Defeating the Fakes: Vocal Fake Detection using Discrete Fourier Transform in Neural Networks

Tina Tami
University of Amsterdam
Amsterdam, Netherlands
tina.tami@os3.nl

Lars Tijsmans
University of Amsterdam
Amsterdam, Netherlands
lars.tijsmans@os3.nl

Supervisor
Zeno Geradts
Netherlands Forensic Institute
The Hague, Netherlands

***Abstract*—We examine two methods to distinguish real voices from deepfake voices. One method uses Fourier transform arrays in a fully connected neural network to classify the voices. The second method uses spectrogram images in a convolutional neural network. We compare the two models in terms of accuracy and computational performance. Moreover, two different datasets are used to evaluate performance on unseen data. Results show that discrete Fourier transforms can be used to successfully detect deepfake voices, reaching an accuracy of 99.9% when using spectrogram images. Though using discrete Fourier transform arrays as input reaches a lower accuracy of 88.6%, training this model is at least 20 times faster.**

***Index Terms*—audio waveform, deepfake detection, discrete Fourier transform, neural network, spectrogram**

I. Introduction

In April of this year, 2021, asset manager Van Lanschot announced they will use voice recognition to identify their clients [1]. They claim to be the first financial institution in the European Union using this so-called *bio-metric voice passport*. Some British banks, including Lloyds Bank, Barclays and HSBC, already use voice identification, as does the American bank Chase [2]. The question rises if this form of identification is safe enough, being spoof-proof for people with malicious intents. Advances in artificial intelligence (AI) deepfake[1] techniques have shown that not only is it possible to generate synthetic voices that sound like real people [3] [4], but that these applications are readily available for use by any internet user. The tools range from the option between a few pre-trained voices [5], to frameworks that turn a five-second audio recording into a cloned voice that you can configure to say anything you want [6]. Such innovations can have positive applications: It can be valuable for people who have lost the ability to speak and wish to being able to synthetically talk to their loved ones. Moreover, it can be used in cases where artificial customer support agents get their own synthetic voice in order to provide clients the feeling of speaking to an actual employee. The downside, however, is that these tools can also be used maliciously in order to make people say things they have not actually said, or in order to fake their identity for illegal purposes such as unwanted money transfers.

[1]Deepfakes (a portmanteau of "deep learning" and "fake") are synthetic media where techniques from machine learning and artificial intelligence are used to manipulate or generate visual and audio content.

In this paper, we will examine two methods to distinguish deepfake voices, from now on referred to as *vocal fakes*, from real, human voices. Both methods involve the use of discrete Fourier transform (DFT) and a neural network that classifies the voice as either a *real voice* or *fake voice*. The first method, using a 2D convolutional neural network (CNN), will take spectrogram images as input. The second method will use the DFT arrays as input for a fully connected neural network (FCNN). These terms will be further explained in section III. Additionally, we will be using two different datasets to conduct our experiments, in order to see how our models perform on new, unseen data. The goal is to see if discrete Fourier transform would be a reliable approach to detect such vocal fakes, and if so, which of the two methods is better suited for the job.

A. Research question

Can discrete Fourier transform be used to distinguish vocal fakes from real voices in a classification model?

As mentioned above, we will be examining two different methods and two different datasets to answer this question. Therefore, the following subquestions are raised:

- How do discrete Fourier transform arrays and spectrograms compare in accuracy?
- How do discrete Fourier transform arrays and spectrograms compare in computational performance?
- How do the models perform on different datasets?

II. Related work

Deepfake software for audio is becoming increasingly better. It is desirable to provide for detection mechanisms that work on yet unknown attacks. In [7], the question is raised on how to adapt detection models to new sources of fake speech. They present four parameter-efficient convolutional architectures for fake speech detection using transfer learning. The log mel spectrogram of both real and spoofed audio is used as input. They assume that for a new attack vector, they have access to a small portion of the data in order to fine-tune the classifier in production. Results show that for all four architectures, only 6.25% to 25% of target data is needed to meet or exceed supervised performance. This research focused on using a small dataset to minimize training time. We, on

the other hand, will examine if input in other forms than spectrograms give similar results, in order to minimize training time.

Some researched methods do not use audio recordings. In DeepSonar [8], a solution is proposed to detect AI synthesized voices by monitoring the learnt neuron behaviors from voice synthesis systems. They trained a binary-classifier, being a shallow neural network, to predict whether a clip of voice is human speech or an AI-synthesized voice. The inputs of the binary-classifier are vectorized captured layer-wise neuron behaviors rather than raw input voices, which they claim are better for a simple classifier to learn the differences between real and fake voices. DeepSonar proved to be effective, however there are also some limitations. When the audio contains too much noise, the detection algorithm becomes much less robust.

