Deepfake Audio Detection

Jaron Li
Department of Computer Science and Software Engineering
California Polytechnic State University - San Luis Obispo
jli213@calpoly.edu

Abstract:

With deepfake audio growing in popularity to perform malicious actions, it's important to have a successful way to detect real audio from the fake and altered audio. Considering the rise of machine learning in today's world, it's an impactful tool that can be used to help automate this process due to its efficiency and effectiveness. This research finds a way to make an effective machine learning model by utilizing the ASVspoof 2019 dataset and its numerous audio files and then converting them to Mel spectrograms to allow for feature extraction. This will provide the model with excellent information to allow the model to be trained, validated, and tested well. With the success of this model, the world can become a safer place with something in place to help prevent these attacks and this type of harm.

Introduction:

The rise of technology continues to grow every single day. Deepfake audio has become more and more popular but for the wrong reasons. Deepfake audio can be used to cause harm and to perform malicious attacks. For example, deepfake audio can be used to conduct voice phishing, which is where someone may impersonate someone else with that person's voice to perpetuate fraudulent activities and scams to retrieve private information. Another example is how people can use deepfake audio to spread false information online. This can be commonly seen with celebrities and politicians where fake audio recordings can be created with their voices to spread false information out to the world and misinform others. One last example is how people use deepfake audio to perform blackmail actions. Manipulating someone's voice to create fake conversations and recordings of them can allow the perpetrator to get what they want. These are only three ways out of many in how people can use deepfake audio for negative causes. This is why it is essential to put attention in this field to help reduce this fraud. To address this issue, the current study utilizes machine learning to help detect deepfake audio, implementing a significant impact on society's safety concerns. With the rise of machine learning, using it to help do this detection and automate this process would be very effective and efficient. It's also essential to know what type of approaches and methods are out there to help detect deepfake audio. One popular method is using spectral analysis, where a frequency of an audio

signal is analyzed. It's also important to note how deepfake audio is created in the first place, which can be through approaches including speech synthesis and voice conversion, where human speech is altered. With machine learning, knowledge of what methods and approaches are out there to detect deepfake audio, how deepfake audio is created, and on top of that, datasets with past data and audio along with research that has been done it this field, it can help create an effective machine learning program and with Python and its various libraries, it can help accurately reduce this anomaly and fraud.

Literature Review:

Anagha, et.al. [1] (2023) talks about their approach to detecting deepfake audio by using Mel-spectrograms. By using Mel-spectrograms, Anagha and their collegues were able to extract key features and have their model learn from this type of representation of the audio files. With this approach, an accuracy of 85% was able to be achieved.

Hai, et.al.[2] (2023) talks about how they mainly focused on targeting background noise and the silent parts before and after a voice is presented to detect deepfake audio. Hai and their colleagues mentioned how many detections systems don't take speaker-irrelative features (SiFs) into account when creating those models, which could contribute to why systems could be producing false results. To conduct their research, a dataset that contained speech generated from multiple spoofing algorithms was used and testing was done with 195 samples to evaluate their model. Through this process, an average success rate of 82% was presented and showed the program was effective and efficient when the number of iterations was 25 or less and an iteration being 20 seconds or less. An evaluation was also able to made that their research's model didn't perform well when the quality of speech is relatively low.

Iqbal, et.al.[3] (2022) did a good job talking about the background of deepfake audio along with their approach to their research and model. To start, Iqbal and their colleagues mentioned replay attack, speech synthesis, and voice conversion were three of the more popular approaches for deepfake audio creation. Furthermore, for replay attack, two types of this form are called far-field and cut and paste. It was also mentioned that text speaker verification was a way to guard against replay attacks, along with deep convolutional networks. Iqbal and their team also described speech synthesis to be artificially creating human speech with software or hardware programs while voice conversion is where the voice from the source is altered to make it seem like another speaker said it. For both methods of speech synthesis and voice conversion, ResNet (Residual Network) is used to detect these type of deepfake audio methods. As far as their research goes, their dataset included 195,000 audio samples, where the samples were created by a computer with recent speech synthesis technologies. On top of that, their dataset was then broken up to three separate datasets, where one of them was "for-norm", one was "for-2sec", and the last one being "for-rerec". After the dataset was broken up, the first step to their process was to preprocess the data so the data is in a

format where the machine can understand the input, which is called data framing. Furthermore, normalizing audio features, extracting audio sampling, and setting the sampling rate was done within this step. The reason why their research did feature extractions were to get the characteristics and features from the audio signal to pass into the machine to obtain better results. Once finished, out of the 270 features that were originally extracted in their research, it got reduced down to 65 features as these 65 were "crucial enough" for their model to learn from. After testing and evaluating, a test accuracy of 93% was achieved.

