2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge
16 September 2021, Online

Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection

Wanying Ge, Jose Patino, Massimiliano Todisco and Nicholas Evans

EURECOM, Sophia Antipolis, France

lastname@eurecom.fr

Abstract

End-to-end approaches to anti-spoofing, especially those which operate directly upon the raw signal, are starting to be competitive with their more traditional counterparts. Until recently, all such approaches consider only the learning of network parameters; the network architecture is still hand crafted. This too, however, can also be learned. Described in this paper is our attempt to learn automatically the network architecture of a speech deepfake and spoofing detection solution, while jointly optimising other network components and parameters, such as the first convolutional layer which operates on raw signal inputs. The resulting raw differentiable architecture search system delivers a tandem detection cost function score of 0.0517 for the ASVspoof 2019 logical access database, a result which is among the best single-system results reported to date.

1. Introduction

End-to-end (E2E) solutions are attracting growing attention across a broad range of speech processing tasks [1, 2, 3]. In contrast to the more common approach whereby front-end feature extraction and the back-end classifier or network are separately optimised, E2E solutions allow for pre-processing and post-processing components to be combined within a single network. With both components being encapsulated within a single model, front-end and back-end components can be jointly optimised. In this case the front-end might have a better chance of capturing more discriminative information for the task in hand [4, 5, 6], whereas the back-end might be able to function more effectively upon the information to produce more reliable scores.

Many solutions to anti-spoofing for automatic speaker verification have focused upon the design of deep neural network (DNN) based back-end classifiers. Most combine fixed, hand-crafted features, usually in the form of some spectro-temporal decomposition [7, 8], with a convolutional neural network (CNN) to learn higher-level representations. The literature shows that the use of specially designed network modules [9, 10, 11] and loss functions [12, 13, 14] generally leads to better performing models. Still, their potential is fundamentally dependent upon the information captured in the initial features; information lost in initial feature extraction cannot be recovered. Several works have also shown that the performance of a given model can vary substantially when fed with different features [9, 10, 12]. These observations point toward the importance of learning and optimising not just the higher-level representation, but also the initial features, in unison with the classifier.

E2E solutions have been a focus of our research group for some time [15]. Fundamental to this pursuit is operation upon the raw signal. A recent attempt [5] adopted the RawNet2 architecture [16, 17]. Using a bank of sinc-shaped filters, it operates directly upon the raw audio waveform through time-domain convolution, with the remaining network components being optimised in the usual way. Results show that systems that use automatically learned features are competitive and complementary to systems that use hand crafted features. While these findings are encouraging, improvements to performance are perhaps only modest. Despite the emphasis upon the E2E learning of both features and classifier, one aspect of our model remains hand-crafted [5]. This is also the case for every E2E solution proposed thus far [4, 6, 16]; the network *parameters* are learned, but the network *architecture* is still hand-crafted.

We have hence explored automatic approaches to learn the network architecture as well. Our first attempt [18] was based upon a specific variant of differentiable architecture search [19] known as partially-connected differentiable architecture search (PC-DARTS) [20]. Architecture search is performed using a pair of core network components referred to as cells. Cells are defined by both architecture parameters and network parameters, both of which are jointly optimised during the first of two stages referred to as the *architecture search* stage.

We showed [18] that PC-DARTS learns more compact models that are nonetheless competitive with the state of the art. As the very first attempt to harness the power of differentiable architecture search for anti-spoofing, this work was performed with hand-crafted features. Our latest work has hence sort to combine architecture search with fully E2E learning. In this paper, we present Raw PC-DARTS. It is the first E2E speech deepfake and spoofing detection solution which operates directly upon the raw waveform while allowing for the joint optimisation of both the network architecture and network parameters.

The remainder of the paper is organised as follows. Section 2 introduces the related works. The proposed system is described in Section 3. Reported in Sections 4 and 5 are our experiments and results. Our conclusions are reported in Section 6.

2. Related works

In this section we introduce the two stages of DARTS-based NAS solutions [19, 20, 21], namely the architecture search stage using partial connections [20] and the train from scratch stage.

The architecture search stage aims to determine a base component or building block upon which the full model is constructed. This base component is referred to as a cell. The term *architecture* refers to the configuration of nodes and interconnections within the cell.

