AI Generated Speech Detection Using CNN

Meriç Demirörs
TOBB University of Economics and Technology
Ankara, Türkiye
mdemirors@etu.edu.tr

Toygar Akgün
TOBB University of Economics and Technology
Ankara, Türkiye
takgun@etu.edu.tr

Ahmet Murat Özbayoğlu
TOBB University of Economics and Technology
Ankara, Türkiye
mozbayoglu@etu.edu.tr

***Abstract*—The recent rapid developments in generative-AI research has made it exceedingly hard to distinguish artificially generated audio-visual content from real ones. As a result, reliably detecting synthetic content has become an important problem to solve. In this study, multiple CNN, FC and SVM models are trained to detect synthetic audio signals obtained by using generative-AI models. The test results show that the best accuracy scores for the CNN, FC and SVM models are 99.06, 99.15 and 98.68%, respectively. These results point out that the synthetic audio signals can be discriminated from the real ones by the trained models. Therefore, the proposed solution can be used in real-life practical applications to tackle this problem. Our analyses show that CNN models are the most suitable compared to other techniques, as FC and SVM models can also detect synthetic audios but have different inherent disadvantages.**

Keywords—Bispectrum, CNN, FC, SVM

I. Introduction

The developments in the last decade truly transformed the field of audio-visual content generation. As of 2024, deep text-to-image models based on the Transformer [1] architecture (DALL-E [2]) or the denoising diffusion probabilistic models (Stable Diffusion [3]) can generate staggeringly realistic visuals. Supported by their success on visual data generation, generative-AI models are getting increasingly popular in the audio generation field as well, for example, Meta's Voicebox [4] is a generative AI model for speech that generalize across multiple tasks. The importance of distinguishing synthetic audio from real ones has been highlighted by realistic synthetic audio produced by ever-developing generative-AI models. Additionally, with increasing debates about AI security and licensing rights, more and more models will be researched in this field.

This work presents analyses conducted to determine whether there are discriminative characteristic features of real audio signals, obtained by looking at the interactions between the various frequency components in the bispectrum of AI-generated audio. Our research demonstrates that audio signals can be classified by obtaining "signature image" features from bispectrums and training CNN (Convolutional Neural Network), FC (Fully Connected), and SVM (Support Vector Machines) models on them. Key point of the research is to compare the performances of CNN, FC and SVM models on audio bispetrum features.

The rest of this paper is organized as follows: Section 1 provides summary of previous studies in this field and highlights the distinctive aspects of this research compared to them; Section 2 explains the properties of audio bispectrums, how to process audios to obtain so-called "signature image", the types of models that are experimented with and the used dataset with its properties; Section 3 discusses the obtained results; Section 4 proposes potential avenues for further exploration in this area; lastly, finishing with the Section 5.

A. Background and Existing Works

There have been many studies for the detection of synthetic audio. Previous research have shown and discussed the effectiveness of various features [5, 6, 7], different type of detection techniques like statistics [8], clustering algorithms [9], machine learning based models [5, 10, 11, 12, 13, 14] and different FC and/or CNN architectures [13, 14, 15, 16, 17, 18]. Raw audio features [5, 8, 10, 11, 12, 13, 14, 15, 17, 18] and other feature extractor models [9, 16] are used to get features. These can be grouped by based on their model-feature selection as machine learning and deep learning models with short- and long-term features from signals. Mostly used models are SVMs and CNN models, which yield the best performance metrics among machine learning and deep learning techniques, paired with bispectrum or MFCC (Mel Frequency Cepstral Coefficients) features. If successes of these model-feature pairs are compared, it is seen that SVMs and CNN models are head-to-head in some cases, but CNN models are ahead in the field (except few cases which can be caused by poor CNN designs or features that are more fitting for SVMs).

After comparisons between SVMs, FC and CNN models, it is seen that SVMs have disadvantages of data preprocessing, manual feature extraction, unscalable feature vectors and curse of dimensionality on big inputs. FC models have disadvantages of data preprocessing (although not as much as SVM), manual feature extraction (if not passing the whole flattened data), unscalable input vectors, massive input layers, overfitting and longer training times compared to SVMs for large models. CNN models have disadvantages of overfitting, which can be solved by regularization techniques, and longer training times at large models compared to SVMs but still shorter compared to FC models. CNN models' advantages over other methods can be listed as: more automated data preprocessing and feature extraction, more scalable type of inputs and being more robust.

II. Methods

For this research, features conveying information related to the high-frequency region, dynamic characteristics, and detailed spectral information were chosen. After both MFCC and bispectrum features were examined, bispectrum was preferred because of the visibly distinctive features they contain. Researches [8, 10, 11] used the same audio bispectrums, but statistical results were extracted from the features, and the entire audio was summarized as a feature vector to feed it to SVM models.

Unlike the previous research works, to obtain individual features, features from audio bispectrums are extracted, and various averaging and normalization methods were applied. Then the features are fed as whole to SVMs, FC and CNN models since these images has obvious visibly distinctive features in them. SVMs and FC models are also fed with preprocessed features for the sake of experimenting.

A. Bispectrum

The bispectrum of the signal represents higher-order correlation in the Fourier domain. These higher order correlations can be detected using bispectral analysis. Bispectrum of the signal is calculated as shown in the (1) to capture third-order correlations, where Y(ω) is the Fourier transform of the signal and Y*(ω) is the complex conjugate. This way it reveals the sorts of "un-natural" higher-order correlations introduced by a nonlinearity. That is, correlations between the triples [ω1,ω1,ω1+ω1], [ω2,ω2,ω2+ω2], [ω1,ω2,ω1+ω2], [ω1,-ω2,ω1-ω2]. The bispectrum is a complex valued quantity. So for the purpose of simplicity and interpretation for this research's problem, it is suitable to represent or use the complex bispectrum with respect to its magnitude and its phase which are computed as shown in the (2) and (3), respectively.

Also, it is helpful to work with the normalized bispectrum (the bicoherence). A common form of normalizing is to divide the signal into multiple segments. For example, the signal can be divided into K overlapping or nonoverlapping segments. The bicoherence is then estimated from the average of each segment's bicoherence spectrum as shown in the (4). All equations are shown in the Fig. 1. Detailed information on the bispectrum can be found in [8, 10, 11] and the references therein.

$$B(\omega_1,\omega_2)=Y(\omega_1)Y(\omega_2)Y^*(\omega_1+\omega_2) \quad (1)$$

$$|B(\omega_1,\omega_2)|=|Y(\omega_1)|\cdot|Y(\omega_2)|\cdot|Y(\omega_1+\omega_2)| \quad (2)$$

$$\angle B(\omega_1,\omega_2)=\angle Y(\omega_1)+\angle Y(\omega_2)-\angle Y(\omega_1+\omega_2) \quad (3)$$

$$\hat{B}_c(\omega_1,\omega_2)=\frac{\frac{1}{K}\sum_k Y_k(\omega_1)Y_k(\omega_2)Y_k^*(\omega_1+\omega_2)}{\sqrt{\frac{1}{K}\sum_k |Y_k(\omega_1)Y_k(\omega_2)|^2\frac{1}{K}\sum_k |Y_k^*(\omega_1+\omega_2)|^2}} \quad (4)$$

Fig. 1. Bispectrum, magnitude, phase and bicoherence equations.

