Capsule-based and TCN-based Approaches for Spoofing Detection in Voice Biometry

Kirill Borodin
Department of Mathematical Cybernetics and Information Technologies, Moscow Technical University of Communications and Informatics, Russia
k.n.borodin@mtuci.ru

Vasiliy Kudryavtsev
Department of Mathematical Cybernetics and Information Technologies, Moscow Technical University of Communications and Informatics, Russia
v.d.kudryavcev@mtuci.ru

Grach Mkrtchian
Department of Mathematical Cybernetics and Information Technologies, Moscow Technical University of Communications and Informatics, Russia
g.m.mkrtchyan@mtuci.ru

Mikhail Gorodnichev
Department of Mathematical Cybernetics and Information Technologies, Moscow Technical University of Communications and Informatics, Russia
m.g.gorodnichev@mtuci.ru (corresponding author)

Received: 3 September 2024 | Revised: 3 October 2024 | Accepted: 9 October 2024

 DOI: https://doi.org/10.48084/etasr.8906

ABSTRACT

Nowadays, deep neural networks are in a phase of rapid development. Simultaneously, the field of biometric forgery is also advancing. Systems that can successfully pass face verification systems are emerging and continuously improving deepfake videos and voice messages are created. These developments can have a negative impact on a person's reputation or cause serious security breaches. This paper proposes an approach for spoofing detection in voice biometrics using the ASVspoof2019 LA dataset The model is trained and validated on subsets representing one type of attack, and evaluated on a subset containing more advanced types of spoofing attacks, demonstrating the model's ability to generalize to more complex attack scenarios. Two models, capsule-based and TCN-based, are proposed, noted as ResCapsGuard and Res2TCNGuard, respectively. ResCapsGuard achieved an Equal Error Rate (EER) value of 2.27, while Res2TCNGuard reached an EER value of 1.49. Notebooks with our models are available in repositories in github. Due to the fact that a random part is cut out of the audio, the results may vary.

Keywords-anti-spoofing; ASVspoof; fake audio; capsules; TCN

I. INTRODUCTION

Spoofing audio can be used to deceive a verification system or to create fake news. The quality of such audio is improving greatly due to the development of deep learning architectures and the increase of training data. This makes the creation of forgeries a relatively simple process. Today, creating spoofing audio requires only a laptop with internet access to find a pretrained model, feed it data, and obtain the desired synthetic speech sample. It is not difficult to generate any passage from a given text using a powerful text-to-speech model (e.g. [1]).

Anti-spoofing models determine whether an utterance is bona fide or spoof. Combining voice verification systems with these models can enhance the security of systems that use voice biometrics. The ASVspoof community is conducting research in this field. In 2013 [2], the collection and development of a database for a unified evaluation of different countermeasure (CM) systems was initiated. In [3], CM systems for synthetic voice detection as well as voice conversion spoofing attacks were developed. In [4], the main focus was on replay attacks. In [5], the dataset consisted of two parts, Physical Access (PA) and Logical Access (LA). The PA subset aims to record and

play back real speech. The LA part of the dataset consists of 19 different types of attacks. The model was trained and validated on 6 different attacks, while the evaluation subset presents 13 different types of audio manipulation. This allows the generalization ability of models to be assessed.

According to the description of the data set, the model's primary criterion is to capture global patterns rather than memorize local ones from training samples. This is why we decided to work on creating lightweight architectures.

In this paper, we propose capsule-based and TCN-based models to detect audio spoofing. Inspired by the work of AASIST [6], we decided to use a sinc-convolutional layer [7] to produce high-level feature maps, along with 6 residual blocks [8]. In capsule architecture, noted ResCapsGuard, ResNet blocks are used in the encoder. This architecture consists of 30 primary capsules, whose output is routed to 2 output capsules using a dynamic routing algorithm. The TCN-based approach, noted Res2TCNGuard, uses Res2Net blocks [9] whose output is converted into spectral and temporal components. The output of both branches is concatenated and fed into fully-connected layers [10].

