-to-end DNNs using raw waveform for text-independent speaker verification," in *Interspeech*, 2018, pp. 3583–3587.