B. Audio Processing

To derive raw features from audio bispectrums, initially audio data is converted into image data. Five separate image features are obtained from an audio, which we call "absolute", "angle", "cum3", "real" and "imag". To obtain these five features, audio is divided into K segments, each containing 400 samples (because of the typical human voice stationarity window size) with an overlap of 200 samples, and a bispectrum is calculated for each segment. Apperanaces of each feature are shown in the Fig. 2.

After all processing steps, sum of each segments' 3rd order cumulant is obtained, which is later used to obtain "cum3" feature. Also a matrix with shape [K (stacked layers along the stacking axis), H (height of bispectrum mag/phase), W (width of bispectrum mag/phase), 1 (a+ib complex number)] is created which will be later used to obtain "signature image". "cum3" feature is computed from average of the summed 3rd order cumulants and other four features are extracted from "signature image". After all the processing, all features are normalized to [0-1] range. Process is as follows:

FEATURE CALCULATION ALGORITHM

```
cum3_sum ← zero matrix to hold summation
RC_layers ← empty matrix to hold complex values
for ith segment of audio:
    bs ← bispectrum of ith segment with hamming window
    mag ← magnitude of the bs
    phase ← phase of the bs
    cum3 ← 3rd order cumulant of the bs
    cum3_sum ← cum3_sum + cum3
    R ← mag * cos(phase)
    C ← mag * sin(phase)
    RC_layers[i] ← R + Cj
cum3 ← cum3_sum / K
signature_image ← empty matrix to hold complex values
sum_RC ← sum of RC_layers along the stacking axis
for each index row i and column j at signature_image:
    L ← RC_layers[:,i,j,:] vector of K complex value along stacking axis
    top ← sum_RC[i,j]
    bottom ← square_root((L.transpose · L.conjugate).real)
    signature_image[i,j] ← top/(bottom + eps)
absolute ← signature_image.absolute
angle ← signature_image.angle
real ← signature_image.real
imag ← signature_image.imag
```

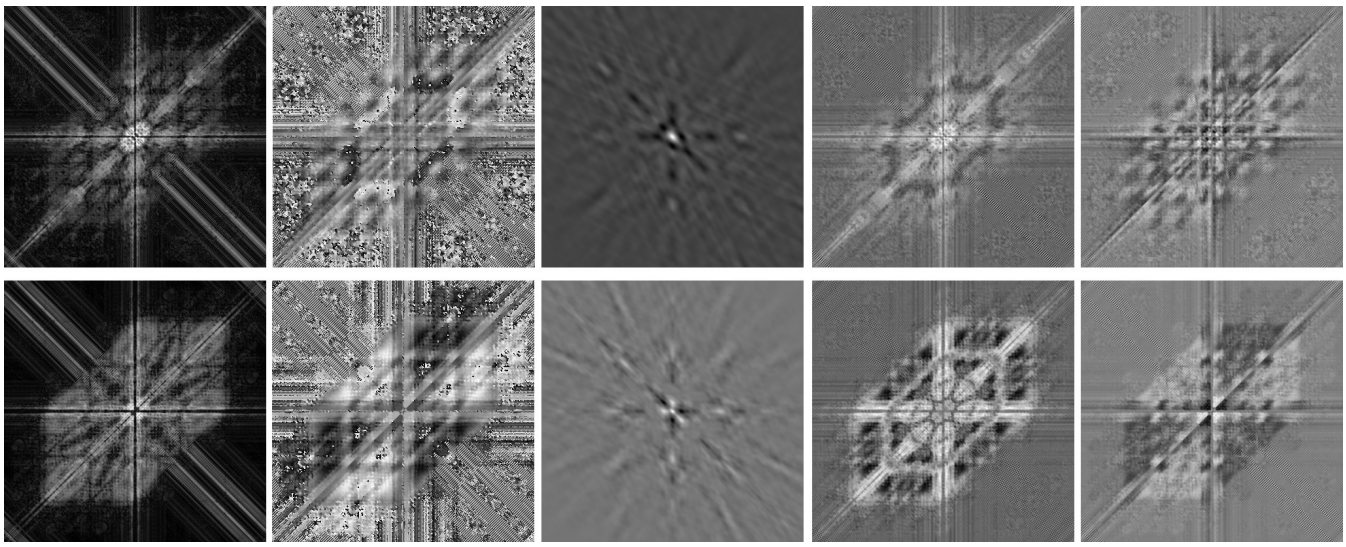

Fig. 2. Features of the same speaker's fake (top) and real (bottom) audios (left to right: absolute, angle, cum3, real, imag).

C. Models

1) FC Models

In order to experiment more than one training, FC models were trained on different processed input data:

- Raw training with whole data flattened
- Dimensionality reduction using PCA (Principal Component Analysis) technique with 64 and 256 output features
- Features which are extracted with ResNet50 model

For each training style there were a total of six data types to train with: five individual features (absolute, angle, cum3, real, imag) and one multi form feature which is obtained by stacking each individual feature back to back. A total of 24 trainings were performed with FC models using all type of input data listed above (All trainings were done with FC models that have suitable input sizes for that data type.):

- Six training with raw data
- Twelve training with PCA: six in 64 dimensions and six in 256 dimensions
- Six more with extracted feature data

A simple base FC architecture, called basic_FC, which's architecture is shown in the Fig. 3 is used to derive other FC models (ReLU activation function is used after first three processing blocks that consists of one fully connected layer followed by a batch normalization layer for the first two layers. Sigmoid activation function is used for the last layer). Architectures of FC models that use dimensionality reduction or feature extraction were similar to basic_FC model's architecture with each having small changes. Their input layers were modified to handle input vectors with shape of dimensionality reduction algorithm's output or extracted features shape depending on the preprocess they use. Also some additional hidden layers were added to even out the competition since their parameter counts were way lower than the basic_FC model's. basic_FC model has parameter counts around hundreds of millions, where dimensionality reduction models are at tens of thousands and feature extraction models are around millions even with the additional hidden layers. basic_FC model's input layer neurons are higher than the other FC models that use dimensionality reduction and feature extraction since input vector was just flattened without no additional process where other FC models' inputs were processed with either dimensionality reduction or feature extraction (even sometimes resizing to adjust for computer capabilities) rather than just flattened.