II. ENCODER

High-level feature-maps were obtained using encoders based on ResNet and Res2Net blocks. Each encoder includes a trainable sinc-conlvolutional layer that can learn to extract important frequency features from the raw waveform.

A. Sinc-Convolutional Layer

Unlike common convolutional layers, where the filter is trained on all data, the sinc-convolutions layer performs waveform convolution using parametrized sinc functions. This layer learns only two parameters from the data: the upper and lower frequencies to be cut. This enables the network to reduce the number of parameters that require training and focus on higher-level features that are important in audio processing.

B. ResNet Blocks

Each of the ResNet blocks consists of two convolutional layers. Before each of them, there is a batch normalization and the SELU activation function. The skip-connection is implemented by adding the original data to the output after two convolutions. If required, the original tensor changes the channel dimension for summation using an additional down sample layer. A maximum pooling operation with kernel size of (1,3) is applied to the obtained result. Due to this size, each block reduces the temporal dimensionality by a factor of three. We used six of these blocks in the encoder.

C. Res2Net Blocks

To extract high-level features, Res2Net blocks were also used, as they are able to process deeper features with approximately the same number of operations. However, this architecture has not been demonstrated to be effective in all cases, as will be explained below. The first layer is a convolutional layer with a kernel size of 1 without padding and a stride equal to 1, after which comes the batch normalization and the ReLU activation function. The output is further split into N parts, denoted as x_i, $i \in \{1,2,...,N\}$. The splitting occurs channel-wise, while preserving the temporal and spectral dimensions. The first split goes directly through without any transformations. The second split passes through a 3×3 convolution, denoted as $K_i()$, and batch normalization. The third and next splits before convolution are summed with the result after the convolution of the previous split and are fed into the next convolution, after which the batch normalization is applied. These operations can be defined by (1):

$$y_i = f(x) = \begin{cases} x_i, \ i = 1 \\ K_i(x_i), \ i = 2 \\ K_i(x_i + y_{i-1}), \ 2 < i \leq N \end{cases} \quad (1)$$

The results are concatenated and are then fed into 1 convolution with batch normalization. Next is the squeeze and excitation block [11]: the result from channel-wise average pooling is passed through the fully-connected layers, with ReLU being after the first and sigmoid after the second. Each number of the original is multiplied by the obtained result. Finally, like ResNet blocks, a skip-connection mechanism is implemented, followed by ReLU and max pooling with the same parameters as in Section II.B.

III. RESCAPSGUARD

This section discusses the architectural specifics of a capsule neural network. The proposed model consists of two levels of capsules: 30 primary and 2 for output. Using exactly this amount was found to be optimal for our model. The encoder comprises ResNet blocks.

A. Primary Capsule Extractor

The encoder output produces an H feature map, $H \in \mathrm{R}^{C \times S \times T}$ which is then fed into the capsule extractor. The first part of the extractor consists of two convolutional layers, the first reduces the number of channels to 64, the second to 16. After each layer there is batch normalization and ReLU. Authors in [12] proposed using statistical pooling to reduce dimensionality in the field of image forgery detection. We used this method because we work with audio forgeries. Between the first and the second parts of the extractor for dimensionality reduction, there is a pooling that calculates the channel-wise mean and standard deviation. The encoder's second part consists of two convolutions. The first increases the channel dimensionality to 8 and the second reduces it to 1. Each layer is followed by batch normalization without activation. The extractor produces 30 capsules, each containing 8 elements.

B. Dynamic Routing

The output from the primary capsules is routed to the output capsules using Algorithm 1.