Furthermore, not all existing methods make use of deepfake generated voices. In [9], the imitation of voices is used to create fake recordings. They assume that any voice signal (original) may seem similar to another voice signal (target) if its wavelet coefficients are sorted. The entropy features of original and forged audio were manually extracted, and a machine learning model with logistic regression was used to classify the audio recordings as either real or fake. They were able to achieve an accuracy of 98% where all forged audios were successfully detected.

In [10], a method is proposed to apply watermarking to an audio file, which can be used to watermark music. The paper describes a trade-off between robustness and imperceptibility, meaning that the watermark should be robust enough so it can be detected, while not being too noticeable when listening to the watermarked music. They use the mid-frequency range to apply a watermark to an audio sample. The signal-to-noise ratio is used to prove that the imperceptibility of the watermark is limited when just listening to the audio. This paper shows that it is possible for audio recordings to contain artifacts that are not easily perceivable by simply listening, but can be shown in the frequency domain. This finding is interesting for our research as deepfake voice recordings can also contain artifacts that might not be audible, but are visible when transformed to the frequency domain.

III. Background

In this section, we provide background information to ensure that our methodology and experiments can be understood.

A. Neural network

An artificial neural network is a computational learning system that uses a network of functions to understand and translate a data input into a desired output. The concept of the artificial neural network was inspired by human biology, and the way neurons of the human brain function together to understand inputs from human senses. The network processes many labeled examples that are supplied during training to learn what characteristics of the input are needed to construct the correct output. Once a sufficient number of examples have been processed during training, the network can begin to process new, unseen data and make predictions [11]. The neurons are aggregated into layers, and different layers perform different transformations on their inputs. Input travels from the first layer (the input layer), to the last layer (the output layer), after going through multiple hidden layers. There are many kinds of neural networks. We will explain the two types we will be using.

A fully connected neural network consists of a collection of nodes or neurons that are all connected to each other. Each connection can transmit a signal (being a real number) to other neurons, where the signal gets processed. The output of each neuron is computed by a non-linear function of the sum of its inputs. During learning, the weights of the neurons and connections are increased or decreased in order to get the correct output. In a 2D convolutional neural network, the network employs a mathematical operation called a convolution, using a kernel that slides along two dimensions. A 2D CNN is often used for image classification. The main difference between a traditional, fully connected neural network and a convolutional neural network is that only the last layer of a CNN is fully connected, whereas in a FCNN, each neuron is connected to every other neuron [12]. We will briefly discuss the different layers that will be used in our models.

1) Convolution layer: The mathematical operation of convolution is performed in this layer. This is done between the input image and a kernel or filter. By sliding the kernel over the input image, the dot product is taken between the kernel and the parts of the input image that the filter covers. The filter is designed to detect a specific type of feature in the input, resulting in a feature map only highlighting the parts of the image that contain this feature. A convolution layer has multiple kernels that perform this operation, and every kernel is looking for different aspects in the image. Convolution layers are not only applied to input images, but also to the output of other layers.

2) Pooling layer: Often, a convolution layer is followed by a pooling layer. The primary goal of this layer is to decrease the size of the feature map obtained in the previous layer, in order to reduce the computational costs. In max-pooling, the kernel slides over the feature map in order to generate a smaller feature map, taking the largest element of every NxN pixels, where N is the size of the kernel.

3) Flatten layer: The feature maps received from the max-pooling layer are 2D arrays. In the flattening layer, all 2D arrays are turned into a single long vector in order to be passed on to the fully connected layers.

4) Fully connected layer: After the desired amount of convolution and max-pooling layers, there are a few fully connected (or dense) layers. These layers learn the classification task. Each neuron in the neural network has a weight, and this weight is updated throughout the fully connected layers in order to get the desired output.

5) Activation functions: An important parameter of the model is the activation function. The activation function decides if the neuron in the neural network should be activated