Design and Methods:

This senior project was inspired by Anagha, et.al, as the current research this quarter aims to recreate their results following their methodology. To begin with their methodology, Anagha and their team utilized a dataset called ASVspoof 2019. Within this dataset, it consisted of both real and fake audio recordings. Before anything could be done with the dataset, it was essential for preprocessing and labelling to be done so the model can understand which audio files are "bonafide" and which audio files are "spoof". To accomplish this, each audio file was either assigned a 0 ("spoof") or a 1 ("bonafide"). To understand the characteristics of real audio files vs fake audio files, melspectograms were utilized rather than using raw data, and this was done by using Python and a Python library called Librosa, where audio files can be loaded in and Mel-spectrograms can then be produced. The purpose of using Mel-spectrograms is to provide a visual representation of the audio signal frequency. Mel-spectrograms are also used for feature extraction as Mel-spectrograms are effective in identifying important details. Overall, Mel-spectrograms does a good job in identifying characteristics and components of the audio that the human ears are more sensitive to. To recognize what these different characteristics may look like, refer to Figure 1 to see an example of a melspectrogram for a "real" Physical Access audio file, Figure 2 to see an example of a melspectrogram for a "fake" Physical Access audio file, Figure 3 to see an example of a melspectrogram for a "real" Logical Access audio file, and Figure 4 to see an example of a melspectrogram for a "fake" Logical Access audio file. For the model used, a convolutional neural network (CNN) was used. This model design was chosen because this model can take in an input of a melspectrogram and its shape and many convolutional layers along with a variety of filters were able to be applied to help recognize patterns and features. In terms of what's passed into the model to help the model learn, two datasets were created with the training dataset using 80% of the data while the validation dataset was using the remaining 20% of the data. A training process called the Adam optimizer along with a categorical cross-entropy loss was used in their research and model to adjust the learning rates. Anagha and their team also applied a Dropout to prevent overfitting from occurring. Once this was completed, the model takes in the training data and validation data to learn and validate from. Then once that is done, to evaluate their models' performance, Anagha and their

colleagues used different metrics to look at their model's performance on the separate and unseen testing data, which also went under the same preprocessing steps as the training and validation dataset. This would conclude how Anagha and their team completed their methodology.

While inspiration was drawn from their research with the goal of attempting to recreate their results, changes and modifications were made throughout the current research's and senior project's methodology process, which will in turn provide different results, which will be evaluated later on. In this project, one change that was made that wasn't mentioned in Anagha and their colleagues' preprocessing was eliminating any audio files with no mapping to either "bonafide" or "spoof". With no mapping associated to these type of files, it will mess up the model as it can't learn from these audio files as they wouldn't understand if it's real or fake, so we exclude these files in our preprocessing step. Another change made in this project compared to theirs is rather than splitting up the training dataset to contain 80% of the data while the validation dataset containing 20% of the data, this research just used all the validation audio files in the validation folder and all the training audio files in the training folder that was provided in the ASVspoof 2019 dataset. Another change was rather than using a categorical cross-entropy loss like Anagha and their colleagues used in their model, this model will use a binary cross-entropy loss instead as there are only two possible classes to consider, which are "spoof" or "bonafide". One last change made in this senior project was rather than having one model like Anagha and their team had, this program had two models instead. The reason behind two models was due to the dataset having two separate folders of audio files. One folder contained Logical Access audio files, which involved speech synthesis and voice conversion attacks. The other folder contained Physical Access audio files, which involved replay attacks. The reason behind creating two separate models where one model is for the Logical Access while the other model is for the Physical Access is that the models would be more accurate at identifying their respective audio files as the characteristics to identify deepfake audio within Logical Access audio files could drastically differ the characteristics to identify deepfake audio within the Physical Access audio files. Other than those changes, the methodology for this project will similarly follow after Anagha and their team's methodology.