As shown in Fig. 1, each cell has a pair of inputs: $\mathbf{x}^{(1)}$ and $\mathbf{x}^{(2)}$. Cells have a single output, denoted by $\mathbf{x}^{(N)}$ ($N = 5$ in Fig. 1). Nodes in between the inputs and output are referred

10.21437/ASVSPOOF.2021-4

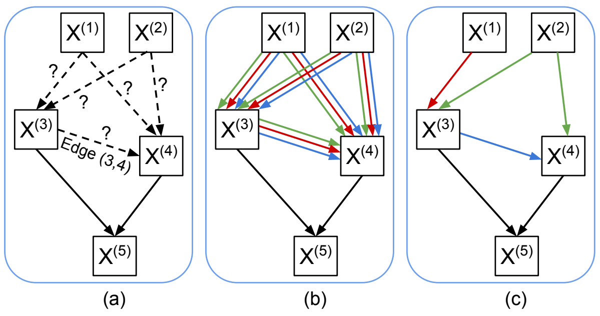


Figure 1: An illustration of architecture search: (a) a neural cell with $N = 5$ nodes; (b) an illustration of the candidate operations performed on each edge that are optimised during architecture search; (c) resulting optimised cell with 2 inputs to each intermediate node.

to as intermediate nodes ($\mathbf{x}^{(3)}$ and $\mathbf{x}^{(4)}$ in Fig. 1). Architecture search involves the selection of candidate operations o from search space $\mathcal{O}$ (solid coloured lines). Operations between intermediate nodes and the output are fixed to concatenation operations (solid black lines). Each intermediate node is calculated according to:

$$\mathbf{x}^{(j)} = \sum_{i<j} o^{(i,j)} \left(\mathbf{x}^{(i)}\right) \tag{1}$$

where $o^{(i,j)}$ is the operation performed on edge (i, j) connecting $\mathbf{x}^{(i)}$ to $\mathbf{x}^{(j)}$. During the architecture search stage, the full set of operation candidates are active, with each being assigned a weight $\alpha_o^{(i,j)}$. The operation performed on edge (i, j) is then defined as:

$$\bar{o}^{(i,j)} \left(\mathbf{x}^{(i)}\right) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o\left(\mathbf{x}^{(i)}\right) \tag{2}$$

When architecture search is complete, only the single operation with the highest weight $\alpha_o^{(i,j)}$ is retained. All other operations are discarded; their weights are set to zero.

Because the set of operation weights $\boldsymbol{\alpha} = \{\alpha^{(i,j)}\}$ are learnable, the search process is a bi-level optimisation problem. We seek to determine the weight parameters $\boldsymbol{\alpha}$ which minimise the validation loss L_{val}, while the set of network parameters $\boldsymbol{\omega}$ is determined by minimising the training loss $L_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha})$:

$$\begin{aligned} &\min_{\boldsymbol{\alpha}} L_{val}(\boldsymbol{\omega}^*, \boldsymbol{\alpha}) \\ &\text{s.t.} \quad \boldsymbol{\omega}^* = \operatorname*{argmin}_{\boldsymbol{\omega}} L_{train}(\boldsymbol{\omega}, \boldsymbol{\alpha}) \end{aligned} \tag{3}$$

The bi-level optimisation process is demanding in terms of GPU memory and computation. Partial channel connections [20] were proposed as a solution to improve efficiency, reducing demands on both computation and memory. A binary masking operator $\mathbf{S}^{(i,j)}$ is used in partially connected (PC) DARTS in order to reduce the complexity of (2). The number of active channels in $\mathbf{x}^{(i)}$ is reduced through either selection (marked as $\mathbf{S}^{(i,j)} = 1$) or masking (marked as $\mathbf{S}^{(i,j)} = 0$) according to:

$$\bar{o}^{(i,j)} \left(\mathbf{x}^{(i)}\right) = \sum_{o\in\mathcal{O}} \frac{\exp\left(\alpha_o^{(i,j)}\right)}{\sum_{o'\in\mathcal{O}} \exp\left(\alpha_{o'}^{(i,j)}\right)} o\left(\mathbf{S}^{(i,j)} \odot \mathbf{x}^{(i)}\right) + \left(1 - \mathbf{S}^{(i,j)}\right) \odot \mathbf{x}^{(i)} \tag{4}$$

where $\odot$ indicates element wise multiplication. In practice, only a number $1/K_C$ of channels in $\mathbf{x}^{(i)}$ are selected. The factor K_C is set as a hyper-parameter and acts to trade off performance (smaller K_C) for efficiency (larger K_C).