Algorithm 1 Dynamic Routing algorithm
procedure Routing($u_{j|i}$, W, r)
$\widehat{W} = W + rand(size(W))$
$\hat{u}_{j|i} = Dropout\left(squash(\widehat{W}u_{j|i})\right)$
for all input capsules i and all output capsules j **do:**
$\beta_{ij} \leftarrow 0$
for r iterations **do:**

for all input capsules i **do:**
$c_i \leftarrow softmax(b_i)$
for all input capsules j **do:**
$s_j \leftarrow \sum_i c_{ij}\, \hat{u}_{j|i}$
for all input capsules j **do:**
$v_j \leftarrow squash(s_j)$
for all input capsules i and output capsules j **do:** $\beta_{ij} \leftarrow v_j\, \hat{u}_{j|i}$
end for
return v_j
end procedure

The proposed dynamic routing mechanism includes a special feature: the addition of Gaussian random noise to the weight vector, and the use of dropout to prior vectors $\hat{u}_{j|i}$. These modifications increase the network's generalisation ability. The squash procedure follows the formula proposed in [13]:

$$v_j = \frac{\left||s_j|\right|}{1+\left||s_j|\right|^2}\frac{s_j}{\left||s_j|\right|} \quad (2)$$

C. Implementation Details

The model was trained using weighted cross-entropy loss. A weight of 0.9 was applied to bona fide speech and one of 0.1 to spoofed speech. For the training, the AdaBound optimizer [14] was used with a learning rate of 10^{-4} and a weight decay of 10^{-5}. The number of iterations r for dynamic routing mechanism was chosen empirically to be 2. The learning rate was reduced by half every 10 epochs during the 25-epoch training period with a batch size of 32. Training was done on an NVIDIA Tesla A100 using PyTorch framework. The model has approximately 1.6 million parameters. However, training for one epoch takes around 6 minutes whereas the validation takes 1.5 minutes. The full training took about 3 hours.

D. Architectural Specifics

The EER and t-DCF metrics are unstable (as shown in Figure 1), they are not relatively monotonically decreasing, they cyclically rise and fall sharply, while the loss is roughly monotonically decreasing. Adding the attention mechanism between capsules proposed in [15] with different hidden layer sizes worsen the result, the EER at validation varied in the vicinity of 40. Reducing the number of capsules with attention mechanism didn't solve anything either. Changing the encoder blocks from ResNet to Res2Net made the metrics more stable with no sudden spikes (Figure 2). However, a problem was raised: there is a sharp jump in metrics at the beginning of the training, which decreases monotonically after the second epoch. After training, the final model in this case no longer performs well on the evaluation subset with an EER of approximately 4.57. Adding Squeeze-Excitation to all Res2Net blocks worsened the results and the model lost stability (Figure 3). The EER value was initially low but sharply increased after the first epoch. Afterward, it slightly decreased, but at the 13th epoch, the model stopped working. The reason for this phenomenon was not found, and further use of exactly the same blocks did not cause similar problems.

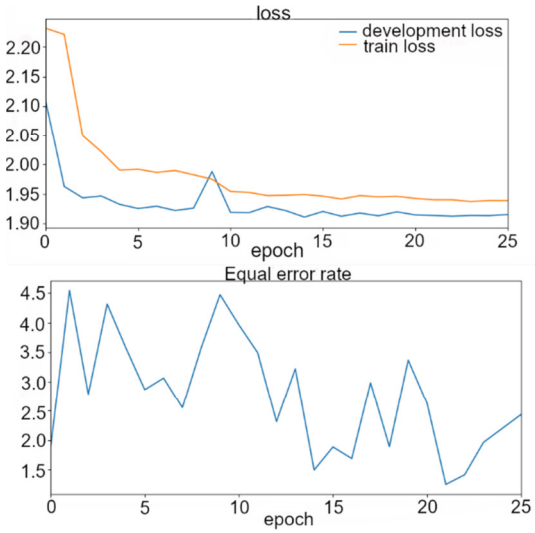

Fig. 1. Loss and EER of the best model.

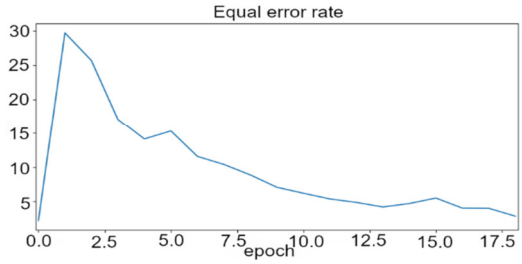

Fig. 2. EER of the model with changed encoder blocks from ResNet to Res2Net.

