The Automatic Speaker Verification Spoofing Countermeasures Workshop (ASVspoof 2024)
31 August 2024, Kos, Greece

AASIST3: KAN-Enhanced AASIST Speech Deepfake Detection using SSL Features and Additional Regularization for the ASVspoof 2024 Challenge

Kirill Borodin[2,*], *Vasiliy Kudryavtsev*[2,*], *Dmitrii Korzh* [1,3,*], *Alexey Efimenko*[2,*,†]
Grach Mkrtchian [2], *Mikhail Gorodnichev* [2], *Oleg Y. Rogov*[1,3]
[1]AIRI, [2]MTUCI, [3]Skoltech, Moscow, Russia
korzh@airi.net

Abstract

Automatic Speaker Verification (ASV) systems, which identify speakers based on their voice characteristics, have numerous applications, such as user authentication in financial transactions, exclusive access control in smart devices, and forensic fraud detection. However, the advancement of deep learning algorithms has enabled the generation of synthetic audio through Text-to-Speech (TTS) and Voice Conversion (VC) systems, exposing ASV systems to potential vulnerabilities. To counteract this, we propose a novel architecture named AASIST3. By enhancing the existing AASIST framework with Kolmogorov-Arnold networks, additional layers, encoders, and pre-emphasis techniques, AASIST3 achieves a more than twofold improvement in performance. It demonstrates minDCF results of 0.5357 in the closed condition and 0.1414 in the open condition, significantly enhancing the detection of synthetic voices and improving ASV security.

1. Introduction

Automatic Speaker Verification (ASV) systems are designed to identify speakers based on their voice characteristics. These systems have a variety of applications, including in the financial sector for user authentication during transactions, in smart devices to ensure exclusive access for the owner to control their equipment, and in forensic analysis to detect fraud cases.

However, the advent of deep learning algorithms has rendered ASV systems susceptible to many assaults. The public accessibility of Text-to-Speech (TTS) and Voice Conversion (VC) systems with pre-trained weights permits any user with access to computational resources, including cloud GPUs, to refine these models for potentially malevolent objectives.

To effectively counter such attacks, the implementation of anti-spoofing systems is imperative. The ASVSpoof community is engaged in active research in this field, as evidenced by the compilation of diverse data corpora for the development of both countermeasure (CM) systems and spoofing-aware speaker verification (SASV) systems. This research is documented in the following publications: [1, 2, 3, 4, 5]. Moreover, the Singfake dataset [6] was developed to detect AI-generated vocals within the musical domain. Additionally, the SVDD project has significantly contributed to advancing voice spoofing countermeasures.

Today, many techniques are utilized to detect voice spoofing, including those based on Convolutional Neural Networks (CNN) [7, 8], ResNet-like architectures [9, 10, 11, 12], Time Delay Neural Networks (TDNN) [13, 14], and transformers [15]. The AASIST architecture [16] has demonstrated particular robustness, as confirmed by numerous studies. Various modifications have been proposed to enhance the generalization capability of AASIST, including the use of a Res2Net encoder [17], wav2vec [18], fusion of different audio representations [19], application of specific training schemes such as SAM [20], ASAM [21], SWL [22], as well as the use of alternative loss functions [23, 24].

In the present study, we propose an innovative architecture, AASIST3, developed on an AASIST base to detect speech deepfakes. The main modifications include:

- The modification of attention layers in GAT, GraphPool, and HS-GAL utilizing KAN[25] is based on the primary PreLU activation function and learnable B-splines, allowing for the extraction of more relevant features.
- The scaling of the model in width using the proposed KAN-GAL, KAN-GraphPool, and KAN-HS-GAL techniques enabled the extraction of more complex parameters, resulting in enhanced model performance.
- The primary methods of data pre-processing employed were diverse augmentations and pre-emphasis, intending to obtain more meaningful discriminative frequency information.

2. Preliminaries

2.1. Kolmogorov-Arnold Network

The Kolmogorov-Arnold theorem [26, 27, 28] postulates that any continuous multivariate function $f : [0, 1]^n \to \mathbb{R}$ can be represented as a finite composition of continuous unary functions and a binary addition operation. More specifically:

$$f(x) = f(x_1, x_2, ..., x_n) = \sum_{q=0}^{2n} \Phi_q \sum_{p=1}^{2n} \phi_{q,p}(x_p), \quad (1)$$

where $\phi_{q,p} : [0, 1] \to \mathbb{R}$ and $\Phi_q : \mathbb{R} \to \mathbb{R}$

As all the functions to be learned by the model are one-dimensional, Liu et al.[25] proposed that each 1D function be parameterized as a B-spline curve and a basis function:

$$\phi(x) = w_b b(x) + w_s \,\mathrm{spline}(x), \quad (2)$$

where $b(x)$ represents the local basis function(eq. 3), while w_b and w_s denote trainable parameters that have been initialized following Kaiming initialization.

$$b(x) = \mathrm{PReLU}(x) = \max(0, x) + a \cdot \min(0, x) \quad (3)$$

where a is the trainable parameter, $\mathrm{spline}(x)$ - linear combination of B-splines:

$$\mathrm{spline}(x) = \sum_{i=0}^{4} c_i B_i(x) \quad (4)$$

∗Equal contribution.
† Contributed during internship in AIRI.