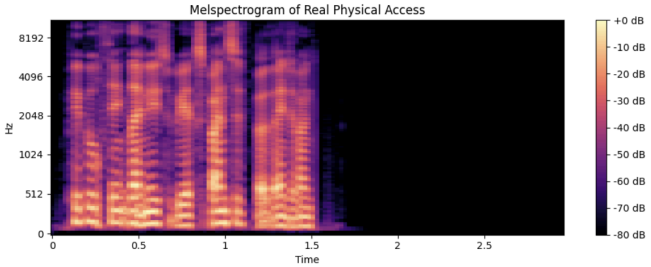

Figure 1: Melspectrogram of a Real Physical Access Audio File

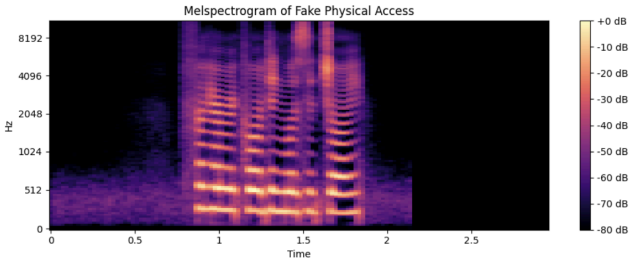

Figure 2: Melspectrogram of a Fake Physical Access Audio File

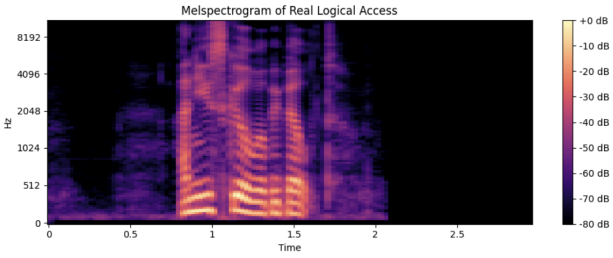

Figure 3: Melspectrogram of a Real Logical Access Audio File

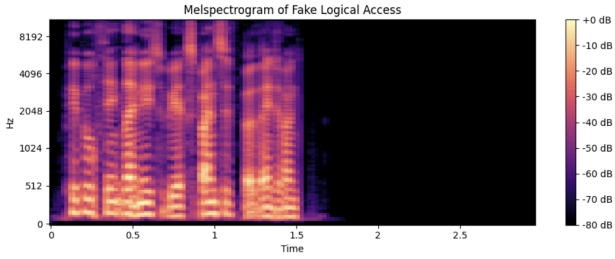

Figure 4: Melspectrogram of a Fake Logical Access Audio File

Once the methodology that was previously stated was accomplished in the previous quarter, the results from those designs could be examined and analyzed. It can be seen that the Logical Access model performed well with high accuracy and low loss. However, for the Physical Access model, while it had high accuracy, it also had high loss. So with this in mind, improving upon the Physical Access model was the main goal as finding a way to lower its high loss was the primary focus. To accomplish this, finding out reasons on what could be contributing to high loss was critical. The conclusions that were thought of was that the model was either underfitting (which was unlikely) as the training loss was sometimes high along with a high validation loss or overfitting (which was more likely) due to the fact the training loss was usually relatively low as the program was run while the validation loss was usually relatively high. This meant that if the model was overfitting, the model could've "memorized" the training data rather than truly understanding the Mel spectrogram's patterns and features, resulting in the model potentially being too complex and having too many parameters. However, if the model was underfitting, then the model potentially wasn't complex enough where it didn't truly understand the patterns of the Mel spectrograms, making it hard to apply those feature extractions to the validation dataset. Either of these reasons would lead to the model not performing well to new and unseen data, thus resulting in a high validation loss. To investigate this issue and try to better the model, it was essential to try out different hyperparameters for our model as well as different parameters when fitting our model. It was also important to try out different combinations of these changes to see what could produce the best results. For the hyperparameter changes, an attempt was made to adjust the number of filters in the Conv2D layers, the dropout layer, pool size, kernel size, and number of units in the Dense layer. Adjusting these specific hyperparameters was important because these hyperparameters are critical and essential to the model and the reasons why the model may potentially not understand the patterns if the model was underfitting or potentially understand the patterns of the training data too well if the model was overfitting. It's also important to find the

right balance for the parameters chosen when fitting the model, such as the number of epochs and batch size. With not finding the right balance of number of epochs and batch size, it could result in underfitting and overfitting as well. For the batch size, finding the right number is important because updating the parameters too much during training or not updating the parameters enough during training can lead to underfitting or overfitting. Another example is by not having enough epochs, it may not give the model enough opportunities to learn the patterns of the data while having too many epochs can give the model too many chances to "memorize" the training data. This is why finding the right combination of hyperparameters of the model and parameters when fitting the model is important with the goal of lowering the loss in mind.

Results:

To summarize the results of Anagha and their colleagues' model represented in Figure 5, the accuracy achieved in their model was 85% while the Area Under the ROC Curve was a .87 and the Average Precesion being a .90.