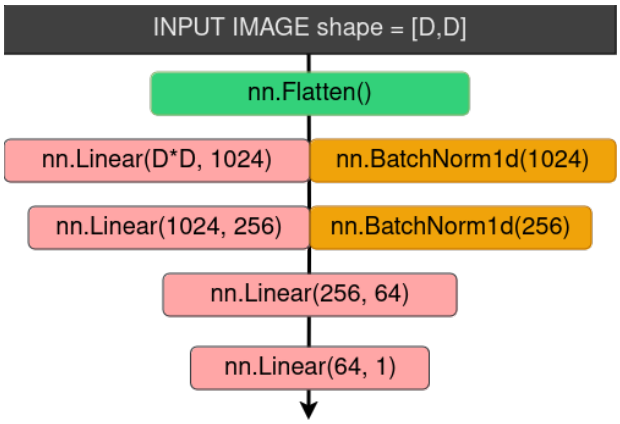

Fig. 3. Architecture of basic_FC model.

2) CNN Models

Three different CNN models were trained, ResNet50, GoogLeNet and custom basic_CNN model whose architecture is illustrated in the Fig. 4 (ReLU activation function is used after each processing block that consists of one convolutional layer followed by a batch normalization layer for the second and third layers and a pooling layer for the second, third and fourth layers, where the ReLU activation function is used before the pooling layers. Sigmoid activation function is used for the last layer). It is obvious that both of the basic_FC and basic_CNN models are not the best ways to design a model and both have different shortcomings such as lack of dropout layers and other experimented and proved techniques such as skip connections (also called shortcut connections), inception modules or squeeze-and-excitation blocks.

In order to explore various configurations of the ResNet50 and GoogLeNet models, three different types of architectures based on each were created: a model was trained from scratch with raw individual feature inputs, another model was trained from scratch with RGB format raw individual feature inputs (the same feature stacked three times in a row), and one last model was trained by freezing the feature extractor input layers at pre-trained weights (input format with the same feature stacked three times). Since CNN models' input format is two or more dimensional, CNN models were trained only on raw features, no dimensionality reduction or feature extraction beforehand applied. Output layers of ResNet50 and GoogLeNet models were modified to perform binary classification.

A total of 35 trainings are performed with seven types of models (basic_CNN + googLeNet (with three types) + ResNet50 (with three types)) with five types of raw individual features (absolute + angle + cum3 + real + imag). Apart from these, three more trainings are performed with three models (basic_CNN_multi, googLeNet_multi and ResNet50_multi) on raw multi form data. That makes a total of 38 training which are shown in the "III. Results and Discussion".

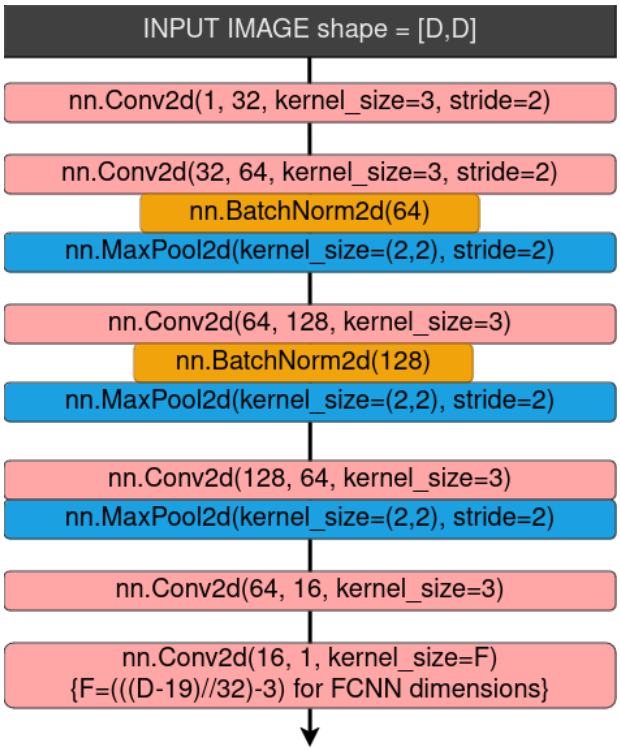

Fig. 4. Architecture of basic_CNN model.

3) SVM Models

In order to experiment more than one training, two different SVM (Linear Support Vector Classification and C-Support Vector Classification from scikit-learn library[19]) models were trained on different processed input data:

- Raw training with whole data flattened
- Dimensionality reduction using PCA and UMAP (Uniform Manifold Approximation and Projection) technique with 64 and 256 output features
- Features which are extracted with ResNet50 model

For each training style there were a total of six data to train with as same in FC trainings. A total of 72 trainings were performed on two SVM models using (All trainings were done with two SVM models that have suitable input sizes for that data type.):

- Twelve training with raw data
- 48 training with PCA and UMAP: 24 training in 64 dimensions and 24 training in 256 dimensions
- Twelve more with extracted feature data

D. Dataset

For dataset, "'In-the-Wild' Dataset" [20] was used, which contains real and synthetic voice recordings of famous names. There are 19963 real and 11816 synthetic voice recordings, the longest of which is 24 seconds, all with a sample rate of 16 kilohertz. The distribution of audio lengths and speaker audios are shown in the Fig. 5 and Fig. 6. In order to ensure a balanced distribution between classes for training, Features from 11816 instances both from real and synthetic voices were extracted. Train, validation and test split rates were 0.8, 0.1, 0.1, respectively, at training.

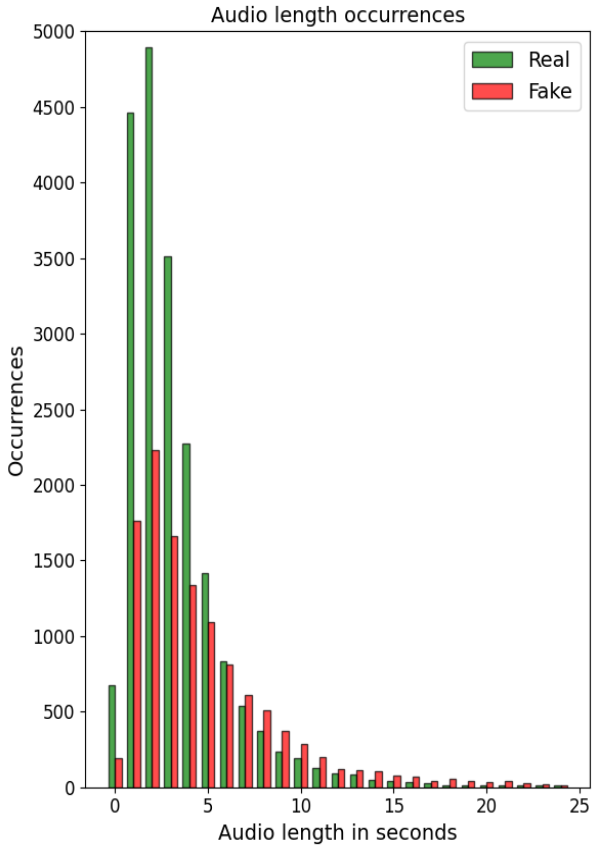

Fig. 5. Number of recording lengths.

