ANALYSING THE PREDICTIONS OF A CNN-BASED REPLAY SPOOFING DETECTION SYSTEM

Bhusan Chettri[1], *Saumitra Mishra*[1], *Bob L. Sturm*[1,2], *Emmanouil Benetos*[1]

[1]School of EECS, Queen Mary University of London, United Kingdom
[2]School of EECS, KTH Royal Institute of Engineering, Stockholm, Sweden

ABSTRACT

Playing recorded speech samples of an enrolled speaker – "replay attack" – is a simple approach to bypass an automatic speaker verification (ASV) system. The vulnerability of ASV systems to such attacks has been acknowledged and studied, but there has been no research into what spoofing detection systems are actually learning to discriminate. In this paper, we analyse the local behaviour of a replay spoofing detection system based on convolutional neural networks (CNNs) adapted from a state-of-the-art CNN ($LCNN_{FFT}$) submitted at the ASVspoof 2017 challenge. We generate temporal and spectral explanations for predictions of the model using the SLIME algorithm. Our findings suggest that in most instances of spoofing the model is using information in the first 400 milliseconds of each audio instance to make the class prediction. Knowledge of the characteristics that spoofing detection systems are exploiting can help build less vulnerable ASV systems, other spoofing detection systems, as well as better evaluation databases[1].

Index Terms— Convolutional neural networks, automatic speaker verification, replay attack, spoofing detection, end-to-end learning.

1. INTRODUCTION

Automatic speaker verification (ASV) [1] systems are used for person authentication in various commercial applications such as call centers, banks, smart phones [2] etc. However, these systems are sensitive to spoofing attacks [3, 4]. The vulnerability of ASV systems against spoofing attacks is an important problem to solve because it poses a serious threat to the security of such systems. When successful, a spoofing attack can grant unauthorized access of private and sensitive data. Spoofing attack methods include generating artificial speech [5], impersonation or mimicry [6], and playing back speech recordings [3, 7]. To counter such spoofing attacks, one can build a system that discriminates between genuine and spoof speech signals; but what attributes should such a system use to make this discrimination? One can hope that statistical machine learning and an appropriate amount of data will be able to discover such attributes.

Several machine learning systems have shown success in spoofing attack detection in both the ASVspoof 2015 [8, 9] and ASVspoof 2017[2] public evaluation challenges. Particularly successful in detecting replay attacks are systems using deep neural networks (DNNs) [10, 11]. Although these systems have shown promising results, what they have actually learned to do has not been answered; they are often used as a black-box. Is a system that appears to detect a spoofing attack actually working with attributes relevant to the problem, or is it merely a product of how a train/test database was constructed [12]? For example, [13] demonstrates how a frame-based Gaussian Mixture Model (GMM) system trained for replay spoofing detection on the version 1.0 of the ASVspoof 2017 database exploited artefacts in the database to make class decisions. Further in [14], the same authors identify a similar issue for frame-based GMM systems on the updated version 2.0 corpus. Can we trust such a system "in the wild"? Answers to these questions can not only help improve the security of ASV systems, but also motivate new spoofing attacks, and improve training databases.

In this paper, we attempt to answer these questions for an utterance-based CNN adapted from a state-of-the-art CNN system ($LCNN_{FFT}$) [11], trained on the version 2.0 ASVspoof 2017 corpus [15]. There exist several methods to understand the global or local behaviour of DNNs/CNNs [16]. Here, we use the SLIME [17] algorithm to generate explanations for individual predictions. SLIME is based on the LIME algorithm [18], which is an acronym for Local Interpretable Model-Agnostic Explanations. The explanations from SLIME highlight temporal and spectral regions that the model weighs heavily to form its decisions for each class. Our findings show that a decision of a recording being "spoof" is weighted heavily by the information present in the first 400 milliseconds of the recording. It appears than that at least some of the attributes the model has learned come from peculiarities of the database, and not from the difference in channel characteristics one would expect in a replay attack. We demonstrate the significance of our analysis in two ways. We show how to manipulate misclassified spoof recordings to be judged as "spoof" by the model, thereby lowering its equal error rate (ERR); and we show how to manipulate spoof recordings such that the model judges them as "genuine", thereby dramatically raising the EER.