10.21437/ASVspoof.2024-8

where c_i represents the trainable parameter and B_i denotes the unique spline. For each spline of order $\beta_d = 4$, a total of G points were utilized:

$$G = 2\beta_d + \beta_N + 1, \tag{5}$$

where grid size β_N is set to 16. The points are located on the interval $[\theta_1, \theta_2]$ eq.6, 7

$$\theta_1 = -\beta_d h + \alpha_1 \tag{6}$$

$$\theta_2 = (N + \beta_d + 1)h + \alpha_1 \tag{7}$$

$$h = \frac{\alpha_2 - \alpha_1}{\beta_N}, \tag{8}$$

where $[\alpha_1, \alpha_2]$ is the grid range. We set $\alpha_2 = 1$ and $\alpha_1 = -1$.

A KAN layer with input dimensionality n_{in} and output dimensionality n_{out} can be represented as a matrix of one-dimensional functions:

$$\Phi\{\phi_{q,p}\}, p = \{1, 2, ..., n_{in}\}, q = \{1, 2, ..., n_{out}\}. \tag{9}$$

In matrix form, the KAN layer can be expressed as follows:

$$\mathbf{x}_{l+1} = \underbrace{\begin{pmatrix} \phi_{1,1}(\cdot) & \phi_{1,2}(\cdot) & \cdots & \phi_{1,n_l}(\cdot) \\ \phi_{2,1}(\cdot) & \phi_{2,2}(\cdot) & \cdots & \phi_{2,n_l}(\cdot) \\ \vdots & \vdots & & \vdots \\ \phi_{n_{l+1},1}(\cdot) & \phi_{n_{l+1},2}(\cdot) & \cdots & \phi_{n_{l+1},n_l}(\cdot) \end{pmatrix}}_{\mathbf{\Phi}_l} \mathbf{x}_l, \tag{10}$$

Where Φ is the matrix of the function of the KAN layer. Thus, the KAN layer can be denoted as:

$$\mathrm{KAN}(X) = \Phi X. \tag{11}$$

2.2. Audio preprocessing

In light of the hypothesis that high frequencies facilitate the model's ability to differentiate between bona fide and spoof utterances, we employed a pre-emphasis technique on the input signal:

$$x_l = x_l - 0.97 \cdot x_{l-1}, \tag{12}$$

where l equals 1, 2, 3, .., L, L represents the length of the audio signal, and 0.97 is the pre-emphasis factor. The pre-emphasis process suppresses low and enhances high frequencies, facilitating the model's ability to focus on more relevant features specific to spoofing or bona fide utterances.

2.3. SincConv frontend

Following AASIST[16], we use the non-trainable SincConv[29] to extract features from preprocessed audio. SincConv applies the function $g(n, f_1, f_2)$ to the speech signal chunks $x(n)$ using the Hamming window function w_n:

$$y(n) = x(n) \cdot g(n, f_1, f_2) * w(n) \tag{13}$$

$$g(n, f_1, f_2) = 2f_2 \operatorname{sinc}(2\pi f_2 n) - 2f_1 \operatorname{sinc}(2\pi f_1 n) \tag{14}$$

$$\operatorname{sinc}(x) = \frac{\sin(x)}{x} \tag{15}$$

$$w(n) = 0.54 - 0.46 \cos \frac{2\pi n}{L}, \tag{16}$$

where f_1, f_2 are fixed parameters equal to the minimum and maximum possible frequencies in the Mel-spectrogram of the passed signal

2.4. Wav2Vec2 frontend

Wav2Vec2 [30], developed by Facebook AI, is a state-of-the-art method for converting audio to text. This model utilizes the Transformer architecture first introduced in [31]. The Wav2Vec2 [30] architecture consists of two main components: Encoder Layer: This layer transforms the input audio data into a sequence of hidden states. It consists of convolutional layers that transform the input audio data into a sequence of hidden states. Predictor Layer: This layer takes the sequence of hidden states from the encoder layer and predicts the next hidden state. It uses the Transformer architecture, which allows the model to consider context when predicting the next state. A key feature of Wav2Vec2 is that it is trained unsupervised, meaning it does not require labeled data for training. Instead, it uses a method called contrastive learning to learn audio representations. As a front-end component for a scientific paper, Wav2Vec2 can automatically transcribe audio to text, which can help analyze audio data or create text versions of audio recordings.

2.5. Encoder

The encoder comprises six convolutional units. Except for the initial unit, each subsequent unit comprises two convolution units. The initial unit, however, comprises a single convolution unit in conjunction with another unit. The convolution unit implements the following transformation:

$$\mathrm{ConvUnit(x)} = \mathrm{Conv(SELU(BatchNorm}(x))) \tag{17}$$

Each unit's input is added to the output following the second convolutional unit in that block and downsampled using a convolutional layer if necessary. Subsequently, MaxPooling is applied following this skip connection.

2.6. KAN-GAL

In our work, we were also inspired by AASIST, which is based on the premise that graphs are fully connected because it is impossible to determine the degree of importance of each node to a given task in advance. In contrast to RawGat [32], the activation functions were not employed due to the novel utilization of KANs.