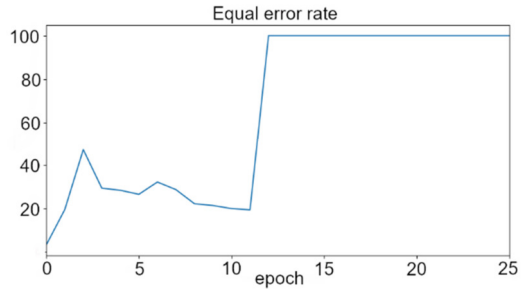

Fig. 3. EER of the model with SE in Res2Net blocks.

Experiments with varying primary capsule extractor did not result in improved outcomes. The performance of the capsule network was degraded in all configurations when more advanced encoders were used. The model architecture proposed in this paper was empirically found to be the optimal solution.

IV. RES2TCNGUARD

This section discusses the architecture of the second considered model, which was based on TCN. The converting of feature maps into temporal and spectral components is inspired by AASIST. However, we suggest a TCN approach instead of graph processing. The Res2Net blocks described above were used as encoder blocks.

A. Branching Mechanism

The output of the encoder is the feature map H, $H \in \mathrm{R}^{C \times S \times T}$. Two matrices are created for the spectral and the temporal components. This transformation is done by maximum selection of the corresponding dimensions. Unlike AASIST, we propose to use a non-absolute maximum:

$$mat_s = \max_t H, \quad mat_s \in \mathbb{R}^{C \times S} \tag{3}$$

$$mat_t = \max_s H, \quad mat_{\mathrm{e}} \in \mathbb{R}^{C \times T} \tag{4}$$

The matrices are then fed into their corresponding TCN modules.

B. TCN-Module Configuration

According to [31], each TCN block consists of two dilated convolutional layers with batch normalization, ReLU and dropout. The dilated convolution operation for an input sequence x, $x \in \mathrm{R}^N$ and weight w_i, $i \in \{0,\ldots,k-1\}$ is:

$$y_t = \sum_{i=0}^{k-1} w_i \cdot x_{t-i*d} \tag{5}$$

where k is the filter size, in our case 2, d is the dilation factor, and $t - i * d$ represents past transformations. A padding equal to $(k-1) * d$ is used and trimmed after one transformation to further exclude its influence. The weights are initialized based on the normal distribution with a mean of 0 and a standard deviation of 0.01.

One block also utilizes residual connections, where the results obtained before applying dilated convolution are added to the results obtained after it. The output is processed through the ReLU activation function. A TCN module comprises five levels of blocks, with the dilation size increasing exponentially for each level.

$$d_i = 2^i, i \in \{0, \ldots, 4\} \tag{6}$$

C. Result Processing after the TCN Modules

Each TCN-module is followed by a fully-connected layer, dropout, and ReLU activation. The results are concatenated and then fed into two fully-connected layers with an ReLU between them. The output provides logits for both spoof and bona fide speech. To evaluate the model, we considered the output of genuine speech. Thus, it should be maximal for bona fide and minimal for spoof.

D. Implementation Details

Weighted cross-entropy loss was used with weights of 0.1 and 0.9 for spoof and bona fide speech, respectively. The model was trained using the Adam optimizer with a learning rate of 10^{-4}. The batch size was selected as 28.

The training consisted of 75 epochs using an NVIDIA Tesla A100, each epoch lasting approximately 20 minutes. The model has around 172k parameters and the total training time was 25 hours.

E. Architectural Specifics of Res2TCNGuard

As previously stated, the maximum value is used to obtain the spectral and temporal sequences. At the same time, the use of mean and minimum did not improve the results. We tried adding branches for the mean and minimum, but this was also unsuccessful. Other frontends improved the results greatly on the development subset, but were much worse on the evaluation. Unlike AASIST, Res2Net has empirically proven to be the best solution when using 4 branches. Scaling to the depth of the TCN-blocks didn't improve the metrics either.