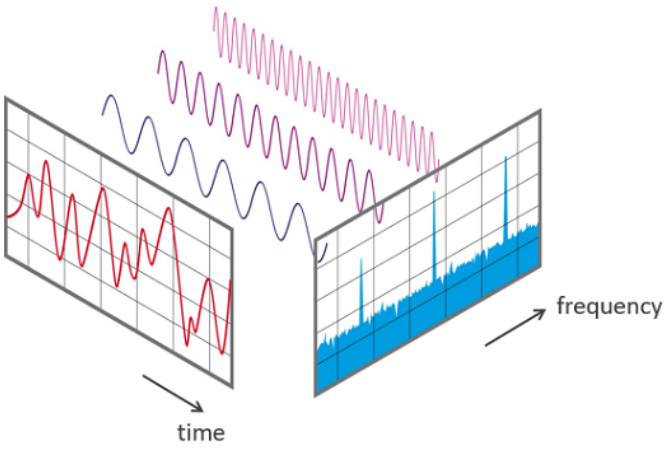

Fig. 1. A visual representation of Fourier transform applied on a waveform [14]. The left side is the audio waveform before Fourier transform, being amplitude (y-axis) vs. time (x-axis). The right side is the audio waveform after Fourier transform, being amplitude (y-axis) vs frequency (x-axis).

or not. The ReLU, short for rectified linear activation function, is a function that sets all negative pixels to zero and introduces non-linearity to the network. The sigmoid activation function is often used for binary tasks, as it exists between zero and one. This function will output the probability that the input belongs to a certain class.

B. Discrete Fourier transform

In this research the discrete Fourier transform will be used, since we are working with audio signals that are digitally stored using discrete values. A Fourier transform is a mathematical model that is commonly used to convert a signal in the time spectrum to a frequency spectrum, to determine the dominant frequencies in a signal. It makes use of the fact that every non-linear function can be represented as a sum of (infinite) sine waves. A Fourier Transform will break apart a time signal, and will return information about the frequency of all sine waves needed to simulate that time signal [13]. It transforms a sequence of N complex numbers $\{x_n\} := x_0, x_1, \ldots, x_{N-1}$ into another sequence of complex numbers, $\{X_k\} := X_0, X_1, \ldots, X_{N-1}$, defined by (1) and (2).

$$X_k = \sum_{n=0}^{N-1} x_n \cdot e - \frac{i2\pi}{N} kn \tag{1}$$

$$X_k = \sum_{n=0}^{N-1} x_n \cdot \left[cos(\tfrac{2\pi}{N} kn) - i \cdot sin(\tfrac{2\pi}{N} kn)\right] \tag{2}$$

Where N = the number of samples, n = the current sample, x_n = the value of the signal at time n, k = the current frequency (0 Hz to N-1 Hz) and X_k = the result of the DFT [13].

A spectrogram is a more advanced Fourier transform, being a visual representation of the spectrum of frequencies of a signal as it varies with time. In contrary to the output of a regular DFT, the time element is re-introduced in a spectrogram. The heat of the map represents the magnitude (amplitude) of an observed frequency (y-axis) at a given time (x-axis). This is done by dividing the audio signal into smaller frames (called windows) and calculating the DFT for each window. Now, we will have the frequencies for each window, and the sequence of windows represents the time.

IV. Datasets

Two datasets were used for our experiments: An already existing dataset called ASVSpoof, and a dataset that we created ourselves, called RTVCSpeech.

A. RTVCSpeech

For RTVCSpeech, we generated our own vocal fakes. The real voices are used from an existing dataset.

1) Real voices: OpenSLR [15] provides a dataset `train-clean-100` containing 100 hours of studio quality, 16kHz read English speech, providing a total of 251 speakers. There are 28.539 audio recordings in total, and each recording has a duration of about 15 seconds. The data is derived from read audio books from the LibriVox project, so each recording has a corresponding transcript. For our research, these recordings are labeled as real voices.

2) Fake voices: Because the OpenSLR dataset only provides real voices, the vocal fakes are created using an open-source deepfake generation program called Real Time Voice Cloning (RTVC) [6]. RTVC is an implementation of Transfer Learning from Speaker Verification to Multi-speaker Text-To-Speech Synthesis (SV2TTS), shown in figure 2. SV2TTS is a three-stage deep learning framework that allows to create a numerical representation of a voice, and to use it to condition a text-to-speech model trained to generate new voices. It uses two datasets to train its model. The first dataset is used to train the model on a certain language, and should contain thousands of speakers and thousands of hours [16]. This part is done by the creator for the English language, which is our desired language. The second dataset consists of utterances of the speaker that is going to be cloned. A few seconds per utterance is enough.

By passing the real voices from OpenSLR to the encoder, the generated outputs were labeled as vocal fakes. The first audio recording of each speaker was used to create the vocal fakes of that person: By using the transcripts, the text spoken by the real voices was also spoken by the fake voices. This process was iterated for every speaker.