After architecture search, the cells are concatenated multiple times (Fig. 2) in similar fashion to a ResNet architecture to produce a deeper, more complex model before being further optimised.

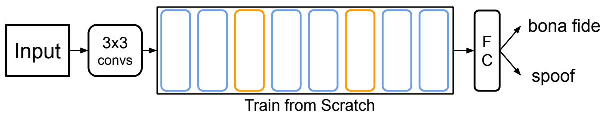


Figure 2: An illustration of train from scratch stage: normal cells (blue) and reduction cells (yellow) are stacked to form a deeper network.

3. Raw PC-DARTS

In this section, we describe the proposed Raw PC-DARTS approach. The model structure is detailed in Table 1. We describe the bank of front-end sinc filters, the application of filter masking, the modifications made to the back-end classifier design and base cell architecture, embedding extraction and the loss function.

3.1. Sinc filters and masking

The input waveform is fixed to a duration of 4 seconds (16000×4 samples) either by concatenation or truncation of source audio data. Feature extraction is performed using a set of C sinc filters [1]. Each filter performs time-domain convolution upon the input waveform. The impulse response of each filter is defined according to:

$$g[n, f_1, f_2] = 2f_2 sinc(2\pi f_2 n) - 2f_1 sinc(2\pi f_1 n) \tag{5}$$

where f_1 and f_2 are the cut in and cut off frequencies, and $sinc(x) = sin(x)/x$ is the sinc function. The cut in and cut off frequencies can be initialised according to any given frequency scale. Both f_1 and f_2 are learnable model parameters, though we consider both learnable and fixed configurations.

Filter masking is applied to mask a number of the sinc filters. This is akin to channel drop-out [22, 23] and frequency masking [13, 24, 25] and acts to encourage the learning of better generalised representations. In practice, sinc filters in the range of $[C_1, C_2)$ are set to zero (masked), where C_1 is the first masked filter selected at random and $C_2 = C_1 + f$. The number of masked filters f is chosen from a uniform distribution $[0, F)$, where F is a pre-defined maximum value. After f is generated, C_1 is then chosen from a uniform distribution $[0, C - f)$.

Table 1: The proposed network structure. Each cell receives outputs of its two previous cells/layers. Conv(k, s, c) stands for a convolutional operation with kernel size k, stride s and output channel c. BN refers to batch normalisation.

Layer	Input:64000 samples	Output shape

Sinc Filters	Conv(128, 1, 64)
	Maxpooling(3)
	BN & LeakyReLU
		(21290, 64)

Conv_1	Conv(3, 2, 64) BN & LeakyReLU	(10645, 64)
Normal Cells	$\left\{\begin{array}{c}\text{BN \& LeakyReLU}\\ \text{Operations}\\ \text{Maxpooling(2)}\end{array}\right\} \times 2$	(2661,256)
Expand Cell	BN & LeakyReLU Operations Maxpooling(2)	(1330, 512)
Normal Cells	$\left\{\begin{array}{c}\text{BN \& LeakyReLU}\\ \text{Operations}\\ \text{Maxpooling(2)}\end{array}\right\} \times 2$	(332, 512)
Expand Cell	BN & LeakyReLU Operations Maxpooling(2)	(166, 1024)
Normal Cells	$\left\{\begin{array}{c}\text{BN \& LeakyReLU}\\ \text{Operations}\\ \text{Maxpooling(2)}\end{array}\right\} \times 2$	(41, 1024)
GRU	GRU(1024)	(1024)
Embedding	FC(1024)	(1024)
Output Score	P2SActivationLayer(2)	(2)

3.2. Search space and cell architectures

In contrast to the approach described in [18] where input features can be seen as a 2D image, operations in Raw PC-DARTS are performed directly upon the raw time-domain waveform. Thus, the search space $\mathcal{O}$ is designed based on 1D convolutional operations, which includes: standard convolution and dilated convolution with kernel size $\{3, 5\}$; max pooling and average pooling with kernel size $\{3\}$; skip connections; no connections.