The initial operation is to apply a dropout with a probability of 0.2. Subsequently, the attention mask is obtained by node-wise multiplication(denoted as "$\times$") of the nodes h, $h \in \mathbb{R}^{N,D}$ and N – the number of nodes, D – node dimensionality, and subsequent passing through the KAN layer. Following this, the hyperbolic tangent is applied. The resulting expression is then matrix multiplied by the attention weights W_{att}, which have been initialized using Xavier initialization. The resulting values are then divided by the temperature T, resulting in an attention map A consisting of the corresponding probabilities, which is obtained using the softmax function:

$$A = \mathrm{softmax}\left(\frac{\tanh(\mathrm{KAN}_1(h \times h))W_{att}}{T}\right). \tag{18}$$

The resulting attention map is projected using KAN, and in parallel, it is multiplied by the matrix and projected. The resulting projections are then added together and normalized by batch:

$$\mathrm{KAN\text{-}GAL}(h) = \mathrm{BatchNorm}(\mathrm{KAN}_2(Ah) + \mathrm{KAN}_3(h)). \tag{19}$$

2.7. KAN-GraphPool

As described in the previous section, the initial operation is to apply a dropout, after which the resulting output passes through the KAN layer and is transformed by the sigmoid function, represented by the symbol $\sigma(\times)$. The resulting value is then multiplied elementwise (denoted as "$\odot$") by the original graph. The dimensionality is subsequently reduced using the rank function, which returns the k most significant nodes in the resulting graph:

$$\text{KAN-GraphPool} = \text{rank}((\sigma(KAN(h)) \odot h), k). \quad (20)$$

2.8. KAN-HS-GAL

The layer accepts three inputs: h_t, which has a node dimensionality of D_t (temporal graph), h_s, which has a node dimensionality of D_s (spatial graph), and S (stack node). Input graphs are projected into another latent space using KAN layers to equalize their dimensions and then merged to form a fully connected heterogeneous graph h_{st} with node dimensionality D_{st}. A dropout with a probability of 0.2 is then applied to the resulting graph:

$$h_{st} = \text{CONCAT}(\text{KAN}_1(h_t), \text{KAN}_2(h_s)). \quad (21)$$

The primary attention map A is derived by multiplying each node in the graph h_{st} by every other node, with the projection through the KAN layer undergoing a hyperbolic tangent transformation:

$$A = \tanh(\text{KAN}_3(h_{st} \times h_{st})). \quad (22)$$

To derive a secondary attention map B, the initial attention map is partitioned into four matrices in accordance with the threshold D_t, defined as the number of nodes in the underlying graph h_s. Subsequently, these segments are multiplied by the weights W_{11}, W_{12} and W_{22}:

$$B = \begin{cases} \sum_{m=1}^{D_t+D_s} A_{ijm} \cdot W_{11m}, & \forall i \leq D_t \text{ and } j \leq D_t \\ \sum_{m=1}^{D_t+D_s} A_{ijm} \cdot W_{22m}, & \forall i \geq D_t \text{ and } j \geq D_t \\ \sum_{m=1}^{D_t+D_s} A_{ijm} \cdot W_{12m}, & \text{otherwise.} \end{cases} \quad (23)$$

The matrix is then divided by the temperature value T and passed through the Softmax function, thereby obtaining a probability map:

$$\hat{B} = \text{softmax}\left(\frac{B}{T}\right). \quad (24)$$

To produce the attention map for the stack node update, the heterogeneous graph h_{st} is taken and multiplied by the Stack Node S node-wise. The resulting graph is then projected through the KAN layer, passed through the tangent, and matrix multiplied by the weights W_m. The value obtained is subsequently divided by the temperature and passed through softmax:

$$A_m = \text{softmax}\left(\frac{\tanh(\text{KAN}_4(h_{st} \odot S))}{T}\right). \quad (25)$$

Combining two projections obtained using KAN layers to update a stack node is necessary. These are the projection of the matrix-multiplied attention map A_m and graph h_{st} and the projection of the stack node:

$$\hat{S} = \text{KAN}_5(A_m h_{st}) + KAN_6(S) \quad (26)$$

The update of h_{st} is achieved by combining two projections derived from KAN layers: the projection of the matrix multiplied secondary attention map $\hat{B}$ and the heterogeneous graph h_{st} and the projection of the graph h_{st} itself. The expression obtained is then subjected to batch normalization:

$$\widehat{h_{st}} = \text{BatchNorm}(\text{KAN}_7(\hat{B}h_{st}) + \text{KAN}_8(h_{st})). \quad (27)$$

The resulting heterogeneous graph is then divided back into two components by multiplication with the mask matrices M_1 and M_2:

$$\widehat{h_t} = \widehat{h_{st}} M_t \quad (28)$$

$$M_t = \begin{pmatrix} I_t \\ 0_s \end{pmatrix}, I_t \in \mathbb{R}^{N \times D_t}, 0_s \in \mathbb{R}^{N \times D_s} \quad (29)$$

$$\widehat{h_s} = \widehat{h_{st} M_s} \quad (30)$$

$$M_s = \begin{pmatrix} 0_t \\ I_s \end{pmatrix}, 0_t \in \mathbb{R}^{N \times D_t}, I_s \in \mathbb{R}^{N \times D_s}. \quad (31)$$

2.9. Models Architectures

2.10. AASIST3

In the closed condition, the front-end is SincConv, whereas, in the open condition, it is Wav2Vec2 XLS-R[30] with additional linear or convolutional layers, which maintains the dimensionality.

The application of Max Pooling, Batch Normalization, and SELU preceded the encoder:

$$\hat{x} = \text{Encoder}(\text{SELU}(\text{BatchNorm}(\text{MaxPool}(x))), \quad (32)$$

where x is an input pre-emphasized audio.