V. EXPERIMENTS AND RESULTS

This section describes the dataset used to train, validate, and evaluate the models. It also presents the metrics used and compares the results with those of other models.

A. Dataset and Metrics

Models were trained in the ASVspoof 2019 LA (Logical Access) training subset [28]. This dataset contains 10256 records of real data and 90192 records of spoofed data with a sampling rate 16 kHz. There were applied TTS (Text-To-Speech) and VC (Voice Conversion) spoofing. The models were evaluated on the development subset. The models were evaluated using the EER and the minimum tandem Detection Cost Function (min t-DCF). Min t-DCF evaluates the quality of CM systems rather than their contribution to the reliability of Automatic Speaker Verification (ASV) systems during spoofing attacks.

B. Results

Table I shows the comparison of the results of the spoofing speech detection systems on the ASVspoof2019 LA dataset. According to [30], we find it inappropriate to use deep learning models when their EER is greater than 5. The same result can be obtained by exploring the features of the data in the dataset and processing them using classical machine learning methods.

The ResCapsGuard outperformed the AdaBoostClassifier by approximately 2.2 times and also outperformed various front-end-based models. Furthermore, the model has a relatively low convergence threshold of 2 epochs, which is just slightly higher than those in [22, 25]. In addition, the model didn't exceed the 2 EER barrier and is not the best solution from a metric point of view. Res2TCNGuard gave better results. Among the models selected for comparison, the proposed approach is just behind graph networks and AASIST-based approaches [6, 8, 17, 18] and CNBNN(MECA) [16].

VI. CONCLUSION

In this paper, we described two proposed approaches for the task of audio spoofing detection on the ASVspoof2019 LA dataset. The proposed approaches have their own advantages and disadvantages. The ResCapsGuard is trained relatively fast, the number of epochs required for convergence is 25, with a total training time of approximately 3 hours, which we consider to be fast enough. The model showed a result of EER equal to 2.27. The Res2TCNGuard showed better EER results, despite the fact that it takes longer to train (75 epochs, 25 hours). Both approaches provide a complete end-to-end solution for spoofing detection. The models can be found in [32] and [33].

TABLE I. RESULTS OF SPOOFING SPEECH DETECTION SYSTEMS ON THE ASVSPOOF2019 LA DATASET

System	Front-end	Architecture	min t-DCF	EER (%)	Epochs
[16]	Raw waveform	CNBNN(MECA)	0.0187	0.64	50
[17]	Raw waveform and power spectrogram	S^2pecNet	0.025	0.77	100
[6]	Raw waveform	AASIST	0.0275	0.83	-
[18]	Raw waveform	AASIST-SAMO	0.0356	0.88	100
[6]	Raw waveform	AASIST-L	0.0309	0.99	-
[8]	Raw waveform	RawGAT-ST	0.0335	1.06	300
Proposed	**Raw waveform**	**Res2TCNGuard**	**0.0457**	**1.49**	**75**
[19]	FastAudio-Tri	ECAPA-TDNN	0.0451	1.54	100
[20]	Raw waveform	Res-TSSDNet	0.0482	1.64	100
[21]	Raw waveform	Raw PC-DARTS	0.0517	1.77	100
[22]	CQT	MCG-Res2Net50+CE	0.052	1.78	20
[23]	LFCC	LCNN-LSTM	0.0524	1.92	-
[24]	LFCC	ResNet18-OS-softmax	0.059	2.19	100
Proposed	**Raw waveform**	**ResCapsGuard**	**0.0744**	**2.25**	**25**
[25]	CQT	SE-Res2Net50+CE	0.0743	2.502	20
[26]	LFCC	GMM	0.0904	3.50	-
[27]	LFB	ResNet18-GAT-T	0.0894	3.50	300
[28]	LFCC	PC-DARTS	0.0914	4.96	50
[29]	MFCC, IMFCC and others	AdaBoostClassifier	-	5	-

REFERENCES

[1] V. Popov, I. Vovk, V. Gogoryan, T. Sadekova, and M. Kudinov, "Grad-TTS: A Diffusion Probabilistic Model for Text-to-Speech." arXiv, Aug. 05, 2021, https://doi.org/10.48550/arXiv.2105.06337.