The original DARTS approach searches for the architectures of two types of cells, namely a normal cell and a reduction cell. The model is formed by stacking these cells sequentially, with the reduction cells being placed at $\frac{1}{3}$ and $\frac{2}{3}$ of the total network depth. While the normal cell preserves the feature map dimension, the reduction cell reduces the dimension by one-half, while the number of channels is doubled. A global average pooling layer is then used after the stacked network to extract embeddings.

This stacked cell design works well for spectro-temporal representations since their dimensions are close to those used typically in image classification tasks to which DARTS was first applied [26, 27]. For speech classification tasks and for solutions that operate upon raw inputs, however, the feature dimension remains large at the stacked cell output and the use of global pooling will result in the substantial loss of information. While a larger number of reduction cells can be added manually to help reduce the feature dimension, this would defeat the purpose of searching the architecture automatically. The introduction of each additional reduction cell also doubles the number of channels, which in turn increases prohibitively both computational complexity as well as demands upon GPU memory.

To address this problem in Raw PC-DARTS, we apply max-pooling to each cell output to reduce the feature dimension by one-half. This simple, yet efficient solution helps the model to learn a more compact, high-level representation, without increasing the number of channels, thereby reducing computational complexity and demands upon GPU memory. An added benefit is that the same architecture depth and initial number of channels can be used for both architecture search as well as train from scratch stages. The so-called *depth gap* [21, 28] is therefore avoided, where the searched operations may not fit the deeper network in the second stage due to the depth mismatch between architecture search and train from scratch stages. Thus, the cells used in Raw PC-DARTS are referred to as a *normal* cell and an *expand* cell. Both cells halve the input feature dimension, whereas only the expand cell doubles the number of channels. Expand cells are placed at the same network depth as reduction cells in the original DARTS approach.

3.3. Embedding extraction and loss function

Frame-level representations produced by the final cell are fed to a gated recurrent unit (GRU) layer to obtain utterance-level representations. These representations are then fed to a fully connected layer which extracts the embedding. We use mean-square error (MSE) for P2SGrad [12] as the loss function. An activation layer is first applied to calculate the cosine distance $\cos\theta$ between the input embedding and the class weight. As in [29], this step is hyper-parameter-free, which reduces the sensitivity of margin-based softmax towards its scale and angular margin parameter settings, thus giving relatively consistent results. The network loss is the MSE between $\cos\theta$ and the target class label. Scores used for performance evaluation are $\cos\theta$ for the bona fide class.

4. Experiments

4.1. Database and metrics

All experiments were performed using the ASVspoof 2019 Logical Access (LA) database [30] which comprises three independent partitions: train, development and evaluation. Each partition is used in the same way reported in [18]. During architecture search, network parameters are updated using 50% of the bona fide utterances and 50% of the spoofed utterances in the training partition. Remaining data is used to update architecture parameters. The cell architectures are selected from those which give the best classification accuracy for the full development partition. During the train from scratch stage, all network parameters, except those of the first convolutional layer, are updated using the full training partition and the best model is selected according to that which gives the best classification accuracy for the full development partition. We report results according to two different metrics: the pooled minimum normalised tandem detection cost function (min-tDCF) [31]; the pooled equal error rate (EER).

4.2. Implementation details

We experimented with 3 different sinc filter frequency scales: Mel, inverse-Mel and linear [5]. We tested two settings in each case, namely *fixed* and *learnable*. Fixed scales are set and left unchanged for both architecture search and train from scratch stages. Learnable scales are initialised in the same way, but the configuration is updated during architecture search. They are then fixed and left unchanged during the train from scratch stage. We also tested a randomly initialised, learnable convolution block denoted Conv_0, in place of sinc filters. The kernel size, stride and the number of output channels for the Conv_0 system are set to the same as that of systems that use sinc filters. The maximum number of masked filters is set to $F = 16$.

Following [18], the number of nodes in each cell is fixed to $N = 7$ and the number of intermediate node inputs is fixed to 2. Models comprise 8 cells (6 normal cells and 2 expand cells) with $C = 64$ initial channels in both stages. During architecture search, we perform 30 epochs of training. In the first 10 designated warm-up epochs, only network parameters are updated. Both architecture parameters and network parameters are updated in the subsequent 20 epochs. In all cases, the batch size is set to 14 and learning is performed using Adam optimisation. Architecture parameters are updated using a learning rate of 6e-4 and a weight decay of 0.001. Network parameters are updated using a learning rate of 5e-5. Partial channel selection is performed with a value of $K_C = 2$. During the train from scratch stage, all models are trained for 100 epochs with a batch size of 32. The initial learning rate of 5e-5 is annealed down to 2e-5 following a cosine schedule.