Subsequently, the acquired features were divided into temporal and spatial components, after which positional embedding (PE) was incorporated. In this manner, graphs were formed, which were subsequently passed through a KAN-GAL and a KAN-GraphPool:

$$h_t = \text{KAN-GraphPool}(\text{KAN-GAL}(\max_t(\text{abs}(\hat{x}) + \text{PE}_t))) \quad (33)$$

$$h_s = \text{KAN-GraphPool}(\text{KAN-GAL}(\max_s(\text{abs}(\hat{x}) + \text{PE}_s))). \quad (34)$$

The resulting graphs and the previously initialized stack node were passed in parallel through four branches. The initial step is to apply KAN-HS-GAL in each branch:

$$\begin{pmatrix} \widehat{h_{t2}} \\ \widehat{h_{s2}} \\ \widehat{S_2} \end{pmatrix} = \text{KAN-HS-GAL} \begin{pmatrix} \widehat{h_{t1}} \\ \widehat{h_{s1}} \\ \widehat{S_1} \end{pmatrix}. \quad (35)$$

The graphs are then passed through KAN-GraphPool, and another KAN-HS-GAL is applied similarly:

$$\begin{pmatrix} \widehat{h_{t3}} \\ \widehat{h_{s3}} \\ \widehat{S_3} \end{pmatrix} = \text{KAN-HS-GAL} \begin{pmatrix} \text{KAN-GraphPool}(\widehat{h_{t2}}) \\ \text{KAN-GraphPool}(\widehat{h_{s2}}) \\ \widehat{S_2} \end{pmatrix}. \quad (36)$$

To produce the final predictions, all previously obtained graphs and Stack Nodes are stacked:

$$H_t = \widehat{h_{t1}} + \widehat{h_{t2}} + \widehat{h_{t3}} \quad (37)$$

$$H_s = \widehat{h_{s1}} + \widehat{h_{s2}} + \widehat{h_{s3}} \quad (38)$$

$$S_f = \widehat{S_1} + \widehat{S_2} + \widehat{S_3}. \quad (39)$$

A dropout with a probability of 0.2 is applied to all obtained graphs and the Stack Node after four branches. Subsequently,

for temporal and spatial graphs, the node-wise maximum $H^{\max}$ and mean H^{mean} are identified, as well as the maximum Stack node S_f^{max}. The resulting values pass through the dropout with a probability of 0.5 and are then concatenated into the final hidden layer L:

$$L = \text{CONCAT}(H_t^{\max}, H_t^{\text{mean}}, H_s^{\max}, H_t^{\text{mean}}, S_f^{\max}) \quad (40)$$

After L, a KAN layer returns logits for each class.

2.11. Wav2Vec2-Conv-AASIST-KAN

In addition to the proposed AASIST3, we utilized the pre-trained Wav2Vec2 encoder for the feature and proceeded with 1D convolutions to provide the AASIST model with a KAN classification layer. It was motivated by inductive biases from a pre-trained speech encoder on a large-scale dataset in self-supervised (SSL) training, which is preferable for the open-set condition.

3. Experiments and Results

3.1. Description of final approaches

For the Closed Condition, we considered our described AASIST3 model. The model was constrained to accept only four seconds of audio as input, which proved insufficient for the models to achieve a deeper comprehension of the audio as a whole. To address this limitation, the audio was fed into the model in four-second parts sequentially with a two-second overlap between them. We applied pre-emphasis for all audios with no augmentations.

The optimal models were identified upon testing the models on a closed test subset: one with two branches and one with four branches. Additionally, based on the hypothesis that SWL can enhance the results, the model incorporating SWL was utilized. The predictions of these models were averaged.

Table 1: *Final evaluation results of submitted prediction in closed and open condition CM track.*

condition	model	t-DCF	EER
closed	AASIST3	0.5357	22.67
open	$\tilde{f}$	0.1414	4.89

As illustrated in the table 2, many of our modifications produced notably superior outcomes compared to AASIST, even during the validation phase. However, using various techniques to enhance the quality of anti-spoofing models proved ineffective, with all approaches ultimately resulting in a decline in the observed results.

For the Open Condition, to provide the final prediction or probability that given audio x is bonafide, we averaged the predictions of two of our models trained differently to increase generalization ability.

$$\tilde{f} = \frac{f_1^{'}(x) + f_2^{'}(x)}{2}, \quad (41)$$

where

$$f_1^{'}(x) = \sum_i^m f_1(x_i) \quad (42)$$

$$f_2^{'}(x) = \sum_j^l f_2(x_j), \quad (43)$$

and $\{x_i\}_{i=1}^m$ and $\{x_j\}_{j=1}^l$ are some parts of the original audio x, specifically, sequential parts with intersection (for example, 0-4s and 3-7s audio intervals) as in the submission for Closed Condition. The f_1 is AASIST3 but with a pre-trained Wav2Vec2 feature encoder. The f_2 is our second model Wav2Vec2+Conv+AASIST+KAN.

f_1 was trained similarly as in closed condition only on the provided training set, whereas f_2 was trained on the union of the given training set plus additional bonafide audios from Mozilla CommonVoice, train part if VoxCeleb2. For f_2 training, a combination of weighted Cross-Entropy, Focal [33], and LibAUCM [34, 35] losses with Adam optimizer. Augmentation methods such as RIR, environmental and Gaussian noises, VAD, and pitch shifting were randomly applied. Pre-emphasizing was used before augmentations.