The rest of the paper is organised as follows. In the next section we provide our system description. Then in section 3 we introduce the SLIME algorithm that we use for generating local explanations for model prediction. We demonstrate the application of SLIME in section 4 for explaining model prediction at both the instance level (few confidently classified audio signals) and model-level (on the entire database). We further show the importance of our analysis through two intervention experiments in section 5. Finally, in section 6 we provide a discussion and conclude the paper.

2. SYSTEM DESCRIPTION

2.1. Database

The ASVspoof 2017 database was released as part of the second automatic speaker verification spoofing and countermeasures chal-

EB is supported by a RAEng Research Fellowship (RF/128). This research was supported by an NVIDIA GPU Grant.

[1]Code available online: https://github.com/BhusanChettri/SLT-2018

[2]http://www.asvspoof.org/

Table 1. Statistics of the ASVspoof 2017 version 2.0 corpus.

Subset	# Speaker	# Genuine	# Replay	Dur (hrs)
Train	10	1507	1507	2.22
Dev	8	760	950	1.44
Eval	24	1298	12008	11.94
Total	42	3565	14465	15.6

lenge that focused on text-dependent replay attack detection 'in the wild' with varying acoustic conditions [19]. As an update to fix the data anomalies found in the version 1.0 of the database [13], version 2.0 has been released online[3] by the organizers. Table 1 shows the statistics of the database. It is divided into three subsets: training, development and evaluation [15]. The training subset contains an equal number of genuine and replayed audio examples, 1507 each. The development subset has 760 genuine and 950 spoof examples. The evaluation subset has 12008 spoofed and 1298 genuine examples. Our work uses the version 2.0 ASVspoof 2017 database.

2.2. Model architecture and the input representation

We use the architecture adapted from the state-of-the-art convolutional neural network $LCNN_{FFT}$ [11] which has shown the lowest equal error rate (EER) on the evaluation subset of the ASVspoof 2017 spoofing detection challenge. As in $LCNN_{FFT}$, our CNN comprises of 5 convolution layers, 4 network-in-network layers, 5 max-pooling layers and 2 fully connected (FC) layers. We use ReLU activations instead of the max-feature-map (MFM) activations since our preliminary results show similar performance for both the activations. Thus we made a number of changes to achieve performance close to the state-of-the-art. We use 32 neurons in the first FC layer and a single neuron in the output layer in contrast to 64 neurons and two neurons used in $LCNN_{FFT}$. We add a batch normalization layer before the activation layer. Further, we reduce the number of kernels in each convolutional layer by a factor of 2 to keep the free parameters to a minimum.

The input to the network is a mean-variance normalized log power spectrogram of 4 seconds, similar to $LCNN_{FFT}$. Since the ASVspoof 2017 dataset uses 10 different phrases [15], the duration of audio files vary across these phrases. To obtain a consistent input representation we replicate[4] the audio samples if the duration is smaller or truncate the samples to 4 seconds duration. We use a 1728 point FFT, and a 108 ms window with a hop of 10 ms. Therefore, the input spectrogram has a dimension 865×400, where 865 is the number of frequency bins and 400 the number of time frames.

2.3. Model training and testing

We initialize the network weights using Xavier initialization [20]. We initialize all biases by zero. We train the network to optimize the binary cross entropy loss between a genuine and a spoof class. We use a learning rate of 1e-4, a batch size of 32 and a momentum of 0.9. We use the ADAM [21] optimizer with default parameters. We apply a dropout of 70% to the inputs of fully connected layers. We implement the CNN using the Keras [22] library. We use early stopping with the criterion: if the validation loss does not improve for 10 training epochs then we abort the training. We use a maximum of 100 training epochs and chose the best performing model on the validation data. At test time, for each audio spectrogram we use the model output (the posterior probability of being genuine) as the score and compute the equal error rate (EER) over the entire dataset using the Bosaris toolkit [23].

Using the above approach, we trained two models M1 and M2 on version 1.0 and 2.0 ASVspoof 2017 database, respectively.

[3]https://datashare.is.ed.ac.uk/handle/10283/3017

[4]We replicate samples in the time domain.

Table 2. Performance (EER%) on the ASVspoof 2017 corpus.