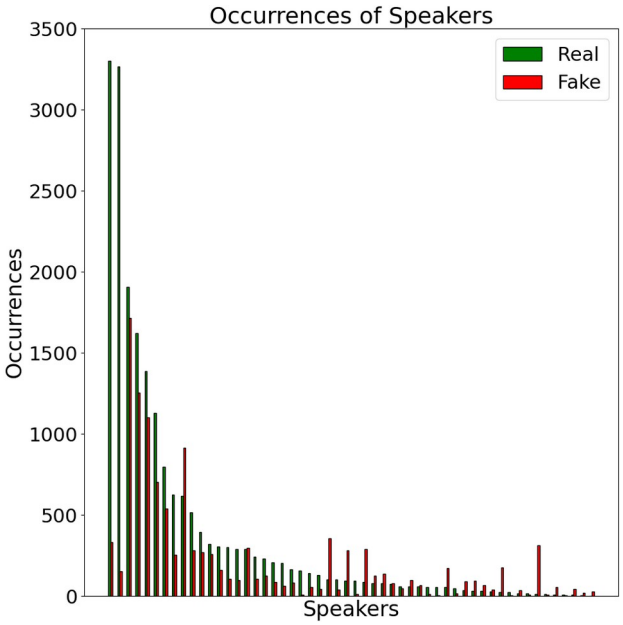

Fig. 6. Number of recordings for each speaker.

III. Results And Discussion

The accuracy scores in percentages of each model on the test set are shown in Tables I through IX. While only Total accuracy is considered when judging a model's success, the importance of Real and Fake accuracy should not be ignored. Models' accuracy distributions are shown in the Fig. 7.

To make an explanation for the understandability of the tables: "dim_red" means dimensionality reduction where "PCA" and "UMAP" names the technique; "feat_ext" means feature extraction; "gray", "RGB" and "pt" indicates the ResNet50 and GoogLeNet versions; suffixes indicates the feature which model trained on and rest it the model type.

A. FC Results

TABLE I. FC Accuracies With Raw Data

Raw FC Trainings	Total	Real	Fake
basic_FC_absolutes	98.096	98.393	97.797
basic_FC_angles	96.099	95.734	96.419
basic_FC_cum3s	97.842	97.399	98.293
basic_FC_imags	97.631	98.497	96.741
basic_FC_reals	98.223	98.377	98.072
basic_FC_multi	**99.154**	**99.579**	**98.723**

TABLE II. FC Accuracies With PCA 64 Dimensions

Dimensionality Reduction FC Trainings (64)	Total	Real	Fake
dim_red_FC_PCA_absolutes	96.192	96.227	96.157
dim_red_FC_PCA_angles	92.131	92.334	91.944
dim_red_FC_PCA_cum3s	97.631	97.339	**97.914**
dim_red_FC_PCA_imags	97.842	98.281	97.416
dim_red_FC_PCA_reals	96.996	97.312	96.654
dim_red_FC_PCA_multi	**98.477**	**99.077**	97.865

TABLE III. FC Accuracies With PCA 256 Dimensions

Dimensionality Reduction FC Trainings (256)	Total	Real	Fake
dim_red_FC_PCA_absolutes	96.912	97.155	96.677
dim_red_FC_PCA_angles	94.416	93.134	95.825
dim_red_FC_PCA_cum3s	97.927	96.841	**99.052**
dim_red_FC_PCA_imags	97.842	97.676	98.003
dim_red_FC_PCA_reals	97.292	97.334	97.252
dim_red_FC_PCA_multi	**98.731**	**98.995**	98.460

TABLE IV. FC ACCURACIES WITH FEATURE EXTRACTION

Feature Extraction FC Trainings	Total	Real	Fake
feat_ext_FC_absolutes	91.920	91.465	92.358
feat_ext_FC_angles	92.555	93.596	91.448
feat_ext_FC_cum3s	91.709	92.857	90.527
feat_ext_FC_imags	93.189	94.062	92.320
feat_ext_FC_reals	92.343	93.299	91.392
feat_ext_FC_multi	**97.927**	**98.340**	**97.497**

B. CNN Results

TABLE V. CNN ACCURACIES WITH RAW DATA

Raw CNN Trainings	Total	Real	Fake
basic_CNN_absolutes	97.546	98.040	97.058
basic_CNN_angles	98.265	99.317	97.231
basic_CNN_cum3s	98.138	97.829	98.456
basic_CNN_imags	98.730	98.958	98.514
basic_CNN_reals	97.800	97.174	98.411
google_gray_absolutes	97.250	98.585	96.025
google_gray_angles	98.857	98.914	98.799
google_gray_cum3s	97.842	98.271	97.431
google_gray_imags	98.392	98.397	98.387
google_gray_reals	97.842	99.048	96.688
google_RGB_absolutes	97.165	98.297	96.047
google_RGB_angles	98.561	99.171	97.925
google_RGB_cum3s	98.604	98.860	98.364
google_RGB_imags	98.604	99.234	97.979
google_RGB_reals	97.504	98.090	96.947
pt_google_absolutes	97.927	98.852	96.940
pt_google_angles	98.477	98.911	98.034
pt_google_cum3s	97.419	97.928	96.888
pt_google_imags	98.730	98.891	98.572
pt_google_reals	98.350	98.114	98.579
resnet_gray_absolutes	97.588	98.012	97.183
resnet_gray_angles	98.350	99.073	97.621
resnet_gray_cum3s	97.335	96.908	97.772
resnet_gray_imags	98.730	98.876	98.591
resnet_gray_reals	97.335	97.751	96.904
resnet_RGB_absolutes	97.461	98.198	96.744
resnet_RGB_angles	98.815	99.230	98.410
resnet_RGB_cum3s	98.434	99.104	97.709
resnet_RGB_imags	98.519	99.253	97.756
resnet_RGB_reals	98.181	97.603	98.830
pt_resnet_absolutes	97.842	98.322	97.354
pt_resnet_angles	98.265	99.394	97.183
pt_resnet_cum3s	97.842	97.986	97.696
pt_resnet_imags	98.265	99.163	97.345
pt_resnet_reals	98.223	98.287	98.160
google_multi	98.984	99.161	98.804
basic_CNN_multi	98.519	98.094	**98.962**
resnet_multi	**99.069**	**99.491**	98.647

C. SVM Results

TABLE VI. SVM ACCURACIES WITH RAW DATA

Raw SVM Trainings	Total	Real	Fake
SVC_absolutes	79.441	59.777	**98.742**
SVC_angles	95.346	95.238	95.454
SVC_cum3s	95.219	92.346	98.127
SVC_imags	98.434	98.549	98.308
SVC_reals	96.954	96.523	97.404
linear_SVC_absolutes	93.993	93.133	94.829
linear_SVC_angles	91.835	90.587	93.034
linear_SVC_cum3s	95.008	92.194	97.687
linear_SVC_imags	96.065	93.765	98.385
linear_SVC_reals	94.712	95.563	93.936
SVC_multi	**98.604**	**98.994**	98.205
linear_SVC_multi	98.011	97.420	98.623