B. ASVSpoof

ASVSpoof is a yearly, online contest that consists of the Automatic Speaker Verification Spoofing and Countermeasures Challenge [18]. The contest provides a labeled dataset called `logical access` (LA) [19], containing 24 hours of telephony and VoIP quality audio. The dataset consists of 25.380 audio recordings in total, where 2580 recordings are real voices, and 22.800 are fake voices. The audio files are sampled at 16kHz with a total of 20 different speakers.

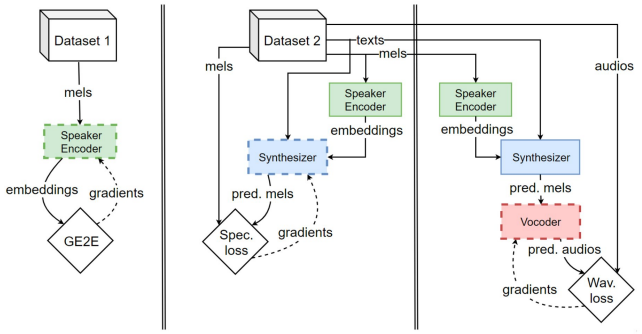

Fig. 2. A general overview of the working of Real Time Voice Cloning [17]. Dataset 1 is used to train the model on a specific language, whereas dataset 2 is used to clone a speaker.

The vocal fakes present in the dataset are generated using eight different generation methods such as spectral filtering and waveform concatenation [20].

V. Preprocessing

A. Dataset preprocessing

For the RTVCSpeech dataset, all audio recordings containing more than 20 words were ommited. This is done because RTVC, during text-to-speech voice cloning, recommends to use "plus minus 20 words" [21]. The amount of recordings left after this step will be the amount of recordings that will be used for our experiments, as shown in table I. The generated vocal fakes had some empty space at the beginning and end of the recording, meaning there was no sound present at all. Therefore, 0.01s was trimmed from the beginning of the file, and 0.1s was trimmed from the end of the file. To make sure all audio recordings have the same length, they were either truncated or looped to have a duration of 5 seconds.

In order to have two equally sized datasets, 2644 randomly selected fake voice recordings have been used from ASVSpoof. As there were not that many real voice recordings, all available real voice recordings are used, as shown in table I. These recordings were also truncated or looped to have a duration of 5 seconds.

TABLE I
Number of recordings used for each dataset.

	Real recordings	Fake recordings
RTVCSpeech	2644	2644
ASVSpoof	2580	2644

B. Input generation

The Fourier transform is calculated from each audio recording using Scipy's `scipy.fftpack`. This returns (x,y)-values representing the frequency (x-axis) and amplitude (y-axis) of each frequency. The amplitude values determine whether a frequency is present or not, and in what degree it is present. This means that all frequencies (x-values) are the same for every recording. Therefore, these values will be omitted from the arrays. The remaining values will be normalized and used as input for the FCNN. For the 2D CNN, a spectrogram is estimated from every recording using `spectrogram()`, the MATLAB built-in function for spectrogram calculation. Parameter settings are gathered using Scipy's `scipy.io.wavfile`.

VI. Experiments

Both datasets are divided into a training and test set, where 80% of the data is used for training, and 20% is used for testing. This results in a RTVCSpeech training and test set, and an ASVSpoof training and test set. Each neural network is trained two times, once for every training set. Then, each network is tested on two different datasets: The test set of the dataset it has been trained with, and the test set of the other dataset. To summarize, for each neural network:

- Experiment 1: training using RTVCSpeech, testing using RTVCSpeech.
- Experiment 2: training using RTVCSpeech, testing using ASVSpoof.
- Experiment 3: training using ASVSpoof, testing using ASVSpoof.
- Experiment 4: training using ASVSpoof, testing using RTVCSpeech.

A. Models

The generated spectrograms, as described in section V-B, are used as input for the CNN. The input image goes through four blocks of a convolution layer with a 3x3 kernel and ReLU activation, followed by a max-pooling layer with a 2x2 kernel. The output is then flattened and goes through two fully connected layers, where the first one uses ReLU activation and the second one uses sigmoid activation, in order to classify it as a real or fake voice. An overview of our model is shown in figure 3.

The input for the FCNN will be the normalized amplitude arrays as described in section V-B. The arrays go through five fully connected layers. ReLU activation is used for all our fully connected layers, with the exception of the last one. The last layer uses sigmoid activation to classify the input as a real or fake voice. An overview of this model is shown in figure 4. For both models, we measured the computational performance when using an NVIDIA GeForce RTX 2080 GPU, and an Intel Xeon 2.10 GHz CPU.