All models reported in this paper are trained once with the same random seed on a single NVIDIA GeForce RTX 3090 GPU. Architecture search takes approximately 21.5 hours, whereas the train from scratch process takes approximately 9.5 hours. Results are reproducible with the same random seed and GPU environment using the implementation available online[1].

5. Results

First we report a set of experiments which assess the performance of Raw PC-DARTS when using different first layer sinc filter scales. Next, we present a comparison of performance to existing state-of-the-art solutions. Finally, we present an analysis of generalisability in terms of performance stability across different spoofing attacks.

5.1. Raw PC-DARTS with different sinc scales

Table 2 shows results in terms of both the min t-DCF and EER for the ASVspoof 2019 LA evaluation partition. Results are shown for four different sinc scale configurations: Mel; inverse-Mel; linear and with randomly initialised, learnable convolution blocks — Conv_0. With the exception of Conv_0, results in each case are shown for both fixed and learnable configurations.

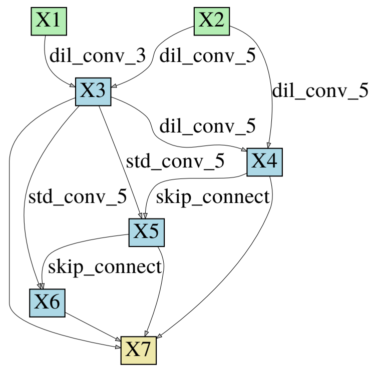
The lowest min t-DCF of 0.0517 (EER of 1.77%) is obtained using fixed Mel scale sinc filters. For both inverse-Mel and linear scales, learnable configurations give better results than fixed configurations, with the second best result with a min t-DCF of 0.0583 (2.1%) being achieved using a linear scale. While the Conv_0 system achieves a respectable EER of 2.49%, the min t-DCF of 0.0733 is notably worse than that of the better performing configurations.

The cell architectures for the best configuration (Mel-Fixed) is illustrated in Fig. 3. We observed that, even though architecture parameters are randomly initialised, after several warm-up epochs, those for dilated convolution operations tend to dominate. This may indicated that, compared to other candidate operations within the search space, dilated convolutions contribute more to representation learning when applied to raw waveforms. Dilated convolutions act to increase the receptive field [6, 32, 33]. The use of greater contextual information then helps to improve performance.

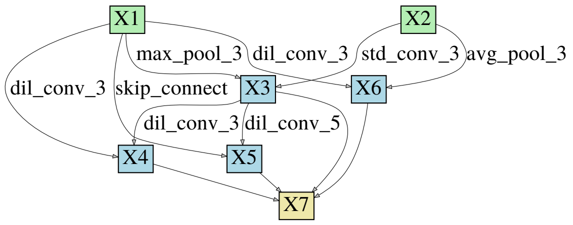
[1]https://github.com/eurecom-asp/raw-pc-darts-anti-spoofing

Table 2: EER results for the ASVspoof 2019 LA database, evaluation partition. Results shown for different Raw PC-DARTS setups using different first layer sinc scale initialisations.

	Fixed		Learnable	
Type	**min-tDCF**	**EER**	**min-tDCF**	**EER**
Mel	0.0517	1.77	0.0899	3.62
Inverse-Mel	0.0700	3.25	0.0655	2.80
Linear	0.0926	3.29	0.0583	2.10
Conv_0	×	×	0.0733	2.49



(a) Normal cell



(b) Expand cell

Figure 3: An illustration of the normal (a) and expand (b) cells produced by the architecture search stage for the Mel-Fixed Raw PC-DARTS configuration.

5.2. Comparison to competing systems

Table 3 shows a comparison of results for the two best performing Raw PC-DARTS systems to that of the top-performing systems reported in the literature[2]. Among the illustrated systems, four operate upon raw inputs, including the top two systems, the first of which is the Res-TSSDNet system reported in [6] and the second of which is the proposed Raw PC-DARTS. The fourth system which operates on the raw waveform is the RawNet2 system reported in [5]. It also uses a first layer of sinc filters, GRU and fully connected layer for embedding extraction.