System	DB version	Train	Dev	Eval.
$LCNN_{FFT}$ [11]	1.0	-	4.53	7.34
M1	1.0	0.0	7.0	9.4
M2	2.0	0.0	7.6	10.6

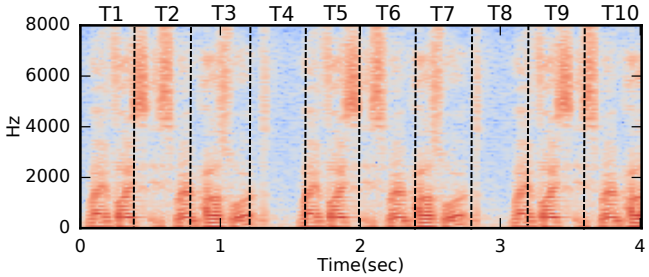

Fig. 1. Temporal segmentation of an input spectrogram ($\mathbf{x}_i$) into 10 uniform segments (Ti), each of duration 400 ms.

2.4. Results

Table 2 shows the performance of M1 and M2. Model M1 show comparable performance with $LCNN_{FFT}$ trained on the version 1.0 database. Further it should be noted that reproducing[5] the exact same results is difficult due to high dropouts and random weight initialization used during the training. Since our main objective in this paper is to understand what the CNN has learned about spoofing detection, we do not focus on minimizing the EER of $LCNN_{FFT}$. Further, we consider model M2 only for further analysis as it uses the updated version (2.0) database.

In the next section we introduce the SLIME [17] algorithm which we use to gain insights on what M2 has exploited from the underlying training data to make class decision.

3. SLIME ALGORITHM

SLIME is an algorithm to analyse the local behaviour of any (deep or shallow) machine listening model. SLIME is based on the LIME algorithm [18], which is an acronym for Local Interpretable Model-Agnostic Explanations. Ribeiro et al. [18] introduced the LIME algorithm and demonstrated its applicability to image recognition and text classification models.

SLIME extends LIME to machine listening systems by defining an *interpretable sequence* $\mathcal{X}_i$ for an input instance $\mathbf{x}_i$ (e.g., a time-frequency representation). An interpretable sequence is composed of elements, called interpretable components, that are in some way related to the classification of $\mathbf{x}_i$. SLIME defines three types of interpretable sequences (temporal, spectral, and time-frequency) depending on the way it segments $\mathbf{x}_i$ into interpretable components. For example, a temporal sequence $\mathcal{X}_i^t$ consists of temporal segments that

[5]Reported models (M1 and M2) are the best obtained out of five different runs of training.

Table 3. Temporal explanations of the most confidently correctly classified audio instances in the training (T), development (D) and evaluation (E) subsets. T1-T10 represent temporal segments in seconds. **T1**: 0-0.4, **T2**: 0.4-0.8, **T3**: 0.8-1.2, **T4**: 1.2-1.6, **T5**: 1.6-2.0, **T6**: 2.0-2.4, **T7**: 2.4-2.8, **T8**: 2.8-3.2, **T9**: 3.2-3.6, **T10**: 3.6-4.0. **G** and **S** denote the genuine and spoof classes.

			Weights assigned to different temporal segments									
Class	**File**	**Probability(%)**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**
G	T_1000780	87.8	0.009	-0.000	-0.002	0.007	-0.005	0.009	-0.003	**0.031**	**0.024**	-0.013
	D_1000260	91.2	0.007	-0.006	0.010	0.018	-0.012	0.005	0.001	**0.041**	**0.042**	-0.023
	E_1002535	88.4	-0.008	0.002	-0.000	0.006	-0.014	0.004	-0.000	**0.040**	**0.034**	-0.022
S	T_1002124	86	**0.335**	0.000	0.008	-0.000	0.008	-0.015	0.012	0.009	-0.005	**0.274**
	D_1001596	83.2	**0.507**	0.004	-0.013	-0.010	0.014	-0.007	0.007	0.006	-0.039	**0.119**
	E_1014008	81.0	**0.353**	-0.005	-0.015	0.010	0.009	-0.001	0.004	-0.018	-0.008	**0.207**

Table 4. Spectral explanations of the most confidently correctly classified audio instances in the training (T), development (D) and evaluation (E) subsets. F1-F10 represent spectral bands (segments) in Hz. **F1**: 0-813, **F2**: 813-1626, **F3**: 1626-2439, **F4**: 2439-3252, **F5**: 3252-4065, **F6**: 4065-4878, **F7**: 4878-5691, **F8**: 5691-6504, **F9**: 6504 to 7318, **F10**: 7318 to 8000.