TABLE VII. SVM ACCURACIES WITH PCA & UMAP 64 DIMENSIONS

Dimensionality Reduction SVM Trainings (64)	Total	Real	Fake
dim_red_SVC_PCA_absolutes	88.620	80.928	96.352
dim_red_SVC_UMAP_absolutes	48.646	14.852	82.612
dim_red_SVC_PCA_angles	94.204	92.537	95.755
dim_red_SVC_UMAP_angles	63.451	51.823	75.021
dim_red_SVC_PCA_cum3s	83.967	69.166	99.226
dim_red_SVC_UMAP_cum3s	73.434	51.445	95.202
dim_red_SVC_PCA_imags	97.927	97.508	98.362
dim_red_SVC_UMAP_imags	87.182	75.515	98.500
dim_red_SVC_PCA_reals	96.235	94.991	97.429
dim_red_SVC_UMAP_reals	56.218	15.169	99.478
dim_red_linear_SVC_PCA_absolutes	87.351	89.277	85.260
dim_red_linear_SVC_UMAP_absolutes	75.000	70.660	79.007
dim_red_linear_SVC_PCA_angles	88.790	86.111	91.496
dim_red_linear_SVC_UMAP_angles	79.314	82.431	76.096
dim_red_linear_SVC_PCA_cum3s	89.297	85.454	93.327
dim_red_linear_SVC_UMAP_cum3s	72.250	90.392	53.641
dim_red_linear_SVC_PCA_imags	94.247	92.595	95.955
dim_red_linear_SVC_UMAP_imags	88.155	78.132	98.297
dim_red_linear_SVC_PCA_reals	93.020	93.665	92.372
dim_red_linear_SVC_UMAP_reals	88.113	90.586	85.516
dim_red_SVC_PCA_multi	**98.688**	**99.067**	98.310
dim_red_SVC_UMAP_multi	71.150	43.238	**99.828**
dim_red_linear_SVC_PCA_multi	97.081	96.277	97.884
dim_red_linear_SVC_UMAP_multi	88.367	80.577	95.659

TABLE VIII. SVM ACCURACIES WITH PCA & UMAP 256 DIMENSIONS

Dimensionality Reduction SVM Trainings (256)	Total	Real	Fake
dim_red_SVC_PCA_absolutes	96.446	95.480	97.471
dim_red_SVC_UMAP_absolutes	49.703	22.036	78.130
dim_red_SVC_PCA_angles	95.896	94.481	97.345
dim_red_SVC_UMAP_angles	58.714	56.711	60.750
dim_red_SVC_PCA_cum3s	94.458	90.369	98.417
dim_red_SVC_UMAP_cum3s	55.076	22.035	88.510
dim_red_SVC_PCA_imags	98.434	98.477	98.392
dim_red_SVC_UMAP_imags	88.071	76.434	98.781
dim_red_SVC_PCA_reals	95.981	96.967	95.041
dim_red_SVC_UMAP_reals	69.373	39.424	99.323
dim_red_linear_SVC_PCA_absolutes	93.527	92.495	94.499
dim_red_linear_SVC_UMAP_absolutes	51.988	5.416	**100**
dim_red_linear_SVC_PCA_angles	91.370	89.217	93.410
dim_red_linear_SVC_UMAP_angles	76.057	79.115	72.790
dim_red_linear_SVC_PCA_cum3s	95.262	94.893	95.626
dim_red_linear_SVC_UMAP_cum3s	79.695	63.423	95.136
dim_red_linear_SVC_PCA_imags	95.939	94.840	97.002
dim_red_linear_SVC_UMAP_imags	84.644	84.060	85.238
dim_red_linear_SVC_PCA_reals	94.162	93.595	94.719
dim_red_linear_SVC_UMAP_reals	89.847	87.290	92.465
dim_red_SVC_PCA_multi	**98.773**	**98.639**	98.905
dim_red_SVC_UMAP_multi	71.700	42.374	99.669
dim_red_linear_SVC_PCA_multi	97.081	97.278	96.897
dim_red_linear_SVC_UMAP_multi	68.062	98.359	38.971

TABLE IX. SVM ACCURACIES WITH FEATURE EXTRACTION

Feature Extraction SVM Trainings	Total	Real	Fake
feat_ext_SVC_absolutes	63.367	30.000	94.444
feat_ext_SVC_angles	58.629	19.156	98.303
feat_ext_SVC_cum3s	51.522	9.621	95.905
feat_ext_SVC_imags	60.152	21.440	96.947
feat_ext_SVC_reals	54.399	10.136	98.067
feat_ext_linear_SVC_absolutes	86.082	78.351	94.067
feat_ext_linear_SVC_angles	90.016	86.530	93.450
feat_ext_linear_SVC_cum3s	88.155	85.762	90.569
feat_ext_linear_SVC_imags	90.270	95.042	85.434
feat_ext_linear_SVC_reals	86.252	77.004	96.147
feat_ext_SVC_multi	75.169	50.171	**99.416**
feat_ext_linear_SVC_multi	**96.065**	**95.274**	96.861

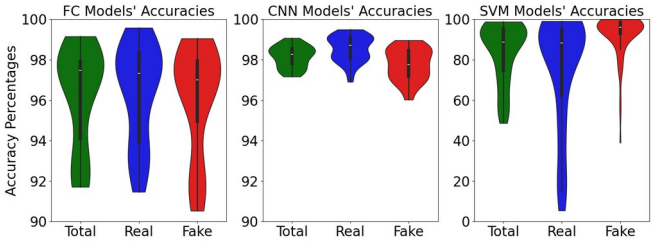

Fig. 7. Models' accuracy distributions

Low SVM scores in Real and Fake Accuracy suggest that SVM models struggle to learn. This might indicate poor data quality if FC models also had low scores. However, since FC models perform well, it highlights the SVM models' limitations in learning complex patterns. The accuracy ranges —FC models: (91.709, 99.154), SVMs: (48.6463, 98.7732), CNN models: (97.1658, 99.0693)—further show that deep learning techniques are much more effective for this task.

When models' accuracy scores are shown, it is obviously seen that CNN models are more robust than FC models with a smaller range of distribution and more densely packed center around 98%. While FC models are scattered with a lower bound around 90%, and a center below the 98%. When SVM models' results are examined it is seen that the center of distribution is around 90%, where lower bound of the distribution drops lower than 50% for Total, 10% for Real and 40% for Fake accuracy.