These results point toward the competitiveness of solutions that operate upon the raw waveform but also show that solutions whose cell architectures are learned automatically can perform almost as well or better that those that are hand-crafted.

[2]Number of learnable parameters and the decomposed EER results for Res-TSSDNet and LCNN-LSTM-sum were obtained using open-source codes available online. Those for Capsule Network were provided by the authors of [34], those for ResNet18-GAT and RawNet2 were provided by the authors of [5, 11].

Table 3: A performance comparison between proposed models and competing state-of-the-art systems reported in the literature. Results for the ASVspoof LA evaluation partition.

Systems	**Features**	**min-tDCF**	**EER**	**Params**	**Worst attack**	**Worst EER**
Res-TSSDNet [6]	waveform	0.0482	1.64	0.35M	A17	6.01
Raw PC-DARTS Mel-F	waveform	0.0517	1.77	24.48M	A08	4.96
ResNet18-LCML-FM [13]	LFB	0.0520	1.81	-	A17	6.19
LCNN-LSTM-sum [12]	LFCC	0.0524	1.92	0.28M	A17	9.24
Capsule Network [34]	LFCC	0.0538	1.97	0.30M	A17	3.76
Raw PC-DARTS Linear-L	waveform	0.0583	2.10	24.40M	A08	6.23
ResNet18-OC-Softmax [14]	LFCC	0.0590	2.19	-	A17	9.22
Res2Net [10]	CQT	0.0743	2.50	0.96M	-	-
ResNet18-AM-Softmax [14]	LFCC	0.0820	3.26	-	A17	13.45
ResNet18-GAT-T [11]	LFB	0.0894	4.71	-	A17	28.02
ResNet18-GAT-S [11]	LFB	0.0914	4.48	-	A17	21.74
PC-DARTS [18]	LFCC	0.0914	4.96	7.51M	A17	30.20
RawNet2 [5]	waveform	0.1294	4.66	25.43M	A18	16.30

5.3. Complexity

The number of network parameters for the systems illustrated in Table 3 is shown in column 5 (where such numbers are available). The two best Raw PC-DARTS architectures have in excess of 24M parameters. For the Mel-Fixed configuration, 77% (18.89M) of the learnable network parameters correspond to GRU layers wereas only 18% (4.52M) correspond to the stacked cells. The RawNet2 system, which also uses a GRU, has over 25M parameters. Other systems have far fewer parameters, including the top Res-TSSDNet system which has 0.35M parameters. It uses ResNet-style 1D convolution blocks and 3 FC layers, without GRUs. The use of dilated convolutions helps to control network complexity while increasing the receptive field [6]. Though the LCNN-LSTM-sum system uses two bidirectional LSTM layers, which is normally computationally expensive, use of a hidden size of 48 nonetheless means that the complexity is the lowest of all illustrated systems. The additional complexity of the Raw PC-DARTS architecture is currently a limitation in the approach, yet a compromise that might be acceptable given that learning and optimisation is a one-step process requiring comparatively little human effort.

5.4. Worst case scenario

Generalisation has been focus of anti-spoofing research since the inception of the ASVspoof initiative. It is well known that even top-performing systems can struggle to detect the full range of spoofing attacks [35]. There is hence interest in minimising not just pooled performance, but also that for the so-called *worst case scenario* which, for the ASVspoof 2019 LA database, is generally the infamous A17 attack.

The worst case attack and corresponding EER for each system is shown in columns 6 and 7 of Table 3. Here we see a distinct advantage of systems that operate upon raw inputs. The Res-TSSDNet [6] and both Raw PC-DARTS systems have among the lowest worse case EERs. This observation indicates that the waveform based systems can capture discriminative artefacts that are missed by systems that use hand-crafted inputs. Were an adversary to discover the attacks to which a system is most vulnerable and exploit only attacks of this nature, then the Raw PC-DARTS countermeasures would offer the second-best protection among all competing systems.

6. Conclusion

In this paper, we proposed an end-to-end differentiable architecture search approach to speech deepfake and spoofing detection, named Raw PC-DARTS. We show that the components of a deep network model, including pre-processing operations, network architecture and parameters, can all be learned automatically from raw waveform inputs and that the resulting system is competitive with the state of the art.