3.2. Experiments with different frontends

Given the results presented by [19], it was hypothesized that combining multiple representations might improve the result. Combining the raw waveform with the CQT and Mel-spectrograms was attempted, but no improvement was seen. In light of the findings in [36], we investigated using the f0 subband independently and in conjunction with SincConv. In addition, given the evidence presented in the study referenced in [37], which indicated that Leaf outperformed SincConv, we compared its performance with our model's. However, none of the modifications resulted in a performance improvement. For open conditions, the best result was shown by Wav2Vec2 [30] pre-trained on XLSR-300, as front-end based on a transformer and convolutional neural networks, which allows suitable encoding of both temporal and spatial information in audio. Also, experiments with XEUS [38] did not provide better results compared to other front-ends.

3.3. Experiments with augmentations

This study employed a series of augmentations, including pitch shift, speed change, Gaussian and environmental noise, different room impulse responses, harmonic and percussive components, stretching, voice activity detection (VAD) application, and pre-emphasis. A random portion of the audio sample was extracted or padded during training. In some experiments, the pre-emphasis was used as an augmentation, in others it was applied to each audio. Additionally, we employed Attention Augmented Convolutional Networks[39] and Rawboost[40]. The findings indicate that pre-emphasis on each audio track represents an effective approach.

3.4. Experiments with different base KAN functions

A comparison of AReLU[41], PReLU, SELU, and RReLU showed that PReLU gave the most robust results.

3.5. Experiments with different KAN-based encoders

As one of the key ideas of our model is the use of KAN, we chose to explore the potential of KAN-based en-

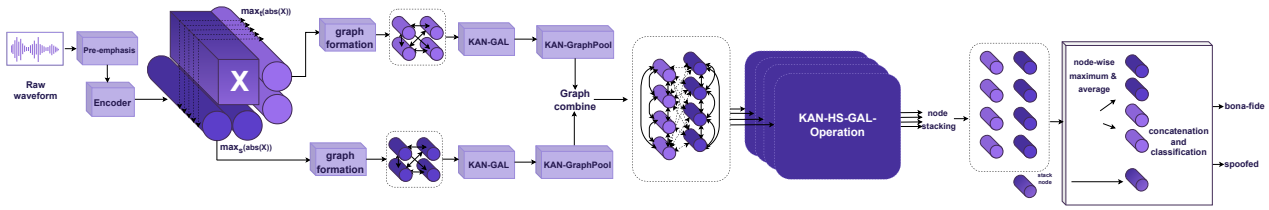

Figure 1: Architecture of the closed condition model.

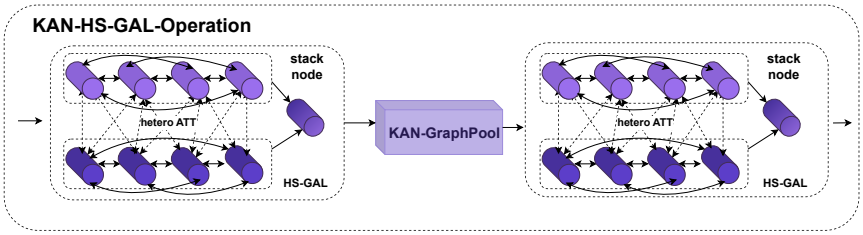

Figure 2: The KAN-HS-GAL Operation.

coders, including KAN+tokenization[42], ReluConvKAN and WavKANConv[43], both in stand-alone form and combined with a sharpness-aware minimisation[20] mechanism. When evaluated using a validation set with SAM, WavKanConv showed favorable results, table 2

Our experiments with different KAN encoders found that the model using the classic RawNet2-based encoder performed best.

3.6. Experiments with different KANs

A comparison was conducted between parametric B-spline functions and other types of polynomials, including Bessel polynomials, Chebyshev polynomials of the second kind, Gaussian radial basis functions, Fibonacci polynomials, radial-basis functions, and Jacobi polynomials. The results showed that the approximation with a B-spline of order 4 gave the most accurate results.

3.7. Experiments with AASIST modification

In order to test the effectiveness of the proposed methodology, several modifications were applied. These included the insertion of the third additional branch with channel-wise maximum, which was used to allow the extraction of more complex features. Furthermore, GraphPools were replaced by GALs to avoid removing a significant amount of information. The minimum was used instead of the maximum to form branches, and four branches with HS-GAL and SE[44] in the encoder were used. Six branches were also used with HS-GAL, and positional encoding was introduced instead of positional embedding[**?**]. Finally, a comparison was made between the results obtained with two branches and those obtained with the proposed methodology. The results show that the best result is obtained using four branches with HS-GAL.

3.8. Experiments with scaling of HS-GAL branches

Additional HS-gals with different temperature values were applied to two branches. As shown in the table 2, the obtained results did not exhibit a considerably enhanced degree of improvement.

The results allow us to conclude that scaling the model in width is more optimal than scaling it in depth.

3.9. Experiments with different encoders

Given that the original AASIST employs a RawNet2-based encoder, we postulated that a RawNet3-based encoder[45] would enhance the model. Concurrently, we anticipate that S2pecNet[19], as the authors have demonstrated, will improve the result through this integration of sound representations. Additionally, we explored the potential of WaveNet[46] as a front-end, but unfortunately, none of the experiments yielded a significant result.

3.9.1. Experiments with Res2Net-based encoders

Following AASIST2[17], we attempted to utilize Res2Net in various configurations with disparate learning rates. Concurrently, we evaluated SR LA RES2net[36] as a more sophisticated analog. The results of our investigation suggest that the application of Res2Net with the proposed AASIST configuration is not a viable approach.