B. Metrics

Both neural networks generate a confusion matrix when predicting the classes for the test set. As our datasets are balanced, we will calculate the accuracy of the model [22], described in (3). Here, TP, TN, FN and FP stand for true positives, true negatives, false negatives and false positives, respectively. The models have been trained and tested using three different seeds, and the mean average of the resulting accuracy is taken as final accuracy for that experiment.

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP} \tag{3}$$

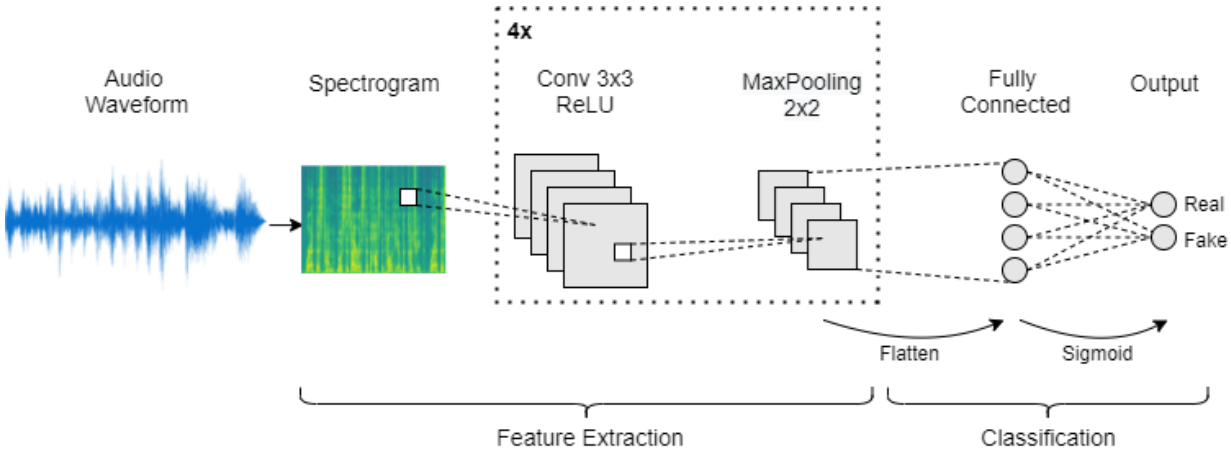

Fig. 3. Overview of the architecture used for the 2D CNN. The audio recordings are used to generate the spectrograms, which is used as input for the model.

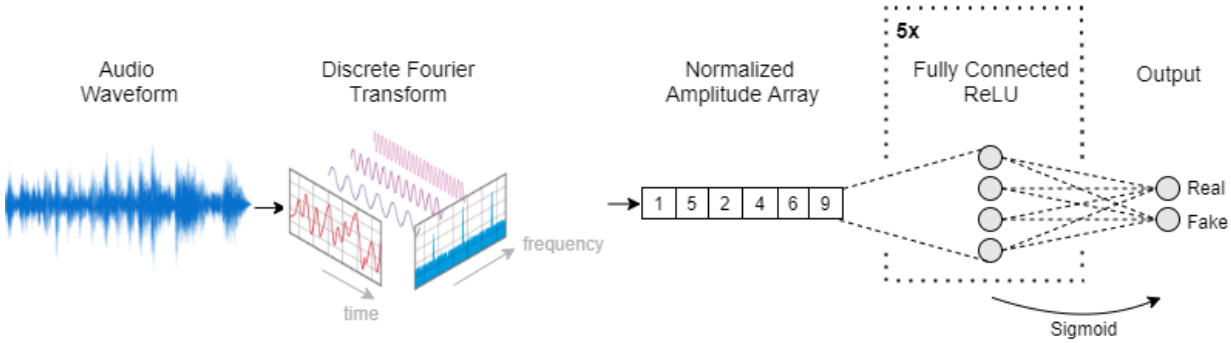

Fig. 4. Overview of the architecture used for the FCNN. The audio recordings are used retrieve the array of amplitudes of the discrete Fourier transform, which is used as input for the model.

VII. Results

The accuracy obtained using the 2D CNN and FCNN are shown in table II and III, respectively. See Appendix A for the confusion matrices generated during our experiments.

A. Model input: Spectrograms

For our first experiment using the 2D CNN, the model was trained on RTVCSpeech. Testing on the RTVCSpeech dataset results in a high accuracy of 99.9%, while testing on the ASVSpoof dataset has a much lower accuracy of 63.0%. When training our model on ASVSpoof, the network reaches an accuracy of 99.8% when testing on ASVSpoof. Testing on RTVCSpeech results in an accuracy of 66.3%. This means that training with ASVSpoof results in a slightly higher accuracy when testing with RTVCSpeech, compared to the other way around.