While the best performance is obtained using a fixed front-end, rather than with a learnable configuration, the latter is only marginally behind, while both systems give among the best performance reported to date for the ASVspoof 2019 logical access database. The use of gated recurrent units means that the resulting models are, however, substantially more complex than competing systems and may exhibit some redundancies. While it may be possible to reduce redundancy, and while the results reported in the paper are the first to show the genuine potential of learned architectures, further work to tackle complexity is required if they are to be competitive when computational capacity is limited and a design criteria, e.g. for embedded applications. One avenue for future research in this direction is to evaluate the replacement of gated recurrent units, with a number of parameters in the millions, with concatenated fully connected layers with orders of magnitude fewer parameters.

We also observe that the Raw PC-DARTS solution generalises better to unseen forms of spoofing attacks than their hand-crafted counterparts. Performance for the worst case A17 attack is notably better than that for competing systems. We are currently working to understand what information or cues missed by handcrafted solutions are captured successfully by fully learned solutions. With answers to these questions, we may be able to combine the benefits of both in order to improve reliability further while also protecting complexity.

7. Acknowledgements

This work is supported by the TReSPAsS-ETN project funded from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie grant agreement No.860813. It is also supported by the ExTENSoR project funded by the French Agence Nationale de la Recherche (ANR).

8. References

[1] M. Ravanelli and Y. Bengio, "Speaker recognition from raw waveform with sincnet," *IEEE Signal Processing Letters*, pp. 1021–1028, 2018.

[2] Y. Luo and N. Mesgarani, "Conv-TasNet: Surpassing ideal time–frequency magnitude masking for speech separation," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 27, no. 8, pp. 1256–1266, 2019.

[3] D. Peter, W. Roth, and F. Pernkopf, "End-to-end keyword spotting using neural architecture search and quantization," *arXiv preprint arXiv:2104.06666*, 2021.

[4] H. Dinkel, N. Chen, Y. Qian, and K. Yu, "End-to-end spoofing detection with raw waveform CLDNNS," in *IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2017, pp. 4860–4864.

[5] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2," in *Proc. ICASSP*, 2021, pp. 6369–6373.

[6] G. Hua, A. B.-j. Teoh, and H. Zhang, "Towards end-to-end synthetic speech detection," *IEEE Signal Processing Letters*, 2021.

[7] M. Todisco, H. Delgado, and N. Evans, "A new feature for automatic speaker verification anti-spoofing: Constant Q cepstral coefficients," in *Proc. Speaker Odyssey*, 2016, pp. 283–290.

[8] M. Todisco, X. Wang, V. Vestman, M. Sahidullah, H. Delgado, A. Nautsch, et al., "ASVspoof 2019: Future horizons in spoofed and fake audio detection," in *Proc. INTERSPEECH*, 2019, pp. 1008–1012.

[9] G. Lavrentyeva, S. Novoselov, A. Tseren, M. Volkova, A. Gorlanov, and A. Kozlov, "STC antispoofing systems for the ASVspoof2019 challenge," in *Proc. INTERSPEECH*, 2019, pp. 1033–1037.

[10] X. Li, N. Li, C. Weng, X. Liu, D. Su, D. Yu, and H. Meng, "Replay and synthetic speech detection with Res2Net architecture," in *Proc. ICASSP*, 2021, pp. 6354–6358.

[11] H. Tak, J.-w. Jung, J. Patino, M. Todisco, and N. Evans, "Graph attention networks for anti-spoofing," *Proc. INTERSPEECH*, 2021.

[12] X. Wang and J. Yamagishi, "A comparative study on recent neural spoofing countermeasures for synthetic speech detection," *Proc. INTERSPEECH*, 2021.

[13] T. Chen, A. Kumar, P. Nagarsheth, G. Sivaraman, and E. Khoury, "Generalization of audio deepfake detection," in *Proc. Speaker Odyssey*, 2020, pp. 1–5.

[14] Y. Zhang, F. Jiang, and Z. Duan, "One-class learning towards synthetic voice spoofing detection," *IEEE Signal Processing Letters*, vol. 28, pp. 937–941, 2021.

[15] G. Valenti, H. Delgado, M. Todisco, N. Evans, and L. Pilati, "An end-to-end spoofing countermeasure for automatic speaker verification using evolving recurrent neural networks," in *Proc. Speaker Odyssey*, 2018, pp. 288–295.