In comparison to the current study's results, shown in Figure 6, the machine learning models made to test and mainly focus on the accuracy and loss of both the Logical Access and Physical Access audio turned out with the Logical Access model having an accuracy of .896 and a loss of .239 while the accuracy of the Physical Access model produced a result of .865 and a loss being about 267789. To provide context for accuracy and loss within this project, accuracy represents the amount of mistakes made on the data while loss represents how big those errors were. For the Logical Access model, with the accuracy being relatively high and loss being relatively low, it means small errors were made on the small mistakes. For the Physical Access model, with the accuracy being relatively high and loss also being relatively high, it means big errors were made on the small mistakes. When comparing the Logical Access model to the Physical Access model, the Logical Access performed better than the Physical Access model. Moreover, in addition to Anagha and their team's model, all three models provided very similar accuracies.

Table I: Evaluation Metrics

Serial Num	Metric Nameā	Result
1	Accuracy	85%
2	Area Under the ROC Curve (AOC)	0.87
3	Average Precesion	0.90

Figure 5: The results showing the Accuracy, Area Under the ROC Curve, and Average Precision achieved in Anagha, et.al. [1] (2023) model.

	Logical Access Model	Physical Access Model
Accuracy	.896	.865
Loss	.239	267789

Figure 6: The results showing the Accuracy and Loss between the Logical Access Model and Physical Access Model in this project.

After seeing the loss being high for the Physical Access model from last quarter, an attempt was made to get results of a high accuracy and low loss for the Physical Access model this quarter. Many different combinations of hyperparameters, epochs,

and batch sizes were made and the loss seen varied a lot. Some of the hyperparameters that were attempted were having three Conv2D layers with 32, 64, and 128 filters respectively, utilizing dropout rates of .3, .4, .5, and .6, using different kernel sizes of (2,2) and (3,3), and much more. Some of the parameters used to try to get the best loss when fitting the model were having the number of epochs be 1, 3, 5, and 10, and some of the batch sizes used were 32, 64, and 128. From the various hyperparameters and parameters used along with these different combinations, some of these loss values seen included around 15, 348863, 4813, 17, and much more. It's also worth noting that for some of these losses, even though they may be relatively low, when run again to see if the model would produce consistent losses, it would sometimes produce high losses, making these models inconsistent. However, after many different combinations of hyperparameters, epochs, and batch sizes, a combination was found that was able to reduce the loss of this model. Not only does this model produce a low loss, but it produces the low loss pretty consistently. By changing the model's Conv2D layers to have 64 filters and 128 filters respectively, kernel size being (5,5), dropout rate being .7, number of epochs being 5 and batch size being 32, a high accuracy was able to be maintained with a .86 while the loss was decreased significantly to .34. For reference, the model from last quarter had filters of 32 and 64 respectively in the Conv2D layers, kernel size was (3,3), dropout rate was .5, number of epochs was 1 while the batch size was 128. Now with the new loss being significantly lowered, it means for the Physical Access model, the model now makes small errors on those small mistakes.

	PA Model 1	PA Model 2	PA Model 3	PA Model 4	PA Model 5	PA Model 6	PA Model 7
Accuracy	.86	.86	.86	.86	.86	.86	.86
Loss	17.89	4813.95	348863	.39	78005	15.71	.37

Figure 7: The results showing the Accuracy and Loss of some of the different Physical Access Models using different combination of hyperparameters and parameters

	Physical Access Model
Accuracy	.86
Loss	.34

Figure 8: The results showing the Accuracy and Loss of the new Physical Access model in this project

```
def create_cnn(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape = input_shape),
        tf.keras.layers.Conv2D(64, kernel_size = (5,5), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
        tf.keras.layers.Conv2D(128, kernel_size = (5,5), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation="relu"),
        tf.keras.layers.Dropout(.7),
        tf.keras.layers.Dense(1, activation="sigmoid")
    ])
    model.compile(optimizer="adam",
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    return model
```

Figure 9: The code of making the Physical Access model with the new hyperparameters, which produced high accuracy and low loss

```
def create_cnn(input_shape):
    model = tf.keras.Sequential([
        tf.keras.layers.Input(shape = input_shape),
        tf.keras.layers.Conv2D(32, kernel_size = (3,3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
        tf.keras.layers.Conv2D(64, kernel_size = (3,3), activation="relu"),
        tf.keras.layers.MaxPooling2D(pool_size = (2,2)),
        tf.keras.layers.Flatten(),
        tf.keras.layers.Dense(128, activation="relu"),
        tf.keras.layers.Dropout(.5),
        tf.keras.layers.Dense(1, activation="sigmoid")
    ])

    model.compile(optimizer='adam',
                  loss='binary_crossentropy',
                  metrics=['accuracy'])

    return model
```