[2] N. Evans, T. Kinnunen, and J. Yamagishi, "Spoofing and countermeasures for automatic speaker verification," in *Interspeech 2013*, Lyon, France, Aug. 2013, pp. 925–929, https://doi.org/10.21437/Interspeech.2013-288.

[3] Z. Wu *et al.*, "ASVspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," presented at the Proc. Interspeech 2015, 2015, pp. 2037–2041, https://doi.org/10.21437/Interspeech.2015-462.

[4] T. Kinnunen *et al.*, "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection," in *Interspeech 2017*, Stockholm, Sweden, Aug. 2017, pp. 2–6, https://doi.org/10.21437/Interspeech.2017-1111.

[5] X. Wang *et al.*, "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech," *arXiv e-prints*. Nov. 01, 2019, https://doi.org/10.48550/arXiv.1911.01601.

[6] J. Jung *et al.*, "AASIST: Audio Anti-Spoofing using Integrated Spectro-Temporal Graph Attention Networks." arXiv, Oct. 04, 2021, https://doi.org/10.48550/arXiv.2110.01200.

[7] M. Ravanelli and Y. Bengio, "Speaker Recognition from Raw Waveform with SincNet." arXiv, Aug. 09, 2019, https://doi.org/10.48550/arXiv.1808.00158.

[8] H. Tak, J. Patino, M. Todisco, A. Nautsch, N. Evans, and A. Larcher, "End-to-end anti-spoofing with RawNet2." arXiv, Dec. 16, 2021, https://doi.org/10.48550/arXiv.2011.01108.

[9] S.-H. Gao, M.-M. Cheng, K. Zhao, X.-Y. Zhang, M.-H. Yang, and P. Torr, "Res2Net: A New Multi-Scale Backbone Architecture," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 43, no. 2, pp. 652–662, Feb. 2021, https://doi.org/10.1109/TPAMI.2019.2938758.

[10] A. A. Alasadi, T. H. Aldhayni, R. R. Deshmukh, A. H. Alahmadi, and A. S. Alshebami, "Efficient Feature Extraction Algorithms to Develop an Arabic Speech Recognition System," *Engineering, Technology & Applied Science Research*, vol. 10, no. 2, pp. 5547–5553, Apr. 2020, https://doi.org/10.48084/etasr.3465.

[11] J. Hu, L. Shen, S. Albanie, G. Sun, and E. Wu, "Squeeze-and-Excitation Networks." arXiv, May 16, 2019, https://doi.org/10.48550/arXiv.1709.01507.

[12] H. H. Nguyen, J. Yamagishi, and I. Echizen, "Capsule-Forensics: Using Capsule Networks to Detect Forged Images and Videos." arXiv, Oct. 26, 2018, https://doi.org/10.48550/arXiv.1810.11215.

[13] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic Routing Between Capsules." arXiv, Nov. 07, 2017, https://doi.org/10.48550/arXiv.1710.09829.

[14] L. Luo, Y. Xiong, Y. Liu, and X. Sun, "Adaptive Gradient Methods with Dynamic Bound of Learning Rate." arXiv, Feb. 26, 2019, https://doi.org/10.48550/arXiv.1902.09843.

[15] Z. Xinyi and L. Chen, "Capsule Graph Neural Network," in *Seventh International Conference on Learning Representations*, New Orleans, LA, USA, Dec. 2019, pp. 1–16.

[16] Q. Ma, J. Zhong, Y. Yang, W. Liu, Y. Gao, and W. W. Y. Ng, "ConvNeXt Based Neural Network for Audio Anti-Spoofing." arXiv, Dec. 22, 2022, https://doi.org/10.48550/arXiv.2209.06434.