[16] J.-w. Jung, H.-s. Heo, J.-h. Kim, H.-j. Shim, and H.-j. Yu, "Rawnet: Advanced end-to-end deep neural network using raw waveforms for text-independent speaker verification," in *Proc. INTERSPEECH*, 2019, pp. 1268–1272.

[17] J.-w. Jung, S.-b. Kim, H.-j. Shim, J.-h. Kim, and H.-j. Yu, "Improved RawNet with feature map scaling for text-independent speaker verification using raw waveforms," in *Proc. INTERSPEECH*, 2020, pp. 1496–1500.

[18] W. Ge, M. Panariello, J. Patino, M. Todisco, and N. Evans, "Partially-connected differentiable architecture search for deepfake and spoofing detection," in *Proc. INTERSPEECH*, 2021.

[19] H. Liu, K. Simonyan, and Y. Yang, "DARTS: Differentiable architecture search," in *Proc. ICML 2019*, 2019.

[20] Y. Xu, L. Xie, X. Zhang, X. Chen, G. Qi, Q. Tian, and H. Xiong, "PC-DARTS: Partial channel connections for memory-efficient architecture search," *8th International Conference on Learning Representations, ICLR*, 2020.

[21] X. Chen, L. Xie, J. Wu, and Q. Tian, "Progressive differentiable architecture search: Bridging the depth gap between search and evaluation," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2019, pp. 1294–1303.

[22] S. Cai, Y. Shu, G. Chen, B. C. Ooi, W. Wang, and M. Zhang, "Effective and efficient dropout for deep convolutional neural networks," *arXiv preprint arXiv:1904.03392*, 2019.

[23] S. Hou and Z. Wang, "Weighted channel dropout for regularization of deep convolutional neural network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2019, vol. 33, pp. 8425–8432.

[24] D. S. Park, W. Chan, Y. Zhang, C.-C. Chiu, B. Zoph, E. D. Cubuk, and Q. V. Le, "SpecAugment: A simple data augmentation method for automatic speech recognition," in *Proc. INTERSPEECH*, 2019.

[25] H. Wang, Y. Zou, and W. Wang, "SpecAugment++: A hidden space data augmentation method for acoustic scene classification," *Proc. INTERSPEECH*, 2021.

[26] J. Deng, W. Dong, R. Socher, L.-j. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2009, pp. 248–255.

[27] A. Krizhevsky, G. Hinton, et al., "Learning multiple layers of features from tiny images," 2009.

[28] A. Yang, P. M. Esperança, and F. M. Carlucci, "NAS evaluation is frustratingly hard," in *International Conference on Learning Representations*, 2020.

[29] X. Zhang, R. Zhao, J. Yan, M. Gao, Y. Qiao, X. Wang, and H. Li, "P2sgrad: Refined gradients for optimizing deep face models," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019, pp. 9906–9914.

[30] X. Wang, J. Yamagishi, M. Todisco, H. Delgado, A. Nautsch, N. Evans, M. Sahidullah, V. Vestman, T. Kinnunen, K. A. Lee, et al., "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *Computer Speech & Language*, vol. 64, pp. 101114, 2020.

[31] T. Kinnunen, H. Delgado, N. Evans, K. A. Lee, V. Vestman, A. Nautsch, M. Todisco, X. Wang, M. Sahidullah, J. Yamagishi, and D. A. Reynolds, "Tandem assessment of spoofing countermeasures and automatic speaker verification: Fundamentals," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 28, pp. 2195–2210, 2020.

[32] K. Tan, J. Chen, and D. Wang, "Gated residual networks with dilated convolutions for supervised speech separation," in *Proc. ICASSP*, 2018.

[33] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," in *4th International Conference on Learning Representations, ICLR*, 2016.

[34] A. Luo, E. Li, Y. Liu, X. Kang, and Z. J. Wang, "A capsule network based approach for detection of audio spoofing attacks," in *Proc. ICASSP*, 2021.

[35] A. Nautsch, X. Wang, N. Evans, T. Kinnunen, V. Vestman, M. Todisco, H. Delgado, M. Sahidullah, J. Yamagishi, and K. A. Lee, "ASVspoof 2019: spoofing countermeasures for the detection of synthesized, converted and replayed speech," *IEEE Transactions on Biometrics, Behavior, and Identity Science*, vol. 3, no. 2, pp. 252–265, 2021.