3.9.2. Experiments with ResNet-based encoders

Utilizing alternative encoders and modifications to the Res2Net encoder yielded no perceptible improvement in results. Therefore, an investigation was conducted into alternative changes to the ResNet encoder. These included the utilization of the f0 subband instead of the SincConv, the use of two ResNet encoders for different segments of audio, with and without RawBoost, the integration of ELA [47], the substitution of BatchNorm with LayerNorm, the utilization of prelu as the activation function, and the integration of SE. The experimental results demonstrate that modifying the encoder does not improve outcomes.

3.10. Experiments with different loss functions

Furthermore, in line with AASIST2[17], AM-Softmax and its predecessor, ArcFace[48], were also tested. Based upon the results of the study [49], focal loss was also tested. Furthermore, we attempted to utilize generalized cross entropy[23], the effec-

tiveness of which has been previously established for AASIST. Additionally, as in the original AASIST, we attempted to utilize weighted cross-entropy. Finally, we selected multitask losses, hypothesizing that the model would be capable of extracting more complex features. However, our findings indicated that regular cross-entropy was indeed efficacious. Consequently, the deployment of loss to address class imbalance in our model can be considered ineffective. However, for better generalization, our second model was trained using a combination of weighted cross-entropy, focal loss, and LibAUCM [34, 35] loss, which is implied for the x-risk minimization.

3.11. Experiments with different optimizers

The following optimizers were selected for our experiments: AdamW, Lion, NAdam, RAdam, and Adam. As illustrated in the table 2, the outcomes with AdamW and Lion exhibited a notable decline in performance. In addition, the results with RAdam were found to be unsatisfactory. The results on the development subset are presented in the table 2.

The findings indicate that Adam is the optimal optimizer for our model.

3.12. Experiments with different learning methods

To enhance the generalization capacity, we employed a variety of techniques, including SAM [20], ASAM [21], and SWL [22]. These were initially utilized in the original paper about AASIST and led to a notable enhancement in the quality of the models. Furthermore, a cosine annealing scheduler and a weighted random sampler are employed. As illustrated in the table 2, these strategies have also been demonstrated to be ineffective.

It was found that none of the proposed learning methods improved the result.

4. Conclusion

The rapid development of various deep learning algorithms has created new opportunities for generating synthetic audio using TTS and VC systems. This progress, however, has introduced a corresponding vulnerability in ASV systems, necessitating the development of a CM system to detect synthetic voices. In this paper, we proposed a novel architecture, AASIST3, which enhances the original AASIST framework by incorporating Kolmogorov-Arnold networks, additional layers, and pre-emphasis. Furthermore, we introduced modifications using B-spline features as training features inspired by previous enhancements in synthetic speech detection models. In addition, we utilized additional data, scores fusion, and a self-supervised pre-trained model as an encoder to achieve the best results. Our findings indicated that these modifications significantly improve model performance, achieving a more than twofold improvement over AASIST. The model demonstrated minDCF results of 0.5357 under closed conditions and 0.1414 under open conditions, affirming the effectiveness of our configuration.

Table 2: *Results of experiments with AASIST3 on dev subset.*

Referenced	AASIST3 modification	t-DCF
[16]	**Classic AASIST**	**0.5671**
sec. 3.1	**AASIST3**	**0.2657**
sec. 3.2	Leaf instead of SincConv	0.52
sec. 3.2	F0 subband + SincConv	0.4406
sec. 3.2	F0 subband instead of SincConv	0.4225
sec. 3.2	SincConv + CQT	0.4083
sec. 3.3	AttentionAugmentedConv2d in encoder	0.4762
sec. 3.3	Augmentations without pre-emphasis	0.4495
sec. 3.3	All augmentations	0.3624
sec. 3.4	AReLU instead of PReLU in KAN	0.3371
sec. 3.4	SELU instead of PReLU in KAN	0.3295
sec. 3.4	RReLU instead of PReLU in KAN	0.3045
sec. 3.5	ReLUConvKAN instead of Conv	0.699
sec. 3.5	UKAN in encoder	0.3062
sec. 3.5	WavKANConv in encoder	0.3047
sec. 3.5	WavKANConv in encoder + SAM	0.2801
sec. 3.6	Bessel polynomials in KAN	0.6805
sec. 3.6	Jacobi Polynomials in KAN	0.5666
sec. 3.6	Legendre Polynomials in KAN	0.4665
sec. 3.6	Gegenbauer polynomials in KAN	0.4051
sec. 3.6	RBF in KAN	0.3994
sec. 3.6	2nd Chebyshev polynomials	0.3392
sec. 3.6	Fibonacci polynomials in KAN	0.492
sec. 3.7	Utilising GATs instead of GraphPool	0.4375
sec. 3.7	Gaussian RBF	0.3536
sec. 3.7	4 branches of 2 HS-GALs + SE	0.2905
sec. 3.7	3rd branch with channel-wise maximum	0.4992
sec. 3.8	2 branches of 3 HS-GALs with temp=100	0.3077
sec. 3.8	2 branches of 3 HS-GALs with temp=150	0.2661
sec. 3.8	2 branches of 3 HS-GALs with temp=200	0.2862
sec. 3.9	S^2pecNet with 40 batch size	0.4291
sec. 3.9	S^2pecNet with 28 batch size	0.4225
sec. 3.9	S^2pecNet with 20 batch size	0.4185
sec. 3.9	RawNet3 instead of RawNet2	0.4901
sec. 3.9.1	Res2Net encoder with lr=1e-6	0.9066
sec. 3.9.1	SR LA Res2Net encoder	0.7203
sec. 3.9.1	Res2Net encoder with lr=1e-5	0.6413
sec. 3.9.1	SR LA Res2Net + f0 subband	0.5463
sec. 3.9.1	Res2Net encoder + PRELU with lr=1e-4	0.485
sec. 3.9.2	LayerNorm instead of BatchNorm	0.3591
sec. 3.9.2	ResNet + effictive local attention	0.3542
sec. 3.9.2	ResNet with PReLU	0.3216
sec. 3.9.2	ResNet + SE	0.2902
sec. 3.10	Generalized Cross Entropy Loss	0.8438
sec. 3.10	ArcFace Loss	0.4389
sec. 3.10	Multitask Loss	0.3933
sec. 3.10	Focal Loss	0.3489
sec. 3.10	AM Softmax	0.3363
sec. 3.11	Lion instead of Adam	0.3702
sec. 3.11	AdamW instead of Adam	0.3200
sec. 3.11	NAdam instead of Adam	0.2889
sec. 3.11	RAdam instead of Adam	0.3006
sec. 3.12	SAM rho=0.5	0.5012
sec. 3.12	SAM rho=0.05	0.3727
sec. 3.12	ASAM	0.3532
sec. 3.12	SWL	0.2694
sec. 3.12	Cosine annealing scheduler	0.2991
sec. 3.12	Weighted random sampler	0.2989