[17] P. Wen, K. Hu, W. Yue, S. Zhang, W. Zhou, and Z. Wang, "Robust Audio Anti-Spoofing with Fusion-Reconstruction Learning on Multi-Order Spectrograms." arXiv, Aug. 18, 2023, https://doi.org/10.48550/arXiv.2308.09302.

[18] S. Ding, Y. Zhang, and Z. Duan, "SAMO: Speaker Attractor Multi-Center One-Class Learning for Voice Anti-Spoofing." arXiv, Nov. 04, 2022, https://doi.org/10.48550/arXiv.2211.02718.

[19] Q. Fu, Z. Teng, J. White, M. Powell, and D. C. Schmidt, "FastAudio: A Learnable Audio Front-End for Spoof Speech Detection." arXiv, Sep. 06, 2021, https://doi.org/10.48550/arXiv.2109.02774.

[20] G. Hua, A. B. J. Teoh, and H. Zhang, "Towards End-to-End Synthetic Speech Detection," *IEEE Signal Processing Letters*, vol. 28, pp. 1265–1269, 2021, https://doi.org/10.1109/LSP.2021.3089437.

[21] W. Ge, J. Patino, M. Todisco, and N. Evans, "Raw Differentiable Architecture Search for Speech Deepfake and Spoofing Detection." arXiv, Oct. 06, 2021, https://doi.org/10.48550/arXiv.2107.12212.

[22] X. Li, X. Wu, H. Lu, X. Liu, and H. Meng, "Channel-wise Gated Res2Net: Towards Robust Detection of Synthetic Speech Attacks." arXiv, Jul. 19, 2021, https://doi.org/10.48550/arXiv.2107.08803.

[23] X. Wang and J. Yamagishi, "A Comparative Study on Recent Neural Spoofing Countermeasures for Synthetic Speech Detection." arXiv, Jun. 13, 2021, https://doi.org/10.48550/arXiv.2103.11326.

[24] 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, https://doi.org/10.1109/LSP.2021.3076358.

[25] X. Li *et al.*, "Replay and Synthetic Speech Detection with Res2net Architecture." arXiv, Feb. 13, 2021, https://doi.org/10.48550/arXiv.2010.15006.

[26] H. Tak, J. Patino, A. Nautsch, N. Evans, and M. Todisco, "Spoofing Attack Detection using the Non-linear Fusion of Sub-band Classifiers." arXiv, May 20, 2020, https://doi.org/10.48550/arXiv.2005.10393.

[27] H. Tak, J. Jung, J. Patino, M. Todisco, and N. Evans, "Graph Attention Networks for Anti-Spoofing." arXiv, Apr. 08, 2021, https://doi.org/10.48550/arXiv.2104.03654.

[28] J. Yamagishi *et al.*, "ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge database." University of Edinburgh. The Centre for Speech Technology Research (CSTR), Jun. 04, 2019, https://doi.org/10.7488/ds/2555.

[29] W. Ge, M. Panariello, J. Patino, M. Todisco, and N. Evans, "Partially-Connected Differentiable Architecture Search for Deepfake and Spoofing Detection." arXiv, Jun. 30, 2021, https://doi.org/10.48550/arXiv.2104.03123.

[30] S. Borzi, O. Giudice, F. Stanco, and D. Allegra, "Is synthetic voice detection research going into the right direction?," in *IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, New Orleans, LA, USA, Jun. 2022, pp. 71–80, https://doi.org/10.1109/CVPRW56347.2022.00017.

[31] S. Bai, J. Z. Kolter, and V. Koltun, "An Empirical Evaluation of Generic Convolutional and Recurrent Networks for Sequence Modeling." arXiv, Apr. 19, 2018, https://doi.org/10.48550/arXiv.1803.01271.

[32] "mtuciru/ResCapsGuard." MTUCI, Sep. 10, 2024, [Online]. Available: https://github.com/mtuciru/ResCapsGuard.

[33] "mtuciru/Res2TCNGuard." MTUCI, Sep. 10, 2024, [Online]. Available: https://github.com/mtuciru/Res2TCNGuard.