Looking at the computational aspect of the 2D CNN, it was crucial to do these experiments using GPU. On GPU, 1 epoch took 20 seconds, while 1 epoch on CPU took 655 seconds, being almost 33 times longer. It would be infeasible to build, improve, train and test this model on CPU within a limited time range.

B. Model input: Discrete Fourier transform arrays

Using the DFT arrays as input for our FCNN results in worse accuracy than using spectrograms. When training on RTVCSpeech, an accuracy of 88.6% is reached when testing on the same dataset, which is significantly lower than when using spectrograms as input. When testing on ASVSpoof, the accuracy drops to 56.3%. This poor accuracy implies randomly assigning classes would have the same outcome. Training the model on ASVSpoof, an accuracy of 85.4% is achieved, being slightly lower than the accuracy obtained when training and testing on RTVCSpeech. It is interesting to see that when testing on RTVCSpeech, an accuracy of 63.4% is reached. This is close to the accuracy obtained in the 2D CNN when training and testing on different datasets.

Computational wise, it took 1 second to complete 1 epoch on both CPU and GPU using the FCNN. Thus, training and testing the model on CPU is sufficient.

C. Comparing the two models

We can see that using ASVSpoof as a training set overall results in a higher accuracy for a different test set than when training on RTVCSpeech. We believe this is caused by the deepfake generation methods used in both datasets: In RTVCSpeech, the deepfake voices are all generated the same

TABLE II
ACCURACY OF THE 2D CNN, LISTED PER TRAINING AND TEST SET.

		Training set	
		RTVCSpeech	**ASVSpoof**
Test set	**RTVCSpeech**	0.999	0.663
	ASVSpoof	0.630	0.998

TABLE III
ACCURACY OF THE FCNN, LISTED PER TRAINING AND TEST SET.

		Training set	
		RTVCSpeech	**ASVSpoof**
Test set	**RTVCSpeech**	0.886	0.634
	ASVSpoof	0.563	0.854

way, being the method that RTVC uses, described in section IV-A2. For ASVSpoof, multiple different methods are used to generate the vocal fakes. Therefore, we think it does a better job in recognizing vocal fakes in general, whereas training on RTVCSpeech trains the model to accurately identify deepfake voices generated by that specific method. As is to be expected, training and testing on the same dataset (without overlap in training and test set) results in a higher accuracy than when training and testing on different datasets. The differences in the current implementation, however, have a significant difference in accuracy of at least 36.9% for the 2D CNN, and 32.3% for the FCNN. This can be caused by many reasons, such as the parameters and architecture used for the neural networks. We will elaborate on this in section IX.

During our experiments, we measured the time required to train 1 epoch. Training the FCNN is 20 (using GPU) to 655 (using CPU) times faster than training the 2D CNN. This is most likely caused by the computational time required to perform the convolutions. We could say this is a trade-off between time and accuracy, but with the current implementation of the models it is clear that the difference in accuracy is too big to choose the FCNN over the 2D CNN.

VIII. CONCLUSION

We examined two methods in order to distinguish vocal fakes using classification models. In order to do this, we used two different datasets, being RTVCSpeech and ASVSpoof, both containing audio recordings of real and fake voices. Our first model is a 2D CNN, taking spectrogram images as input. The second model, a FCNN, takes as input the normalized amplitude arrays calculated from the discrete Fourier transform. We conclude that Fourier transforms can be used to distinguish deepfake voices from real, human voices in classification models. These models do, however, require careful configuration and a well selected dataset.

With the current implementation, using spectrogram images in a 2D CNN results in a higher accuracy (highest being 99.9%, lowest being 63.0%) than using discrete Fourier transform amplitude arrays in the FCNN (highest being 88.6%, lowest being 56.3%). Moreover, we found that the selected training set has impact on the accuracy when testing on a different test set: Training on RTVCSpeech and testing on ASVSpoof gives an accuracy of 63.0% and 56.3% for the 2D CNN and FCNN, respectively. The other way around results in an accuracy of 66.3% for the 2D CNN, and 63.4% for the FCNN. This might have to do with the methods used to generate the vocal fakes, as the fake voices in RTVCSpeech are all generated using the same approach, whereas the fake voices in ASVSpoof are created in many different ways.

Though training the 2D CNN requires the use of a GPU, and is 20 times slower than training the FCNN, the loss in accuracy when using the FCNN is too significant at the moment.