Figure 10: The code of making the Physical Access model last quarter with the old hyperparameters, which produced high accuracy and high loss

```
x_train_training_pa = x_train_training_pa.reshape(-1, 128, 128, 1)
model1 = create_cnn(x_train_training_pa.shape[1:])
model1.fit(x_train_training_pa, y_train_training_pa, epochs=5, batch_size = 32, validation_data=(x_train_val_pa, y_train_val_pa))
loss_pa, accuracy_pa = model1.evaluate(x_train_testing_pa, y_train_testing_pa)
```

Figure 11: The code of fitting the new Physical Access model with new epoch and batch size parameters

```
x_train_training = x_train_training.reshape(-1, 128, 128, 1)
model = create_cnn(x_train_training.shape[1:])
model.fit(x_train_training, y_train_training, epochs=1, batch_size = 128, validation_data=(x_train_val, y_train_val))
loss, accuracy = model.evaluate(x_train_testing, y_train_testing)
```

Figure 12: The code of fitting the old Physical Access model with old epoch and batch size parameters

Discussion:

To go more in-depth about what the results mean in context of this project, with the Logical Access model having an accuracy of about .896 while the loss is about .239, the high accuracy paired with the low loss means the model didn't make many mistakes in terms of identifying whether an audio file was "bonafide" or "spoof". With the few mistakes that the model did make, the errors were small as the real value of the audio file compared to the value produced by the model weren't far off from each other. For the Physical Access model, the accuracy was about .865 while the loss was about 267789. This means that in this project, with a high accuracy along with a high loss, the model didn't make many mistakes in terms of identifying between a "bonafide" audio file and a "spoof" audio file. However, with the mistakes made by the model, the errors were very big, meaning that the real value of the audio file was very far off compared to the value predicted by the model. In terms of the objective and purpose of this project, which was attempting to have a model be able to accurately detect deepfake audio, both the Logical Access model and Physical Access model did a good job in differentiating between "bonafide" and "spoof" audio files with relatively high accuracy. The difference between the two models lies in how far off the models were when inaccurately determining the type of the audio file.

Now from the new Physical Access model created, we can now say with the .86 accuracy and .34 loss, this new model doesn't make many mistakes in determining whether or not an audio file was "bonafide" or "spoof", and with those few mistakes the model may make, the errors will be very small now. This now means both the Logical Access Model from last quarter and the Physical Access Model from this quarter both does a good job in identifying deepfake audio and differentiating between "bonafide" and "spoof" audio files and both models will be the same in terms of not being very far off from the real value when making an inaccurate decision.

As far as challenges faced throughout this process of lowering Physical Access model's loss, it was mainly just taking the time to figure out the right combination of hyperparameters and parameters along with understanding the nuances of the model and being patient with the process.

Conclusion:

Now knowing what the results look like for both the Logical Access model and Physical Access model, both models did well in terms of being able to differentiate between "bonafide" and "spoof" audio files. As far as what future work entails, there can be a lot done to improve this process. For one, something that could be investigated is how to lower the loss being produced in the Physical Access model as right now, it is a very high value. Another thing that could be looked into is making just one model like how Anagha and their colleagues do. With this one model, it can take in either a Logical Access audio file or Physical Access audio file and still be able to differentiate between "bonafide" and "spoof" with high accuracy. Overall, from this project and

experience, everything is on the right track and going in the right direction in terms of accurately detecting deepfake audio and helping prevent this type of harm.

Looking back at last quarter, something that could've been investigated and improved upon was able to be accomplished this quarter. Being able to lower the loss produced by the Physical Access Model was fulfilling as well as being able to learn even more about machine learning, deepfake audio, and the nuances of these models. Being able to produce two good models in Logical Access model and Physical Access model within these two quarters along with seeing this progress is exciting to see from a personal standpoint as well as a societal standpoint as work is being made to reduce the harm of deepfake audio attacks.

References:

[1] Anagha, R., Arya, A., Narayan, V.H., Abhishek, S.G., & Anjali, T. (2023). Audio Deepfake Detection Using Deep Learning. 2023 12th International Conference on System Modeling & Advancement in Research Trends (SMART), 176-181.

[2] Hai, X., Liu, X., Tan, Y., & Zhou, Q. (2023). SiFDetectCracker: An adversarial attack against fake voice detection based on speaker-irrelative features. Proceedings of the 31st ACM International Conference on Multimedia. https://doi.org/10.1145/3581783.3613841

[3] Iqbal, F., Abbasi, A., Javed, A.R., Jalil, Z., & Al-Karaki, J.N. (2022). Deepfake Audio Detection Via Feature Engineering And Machine Learning. CIKM Workshops.