5. References

[1] Nicholas Evans, Tomi Kinnunen, and Junichi Yamagishi, "Spoofing and countermeasures for automatic speaker verification," in *INTERSPEECH 2013, 14th Annual Conference of the International Speech Communication Association, August 25-29, 2013, Lyon, France*, ISCA, Ed., Lyon, 2013.

[2] Zhizheng Wu et al., "Asvspoof 2015: the first automatic speaker verification spoofing and countermeasures challenge," in *INTERSPEECH 2015*, ISCA, Ed., Dresden, 2015.

[3] Tomi Kinnunen, Md. Sahidullah, Héctor Delgado, et al., "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection," in *Proc. Interspeech 2017*, 2017, pp. 2–6.

[4] Xin Wang, Junichi Yamagishi, Massimiliano Todisco, et al., "Asvspoof 2019: A large-scale public database of synthesized, converted and replayed speech," 2020.

[5] Xuechen Liu et al., "Asvspoof 2021: Towards spoofed and deepfake speech detection in the wild," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 31, pp. 2507–2522, 2023.

[6] Yongyi Zang, You Zhang, Mojtaba Heydari, and Zhiyao Duan, "Singfake: Singing voice deepfake detection," 2024.

[7] Galina Lavrentyeva, Sergey Novoselov, Andzhukaev Tseren, Marina Volkova, Artem Gorlanov, and Alexandr Kozlov, "STC Antispoofing Systems for the ASVspoof2019 Challenge," in *Proc. Interspeech 2019*, 2019, pp. 1033–1037.

[8] Sunmook Choi, Il-Youp Kwak, and Seungsang Oh, "Overlapped Frequency-Distributed Network: Frequency-Aware Voice Spoofing Countermeasure," in *Proc. Interspeech 2022*, 2022, pp. 3558–3562.

[9] Moustafa Alzantot, Ziqi Wang, and Mani B. Srivastava, "Deep Residual Neural Networks for Audio Spoofing Detection," in *Proc. Interspeech 2019*, 2019, pp. 1078–1082.

[10] Cheng-I Lai, Nanxin Chen, Jesús Villalba, and Najim Dehak, "ASSERT: Anti-Spoofing with Squeeze-Excitation and Residual Networks," in *Proc. Interspeech 2019*, 2019, pp. 1013–1017.

[11] Diego Castan et al., "Speaker-Targeted Synthetic Speech Detection," in *Proc. The Speaker and Language Recognition Workshop (Odyssey 2022)*, 2022, pp. 62–69.

[12] Il-Youp Kwak et al., "Voice spoofing detection through residual network, max feature map, and depthwise separable convolution," *IEEE Access*, vol. PP, pp. 1–1, 01 2023.

[13] Xinhui Chen, You Zhang, Ge Zhu, and Zhiyao Duan, "UR Channel-Robust Synthetic Speech Detection System for ASVspoof 2021," in *Proc. 2021 Edition of the Automatic Speaker Verification and Spoofing Countermeasures Challenge*, 2021, pp. 75–82.

[14] Lei Wu and Ye Jiang, "Attentional fusion tdnn for spoof speech detection," in *2022 5th International Conference on Pattern Recognition and Artificial Intelligence (PRAI)*, 2022, pp. 651–657.

[15] Awais Khan and Khalid Malik, "Spotnet: A spoofing-aware transformer network for effective synthetic speech detection," in *2nd ACM International Workshop on Multimedia AI against Disinformation (MAD'23)*, 06 2023.

[16] Jee weon Jung et al., "Aasist: Audio anti-spoofing using integrated spectro-temporal graph attention networks," 2021.

[17] Yuxiang Zhang, Jingze Lu, Zengqiang Shang, Wenchao Wang, and Pengyuan Zhang, "Improving short utterance anti-spoofing with aasist2," 2024.

[18] Hemlata Tak et al., "Automatic speaker verification spoofing and deepfake detection using wav2vec 2.0 and data augmentation," 2022.