IX. DISCUSSION

Even though we we were able to show that vocal fakes are detectable in a classification model, there are some enhancements that could be made to increase the performance of these models. Firstly, the neural networks used in our experiments are not fully optimized. We believe that better results, being a higher accuracy, could be obtained by optimizing the model parameters and the architecture of the neural networks as a whole. There are many different ways a neural network can be improved, e.g. increasing or decreasing the number of layers and adding other, not currently used layers. Relatively small changes, such as using regularization techniques like early stopping to prevent overfitting, could have a big impact on the accuracy of the model: When training a neural network, there will be a point where the model will stop generalizing, and start learning the statistical noise in the training dataset. Overfitting occurs when a model fits the data in the training set well, while having a larger generalization error on previously unseen data. The goal of regularization techniques is to prevent the model from doing this. Right now, the 2D CNN has higher accuracy than the FCNN. While we do believe 2D CNNs are well suited for image classification tasks [23], there might be better options for the Fourier transform amplitude arrays.

However, even if the models would be optimized and perform better, the results also depend on the generation and quality of the vocal fakes. The dataset used to train the model has impact on the accuracy it can reach. Therefore, it is important to choose a dataset where both real and fake voices are generated in different ways. Moreover, humans are often able to distinguish real voices from deepfake voices by simply listening to the audio. In the (near) future, vocal fakes might be improved in such a way that they will become unidentifiable for humans. The question is whether detection would still be possible this way with better quality vocal fakes.

X. FUTURE WORK

For future work, optimizing the model parameters and architecture, as described in section IX, would be a priority. This is not only possible by changing the current implementation, but also by looking into transfer learning. In transfer learning, a machine exploits the knowledge gained from a previous task to improve generalization about another task. It is currently popular in deep learning as it can train neural networks with comparatively little data. Research could be done on which existing, trained model can be re-purposed for this task.

Additionally, adding explainability would be desired. As the size and complexity of problems continue to increase, so does the complexity of the machine learning algorithms applied to these problems. This complexity makes it hard to understand why a certain audio recording is classified as either a real or deepfake voice. Machine learning explainability is the process of explaining and interpreting deep learning models. This helps developers to better understand and interpret the model's behavior, and makes debugging and improving the model easier [24].

Once the models have been optimized in a way that produces high accuracy and some level of explainability, it would be interesting to see how they perform in a real-time scenario. Taking the bio-metric voice passport as an example, mentioned in section I: Would their system be able to distinguish the caller's voice from a cloned, deepfake voice, and therefore confirm their identity? We believe these additional findings could be useful in the near future.

REFERENCES

[1] Van Lanschot. *Van Lanschot introduceert biometrische stemverificatie*. URL: https://www.vanlanschot.nl/nieuws/2021/van-lanschot-introduceert-biometrische-stemverificatie---nieuws-20-april-2021 (visited on 06/02/2021).

[2] Security.nl. *Van Lanschot introduceert "biometrisch stempaspoort" voor klanten*. URL: https://www.security.nl/posting/699935/ (visited on 06/02/2021).

[3] Yu Gu and Yongguo Kang. *Multi-task WaveNet: A Multi-task Generative Model for Statistical Parametric Speech Synthesis without Fundamental Frequency Conditions*. 2018. arXiv: 1806.08619.

[4] Jonathan Shen et al. *Natural TTS Synthesis by Conditioning WaveNet on Mel Spectrogram Predictions*. 2018. arXiv: 1712.05884.

[5] *TTS demo*. URL: https://ttsdemo.com/ (visited on 06/02/2021).

[6] Corentin Jemine. *Real-Time Voice Cloning*. URL: https://github.com/CorentinJ/Real-Time-Voice-Cloning (visited on 06/01/2021).

[7] Nishant Subramani and Delip Rao. "Learning Efficient Representations for Fake Speech Detection". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34 (Apr. 2020), pp. 5859–5866. DOI: 10.1609/aaai.v34i04.6044.

[8] Run Wang et al. "DeepSonar: Towards Effective and Robust Detection of AI-Synthesized Fake Voices". In: (May 2020).

[9] Yohanna Rodriguez-Ortega, Dora Ballesteros, and Diego Renza. "A Machine Learning Model to Detect Fake Voice". In: Oct. 2020, pp. 3–13. ISBN: 978-3-030-61701-1. DOI: 10.1007/978-3-030-61702-8_1.

[10] Euschi Salah et al. "A Fourier transform based audio watermarking algorithm". In: *Applied Acoustics* 172 (2021), p. 107652. DOI: https://doi.org/10.1016/j.apacoust.2020.107652.

[11] Anders Krogh. "What are artificial neural networks?" In: *Nature Biotechnology* 26.2 (2008), pp. 195 –197.