D. Comparison of Models

Overall, multi-form inputs yielded the best results. PCA outperformed UMAP in preserving distinguishable features during dimensionality reduction. Among the top models, there is one SVM, two FC, and seven CNN models in the top ten, and four SVMs, three FC, and thirteen CNN models in the top 20. Which confirms that CNN models generally performed better, although one model slightly surpassed them with an accuracy difference of 0.08%. When all aspects are considered as a whole; CNN models are the outstanding ones. It is also supported with the CNN models' smaller storage requirements compared to same level performing models as shown in the Fig. 8. Detailed analysis of model types advantages and disadvantages also supports this claims. And it is also seen that CNN models are better fit to the problem, when advanatages and disadvantages during and after the train of models listed.

1) FC Pros and Cons:

✓ Successful training scores at each training (all scores are over 91%)

✓ Ease of working with big datasets and updatable model with batch/online training

✘ Massive input layer size in raw input trainings, hence massive model files in size (up to 600-650 Megabytes)

✘ Long training time at big models (longest training session is around 36 hours)

✘ Preprocessing needed for to make data available to training

✘ New model needed for different formatted data because of fixed input size

2) CNN Pros and Cons:

✓ Successful training scores at each training (all scores are over 97%)

✓ Ease of working with big datasets and updatable model with batch/online training

✓ Relatively smaller models when compared to other models in same metric level (at most 100 Megabytes)

✓ Ability to process different formatted data because of fully convolutional network architecture

✘ Long training time at big models (longest training session is around 26 hours)

3) SVM Pros and Cons:

✓ Faster training times compared to other models (longest training session is around 7 hours)

✓ Small models (at most 100 Megabytes, except C-Support Vector Classification model on raw data trainings which is around 300-350 Megabytes)

✘ Although some SVMs perform well, in general, they exhibit poorer performance and lower scores compared to FC and CNN models (lowes score is below 50%)

✘ Difficulty of working with big datasets (Reshaping the image features and decreasing floating point precision were done in some cases to load training dataset to RAM) and unability to update the model since batch training is not possible on SVMs.

✘ New model needed for different formatted data because of fixed input size

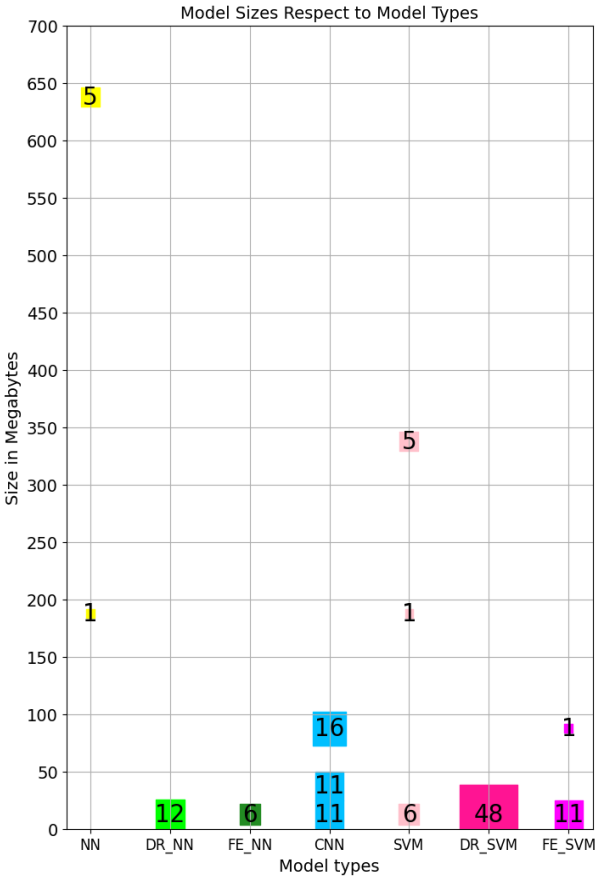

Fig. 8. Number of model types in each storage size range.

Considering model size, speed, accuracy score, updatability, ease of training and need of additional pipeline processes, it seems that CNN models are go to solutions.

When it comes to inference times, SVMs are the fastest ones (excluding the ones that use extracted features because of feature extraction latency) as expected, after that FC models are ahead of CNN models with a very small margin (again excluding the ones that use extracted features). But all models are able to run on real time applications. And when it comes to accuracy scores, it is seen that CNN models are dominating the task, and it is easier to find a CNN model that fits the task and train it, rather than experimenting with different data pre-processing pipelines for other models.

As model results state, features extracted from audio bispectrums can be used as an indicator feature for classifying synthetic audio and real audio. With proper architectures and sufficiently diverse datasets, it is able to detect synthetic audios among real ones with almost perfect score.

Although the results are good, it is crucial to acknowledge that this research is based on a dataset which can be considered old given the rapid developments in the AI field. Also, since there is no augmentation applied to training data, it is possible that models learned to discriminate raw audios but not the ones with noise, or any synthetic audio that is passed through post-processes.

IV. Future Work

There is still room for exploration in this field. For example, audio datasets with various languages and audios generated by different methods can be utilized to train models. More advanced model architectures can be experimented with. Crucially, comparing the detection performance of existing techniques with those developed in this research is essential. Possible improvements on this topic are discussed below.

Various audio signals must be also detected, such as audio signals that are re-recorded, generated by newer generative-AI models, multilingual or audio signals with additive noise. These audio variations can be found from additional datasets such as Fake or Real (FoR) [21] for especially re-recorded audios, WaveFake [22] and ASVspoof [23] for more variety of fake audios, VCTK [24], LJ Speech [25] and LibriSpeech [26] for more real audio and CommonVoice by Mozilla [27], CML-TTS [28], Multilingual TEDx [29] and Multilingual LibriSpeech [30] for more real multilingual audio. Multilingual fake audios can be generated with published generative-AI models such as bark [31], TTS [32], OpenVoice [33], StyleTTS2 [34] and VALL-E-X [35]. Researching these topics also can give us better insight on "what does models use as a 'distinguishing thing' on features to detect synthetic audios".

Because synthetic audio signals can placed between parts of real audios, these detection techniques may fail to detect presence of planted synthetic audios since models are performing by looking at audio over features that describes the whole audio as one summary image. Transformers models or recently growing MAMBA [36] models can be trained on sequence of features obtained from the same audio, where an audio is splitted into parts and feature sequence is created from these part sequences. This way, detection can evolve to showing probabilities of each audio slices being synthetic. Also, to solve this problem, models can be trained on a newly created dataset that contains audios with their labels that indicate which parts of the audio are real and fake. But without no additional dataset creation or model training effort, detection on slices of audio can performed rather than all audio at once with shown trained models. The technique used in this research generates one set of features using all the audio and creates features that are summarizing the audio as whole. Instead of this, audio can be chopped to slices, each containing S samples with overlap of V samples. Each of these slices can be treated as an audio itself, resulting in multiple detection values apart from each other with V samples across the audio.