[19] Penghui Wen et al., "Robust Audio Anti-Spoofing with Fusion-Reconstruction Learning on Multi-Order Spectrograms," in *Proc. INTERSPEECH 2023*, 2023, pp. 271–275.

[20] Pierre Foret, Ariel Kleiner, Hossein Mobahi, and Behnam Neyshabur, "Sharpness-aware minimization for efficiently improving generalization," 2021.

[21] Jungmin Kwon, Jeongseop Kim, Hyunseo Park, and In Kwon Choi, "Asam: Adaptive sharpness-aware minimization for scale-invariant learning of deep neural networks," 2021.

[22] Zhiyong Wang, Ruibo Fu, Zhengqi Wen, Yuankun Xie, Yukun Liu, Xiaopeng Wang, Xuefei Liu, Yongwei Li, Jianhua Tao, Yi Lu, Xin Qi, and Shuchen Shi, "Generalized fake audio detection via deep stable learning," 2024.

[23] Hye jin Shim, Md Sahidullah, Jee weon Jung, Shinji Watanabe, and Tomi Kinnunen, "Beyond silence: Bias analysis through loss and asymmetric approach in audio anti-spoofing," 2024.

[24] Siwen Ding, You Zhang, and Zhiyao Duan, "Samo: Speaker attractor multi-center one-class learning for voice anti-spoofing," 2022.

[25] Ziming Liu, Yixuan Wang, Sachin Vaidya, Fabian Ruehle, James Halverson, Marin Soljačić, Thomas Y. Hou, and Max Tegmark, "Kan: Kolmogorov-arnold networks," 2024.

[26] A. N. Kolmogorov, "On the Representation of continuous functions of several variables as superpositions of continuous functions of a smaller number of variables.," *Dokl. Akad. Nauk*, vol. 108, no. 2, 1956.

[27] Jürgen Braun and Michael Griebel, "On a constructive proof of Kolmogorov's superposition theorem," *Constructive approximation*, vol. 30, pp. 653–675, 2009, Publisher: Springer.

[28] A. N. Kolmogorov, "On the representation of continuous functions of many variables by superposition of continuous functions of one variable and addition," in *Doklady Akademii Nauk*. 1957, vol. 114, pp. 953–956, Russian Academy of Sciences.

[29] Mirco Ravanelli and Yoshua Bengio, "Speaker recognition from raw waveform with sincnet," 2019.

[30] Alexei Baevski, Henry Zhou, Abdelrahman Mohamed, and Michael Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," 2020.

[31] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[32] Hemlata Tak, Jee weon Jung, Jose Patino, Madhu Kamble, Massimiliano Todisco, and Nicholas Evans, "End-to-end spectro-temporal graph attention networks for speaker verification anti-spoofing and speech deepfake detection," 2021.

[33] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár, "Focal loss for dense object detection," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2980–2988.

[34] Tianbao Yang, "Algorithmic foundation of deep x-risk optimization," *arXiv preprint arXiv:2206.00439*, 2022.

[35] Zhuoning Yuan, Dixian Zhu, Zi-Hao Qiu, Gang Li, Xuanhui Wang, and Tianbao Yang, "Libauc: A deep learning library for x-risk optimization," in *29th SIGKDD Conference on Knowledge Discovery and Data Mining*, 2023.

[36] Cunhang Fan, Jun Xue, Jianhua Tao, Jiangyan Yi, Chenglong Wang, Chengshi Zheng, and Zhao Lv, "Spatial reconstructed local attention res2net with f0 subband for fake speech detection," 2023.

[37] Neil Zeghidour, Olivier Teboul, Félix de Chaumont Quitry, and Marco Tagliasacchi, "Leaf: A learnable frontend for audio classification," 2021.

[38] William Chen et al., "Towards robust speech representation learning for thousands of languages," *arXiv preprint arXiv:2407.00837*, 2024.

[39] Irwan Bello, Barret Zoph, Ashish Vaswani, Jonathon Shlens, and Quoc V. Le, "Attention augmented convolutional networks," 2019.

[40] Hemlata Tak, Madhu Kamble, Jose Patino, Massimiliano Todisco, and Nicholas Evans, "Rawboost: A raw data boosting and augmentation method applied to automatic speaker verification anti-spoofing," 2021.

[41] Dengsheng Chen, Jun Li, and Kai Xu, "Arelu: Attention-based rectified linear unit," 2020.

[42] Chenxin Li et al., "U-kan makes strong backbone for medical image segmentation and generation," 2024.

[43] Ivan Drokin, "Kolmogorov-arnold convolutions: Design principles and empirical studies," 2024.

[44] Jie Hu, Li Shen, Samuel Albanie, Gang Sun, and Enhua Wu, "Squeeze-and-excitation networks," 2017.

[45] Jee weon Jung, You Jin Kim, Hee-Soo Heo, Bong-Jin Lee, Youngki Kwon, and Joon Son Chung, "Pushing the limits of raw waveform speaker recognition," 2022.

[46] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu, "Wavenet: A generative model for raw audio," 2016.

[47] Wei Xu and Yi Wan, "Ela: Efficient local attention for deep convolutional neural networks," 2024.

[48] Jiankang Deng, Jia Guo, Jing Yang, Niannan Xue, Irene Kotsia, and Stefanos Zafeiriou, "Arcface: Additive angular margin loss for deep face recognition," 2018.

[49] Qiaowei Ma, Jinghui Zhong, Yitao Yang, Weiheng Liu, Ying Gao, and Wing W.Y. Ng, "Convnext based neural network for audio anti-spoofing," 2022.