[12] I. Gogul and Sathiesh Kumar. "Flower species recognition system using convolution neural networks and transfer learning". In: Mar. 2017, pp. 1–6. DOI: 10.1109/ICSCN.2017.8085675.

[13] Douglas L. Jones. "The DFT, FFT, and Practical Spectral Analysis". In:

[14] NTI Audio. *Fast Fourier Transformation FFT - Basics*. URL: https://www.nti-audio.com/en/support/know-how/fast-fourier-transform-fft (visited on 06/22/2021).

[15] MultiMedia LLC. *LibriSpeech ASR corpus*. URL: http://www.openslr.org/12/ (visited on 06/01/2021).

[16] Corentin Jemine. *Support for other languages*. URL: https://github.com/CorentinJ/Real-Time-Voice-Cloning/issues/30 (visited on 06/22/2021).

[17] Corentin Jemine. "Master thesis: Automatic Multi-speaker Voice Cloning". In: 2019.

[18] *ASVSpoof*. URL: https://www.asvspoof.org/ (visited on 06/22/2021).

[19] *ASVspoof 2019: The 3rd Automatic Speaker Verification Spoofing and Countermeasures Challenge database*. URL: https://datashare.ed.ac.uk/handle/10283/3336 (visited on 06/22/2021).

[20] Xin Wang et al. "ASVspoof 2019: A large-scale public database of synthesized, converted and replayed speech". In: *Computer Speech Language* 64 (2020), p. 101114. ISSN: 0885-2308. DOI: https://doi.org/10.1016/j.csl.2020.101114.

[21] Corentin Jemine. *Real-Time Voice Cloning Github*. URL: https://github.com/CorentinJ/Real-Time-Voice-Cloning/blob/master/demo_cli.py (visited on 06/22/2021).

[22] Sirko Straube and Mario M. Krell. "How to evaluate an agent's behavior to infrequent events?—Reliable performance estimation insensitive to class distribution". In: *Frontiers in Computational Neuroscience* 8 (2014), p. 43. ISSN: 1662-5188. DOI: 10.3389/fncom.2014.00043.

[23] M.V. Valueva et al. "Application of the residue number system to reduce hardware costs of the convolutional neural network implementation". In: *Mathematics and Computers in Simulation* 177 (2020), pp. 232–243. DOI: https://doi.org/10.1016/j.matcom.2020.04.031.

[24] Oracle. *Model Explainability*. URL: https://docs.oracle.com/en-us/iaas/tools/ads-sdk/latest/user_guide/mlx/mlx.html (visited on 07/03/2021).

APPENDIX A
CONFUSION MATRICES

TABLE IV
SPECTROGRAM IMAGES FOR 2D CNN: RTVCSPEECH AS TRAINING AND TEST TEST

Predicted class

Actual class		**Real**	**Fake**
	Real	527	1
	Fake	0	528

TP equals

	Real	**Fake**
Accuracy	0.999	0.999

TABLE V
SPECTROGRAM IMAGES FOR 2D CNN: RTVCSPEECH AS TRAINING SET, ASVSPOOF AS TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	516	0
	Fake	386	142

TP equals

	Real	**Fake**
Accuracy	0.630	0.630

TABLE VI
SPECTROGRAM IMAGES FOR 2D CNN: ASVSPOOF AS TRAINING AND TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	514	2
	Fake	0	528

TP equals

	Real	**Fake**
Accuracy	0.998	0.998

TABLE VII
SPECTROGRAM IMAGES FOR 2D CNN: ASVSPOOF AS TRAINING SET, RTVCSPEECH AS TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	198	330
	Fake	26	502

TP equals

	Real	**Fake**
Accuracy	0.663	0.663

TABLE VIII
DFT ARRAYS FOR FCNN: RTVCSPEECH AS TRAINING AND TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	449	61
	Fake	81	467

TP equals

	Real	**Fake**
Accuracy	0.886	0.886

TABLE IX
DFT ARRAYS FOR FCNN: RTVCSPEECH AS TRAINING SET, ASVSPOOF AS TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	175	334
	Fake	123	413

TP equals

	Real	**Fake**
Accuracy	0.563	0.563

TABLE X
DFT ARRAYS FOR FCNN: ASVSPOOF AS TRAINING AND TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	457	70
	Fake	83	435

TP equals

	Real	**Fake**
Accuracy	0.854	0.854

TABLE XI
DFT ARRAYS FOR FCNN: ASVSPOOF AS TRAINING SET, RTVCSPEECH AS TEST SET

Predicted class

Actual class		**Real**	**Fake**
	Real	271	235
	Fake	152	400

TP equals

	Real	**Fake**
Accuracy	0.634	0.634