For the purpose of methods performance, 16 different experiments with one real and one fake audio sample from the same speaker are shown in four different groups. Figures are grouped by different experimental attacks that can be explained as "randomly planting fake audio parts with randomly selected lengths from experimental attack ranges into the real audio". Experimental attack ranges are (3200-6400), (6400-9600), (9600-12800), (12800-16000) samples respectively for each figure, and they indicate the audio part sizes which vary from planting fake audio with lengths from 0.2 to 1 second.

Fake and Real audios are shown with red and waves respectively. Detection results are normalized between -0.5 and 0.5 range to sit on top of the same axis with audio waves, positive values are fake detections and negative values are real detections. Each slice’s prediction value is shown at the middle of the slice. Y=0 axis corresponds to 0% confidence, where -0.5 and 0.5 are 100% confidence in fake detection and real detection respectively. Confidence limits are also drawn with dotted lines. ResNet50 multi form model (resnet_multi) is used for demonstration. Experiment results with annotation and graph placement explanations are shown in the Fig. 9.

For each group, four experiments are done with slice size and overlap pairs of (2400, 1200), (4000, 2000), (6400, 3200), (8000, 4000) samples. Lower slice size resulted in more dense detection values but less accurate values. Detections at the end of audios may be less accurate than the others since last slices overflows outside the audio if not fitted fully. There are few flaws with this method as expected since the used model is not tailored to this type of use.

Silent parts are detected as real, no matter what the slice size is, even with the slices that are fully silent. This could be caused by models learning to distinguish using features from loud parts and ignoring the features from silent parts. A solution to this issue could be training models on silent parts of audios to be able to perform detection of silent parts. And if that approach results are not satisfying, it comes to the second possible cause which suggests silent parts not having distinguishing features.

Big slices with mixed audio are mostly detected as the audio type with more samples, which is also supported by the multiple other experiments using the features created from mixed audios from different speakers which results in a detection type of audio type with more sample in mixed audio with high confidence.

There are no obvious choices of slice size and overlap, but better results can be obtained with different parameters. This research uses audios with sample rate of 16 kilohertz and generates features from 400 samples, another study with higher sample rate and/or smaller feature extraction sample count could perform better.

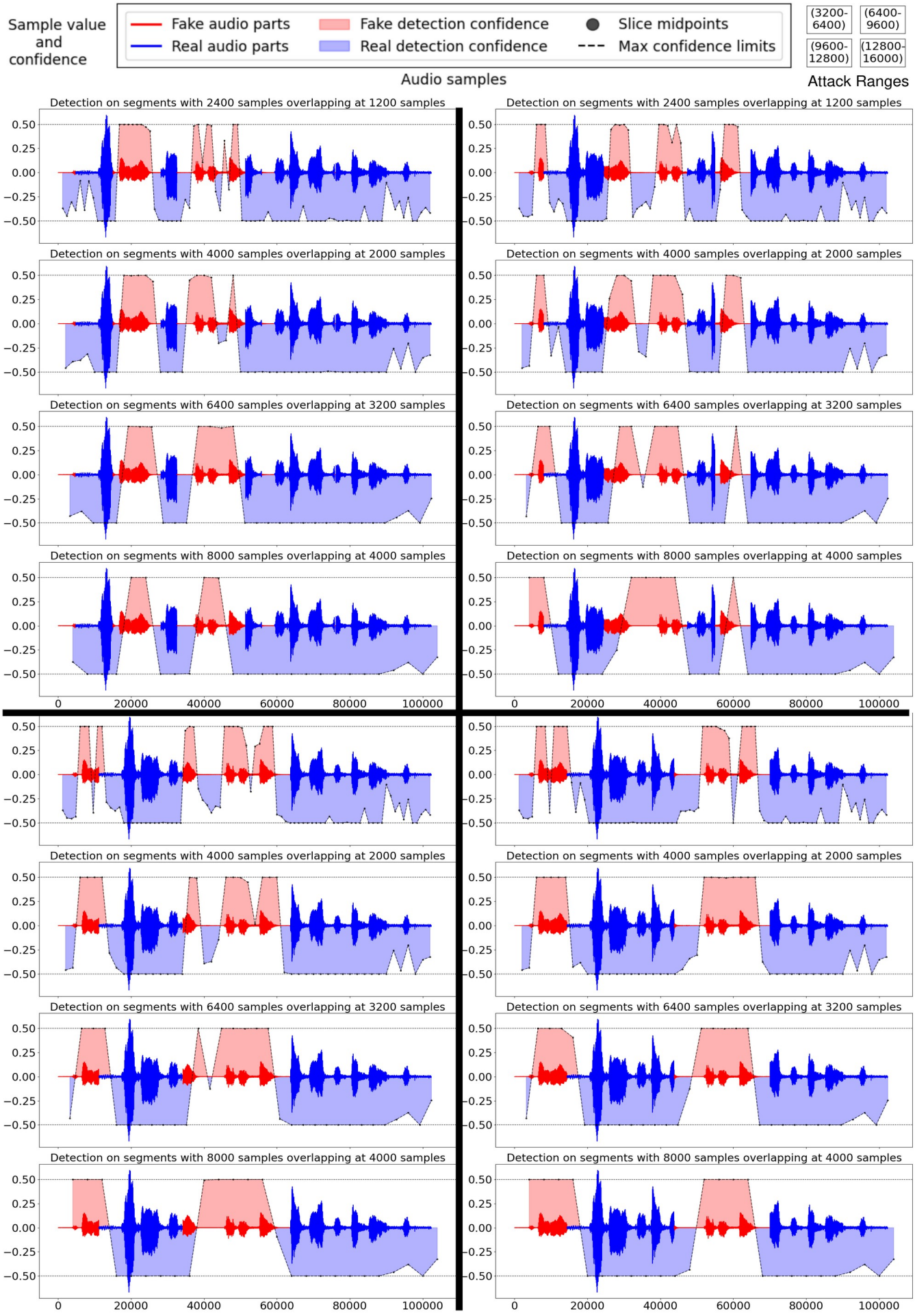

Fig. 9. Experiment with attack ranges

It is essential to experiment with various features and model architectures. A model pool with only three models and a feature pool with only five features is enough to prove that this is possible but probably not the optimal solution to the problem. More models can be added to model pool such as AlexNet, VGG and other GoogLeNet variations. Also, improvements to basic_CNN architecture can be done such as adding dropout layers, increasing the number of layers or adding skip connections, inception modules, squeeze-and-excitation blocks. We will explore more advanced techniques such as attention mechanisms, transformer-based models, and adversarial training on future experiments. Other audio features such as “power spectrum” and more can also be experimented with. Searching new architectures and features will lead the way to "finding the best architecture for processing audio features for classification".

To improve the performance of generative-AI models in this field, to-be-developed successful detection models can be used to teach generative models where are they making mistakes, how can they be more realistic. By using a proficient Vision Transformer or MAMBA model that was described above to act as a "discriminator" as in GANs, one could attempt to teach generative models exactly where they fail to sound realistic and why they fail.

V. Conclusion

In this work, various ways to detect AI-generated audio from real ones using both machine and deep learning methods are presented and compared the possible solutions in all aspects. Multiple CNN, FC architectures and different SVMs on image features explained in previous sections are trained. Best-performing model achieved an accuracy score of 99.154%. Specifically, out of the 134 training instances, two accuracy scores fell within the range of [99%, 100%), 32 within [98%, 99%), and 26 within [97%, 98%).

For synthetic audio detection task, after examining all properties as a whole, although there are models which pass CNN models on different points with small differences while falling behind on other points with bigger differences, CNN models are the best fit solution for using/training suited to the task is shown by the results.

Our following research will be extended based on the ideas presented in section "IV. Future Work" to search for and find more distinguishing features and better models for the detection task over multiple languages. A vision transformer and a MAMBA model are planned to be trained on the most distinguishing features found so far for detecting planted synthetic parts in real audios. Lastly, when a successful model is obtained, better generative models can be trained using feedback from that model.

References

[1] Ashish Vaswani et al., "Attention Is All You Need", June 2017

[2] Aditya Ramesh et al., "Zero-Shot Text-to-Image Generation", February 2021

[3] Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser and Björn Ommer, "High-Resolution Image Synthesis with Latent Diffusion Models", April 2022

[4] Matthew Le et al., "Voicebox: Text-Guided Multilingual Universal Speech Generation at Scale", October 2023

[5] Md Sahidullah, Tomi Kinnunen and Cemal Hanilçi, "A Comparison of features for synthetic speech detection", September 2015

[6] Zaynab Almutairi and Hebah Elgibreen, "A review of modern audio deepfake detection methods: challenges and future directions", May 2022

[7] Jiangyan Yi et al., "Audio deepfake detection: a survey", Aug 2023

[8] Hany Farid, "detecting digital forgeries using bispectral analysis", August 2001

[9] Alessandro Pianese, Davide Cozzolino, Giovanni Poggi and Luisa Verdoliva, "Deepfake audio detection by speaker verification", September 2022

[10] Ehab A. AlBadawy, Siwei Lyu and Hany Farid, "Detecting AI-synthesized speech using bispectral analysis", June 2019

[11] Arun Kumar Singh and Priyanka Singh, "detection of AI-synthesized speech using cepstral & bispectral statistics", April 2021

[12] Ameer Hamza et al., "Deepfake audio detection via MFCC features using machine learning", January 2022

[13] Tianyun Liu, Diqun Yan, Rangding Wang, Nan Yan and Gang Chen, "Identification of fake stereo audio using SVM and CNN", June 2021

[14] Mohammed LataifehNassif, Ashraf Elnagar, Ismail Shahin and Ali Bou Nassif, "Arabic audio clips: Identification and discrimination of authentic cantillations from imitations", December 2020

[15] Janavi Khochare, Chaitali Joshi, Bakul Yenarkar, Shraddha Suratkar and Faruk Kazi, "A Deep learning framework for audio deepfake detection", November 2021

[16] Chengzhe Sun, Shan Jia, Shuwei Hou and Siwei Lyu, "AI-synthesized voice detection using neural vocoder artifacts"

[17] Tianxiang Chen, Avrosh Kumar, Parav Nagarsheth, Ganesh Sivaraman and Elie Khoury, "Generalization of audio deepfake detection", April 2023

[18] Yipin Zhou and Ser-Nam Lim, "Joint audio-visual deepfake detection", October 2021

[19] Pedregosa et al., "Scikit-learn: Machine Learning in Python", JMLR 12, pp. 2825-2830, 2011.

[20] Nicolas M. Müller, Pavel Czempin, Franziska Dieckmann, Adam Froghyar and Konstantin Böttinger, "Does audio deepfake detection generalize?", Interspeech, 2022

[21] Ricardo Reimao, Vassilios Tzerpos, "FoR: A Dataset for Synthetic Speech Detection", October 2019

[22] Frank, J., & Schönherr, L. (2021). WaveFake: A data set to facilitate audio DeepFake detection (1.2.0) [Data set]. Zenodo. https://doi.org/10.5281/zenodo.5642694

[23] "Automatic Speaker Verification and Spoofing Countermeasures Challenge", https://www.asvspoof.org/

[24] Yamagishi, Junichi; Veaux, Christophe; MacDonald, Kirsten. (2019). CSTR VCTK Corpus: English Multi-speaker Corpus for CSTR Voice Cloning Toolkit (version 0.92), [sound]. University of Edinburgh. The Centre for Speech Technology Research (CSTR). https://doi.org/10.7488/ds/2645.

[25] Keith Ito and Linda Johnson, "The LJ Speech Dataset", https://keithito.com/LJ-Speech-Dataset/, 2017

[26] Vassil Panayotov, Guoguo Chen, Daniel Povey and Sanjeev Khudanpur, "LibriSpeech: an ASR corpus based on public domain audio books", August 2015

[27] "Common Voice dataset by Mozilla", https://commonvoice.mozilla.org/en/datasets

[28] Oliveira, F., Casanova, E., Junior, A., Soares, A., & Galvao Filho, A. (2023). CML-TTS: A Multilingual Dataset for Speech Synthesis in Low-Resource Languages. In Text, Speech, and Dialogue (pp. 188–199). Springer Nature Switzerland.

[29] Elizabeth Salesky et al., "Multilingual TEDx Corpus for Speech Recognition and Translation", Proceedings of Interspeech, 2021

[30] Vineel Pratap, Qiantong Xu, Anuroop Sriram, Gabriel Synnaeve, & Ronan Collobert (2020). MLS: A Large-Scale Multilingual Dataset for Speech Research. ArXiv, abs/2012.03411.

[31] "bark", 2023, https://github.com/suno-ai/bark

[32] The Coqui TTS Team, "A deep learning toolkit for Text-to-Speech, battle-tested in research and production", January 2021, https://github.com/coqui-ai/TTS

[33] Qin, Zengyi and Zhao, Wenliang and Yu, Xumin and Sun, Xin, "OpenVoice: Versatile Instant Voice Cloning", 2023, arXiv preprint arXiv:2312.01479

[34] Yinghao Aaron Li, Cong Han, Vinay S. Raghavan, Gavin Mischler, Nima Mesgarani, "StyleTTS 2: Towards Human-Level Text-to-Speech through Style Diffusion and Adversarial Training with Large Speech Language Models", November 2023

[35] "VALL-E X: Multilingual Text-to-Speech Synthesis and Voice Cloning", 2023, https://github.com/Plachtaa/VALL-E-X

[36] Albert Gu and Tri Dao, "Mamba: Linear-Time Sequence Modeling with Selective State Spaces", May 2024