			Weights assigned to different spectral segments									
Class	**File**	**Probability(%)**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
G	T_1000780	87.8	**0.019**	0.015	0.007	0.016	0.014	0.012	0.015	**0.019**	0.014	0.009
	D_1000260	91.2	0.018	0.018	0.020	0.013	0.017	**0.022**	**0.025**	0.011	0.020	0.013
	E_1002535	88.4	0.005	0.013	0.011	**0.016**	0.012	0.015	0.015	**0.018**	0.013	0.009
S	T_1002124	86	0.095	0.094	**0.108**	0.090	**0.105**	0.082	0.076	0.078	0.062	0.044
	D_1001596	83.2	0.094	**0.102**	0.091	0.100	0.091	0.093	**0.105**	0.097	0.080	0.055
	E_1014008	81.0	**0.136**	0.104	**0.145**	0.076	0.132	0.139	0.091	0.116	0.098	.077

SLIME generates by segmenting $\mathbf{x}_i$ (uniformly or non-uniformly) along the temporal dimension as shown in Fig. 1. SLIME maps an input instance $\mathbf{x}_i$ to its interpretable representation $\mathbf{x}_i^* \in \{0,1\}^{|\mathcal{X}_i|}$. In order to generate local explanation for the prediction $f(\mathbf{x}_i)$ where $f : \mathbb{R}^n \rightarrow \mathbb{R}$ is a classifier, SLIME first generates N artificial samples ($\mathbf{z}_i^*$) by perturbing the interpretable representation. SLIME perturbs $\mathbf{x}_i^*$ by randomly setting the interpretable components to zero. For example, for the instance in Fig. 1, if we set the temporal segments $T1$, $T4$ and $T7$ to zero, then a possible $\mathbf{z}_i^*$ is given as (0, 1, 1, 0, 1, 1, 0, 1, 1, 1). Later, SLIME maps each perturbed representation $\mathbf{z}_i^*$ to the feature space with an assumption that such a mapping exists. In other words, SLIME assumes that for each $\mathbf{z}_i^*$ there exists a corresponding $\mathbf{z}_i$ in the feature space. Finally, SLIME uses the perturbed representations $\mathbf{z}_i^*$ and their corresponding predictions $f(\mathbf{z}_i)$ to approximate f with a linear model g in the interpretable space $\tau = \{0,1\}^{|\mathcal{X}_i|}$. The explanation to the prediction $f(\mathbf{x}_i)$ is given by the weights $\mathbf{w}$ of the linear model $g(\mathbf{z}^*) = \mathbf{w}^T\mathbf{z}^*; \mathbf{z}^* \in \tau$. Formally, SLIME generates an explanation by the optimisation

$$\min_{g \in G} \; L(f, g, \rho_{\mathbf{x}_i}) + \Delta(g) \tag{1}$$

where L is a loss function (squared error between the original prediction $f(\mathbf{z}_i)$ and the model approximation $g(\mathbf{z}_i^*)$), $\rho_{\mathbf{x}_i}$ measures the distance between the input instance $\mathbf{x}_i$ and the generated sample $\mathbf{z}_i$, and $\Delta(g)$ measures the complexity of g (e.g., sparsity).

4. EXPLAINING THE PREDICTIONS USING SLIME

We use temporal and spectral interpretable sequences to generate explanations for a prediction of M2. For temporal analysis, we segment the input spectrogram into 10 temporal windows labeled T1-T10, each of 400 ms. For spectral analysis we segment the spectrogram into 10 spectral bands by grouping different frequency bins. Each of the nine bands labeled F1-F9 has a bandwidth of 813 Hz; F10 has a bandwidth of 683 Hz. We generate $N = 2000$ samples for producing explanations from SLIME.

4.1. Instance-level explanations

We take the six most confidently correctly classified genuine and spoof audio instances from the training, development and evaluation subsets and have SLIME generate explanations for their predictions.

Table 3 shows the weights assigned to each of the ten temporal segments for these six instances. The polarity of the learned weights signifies how the presence (or the absence) of a segment influences a model prediction. For example, the T8 and T5 segments in all the genuine instances in Table 3 are segments in favour of and against the prediction, respectively. The bold numbers in the table represent the top two weighing segments/components. We refer them as *top1* and *top2* explanations. For the genuine instances, SLIME assigns T8 and T9 as the top two explanations. We observe a marginal difference in the magnitude of weights assigned to T8 and T9 but a relatively larger difference for other temporal segments. These weights suggest that T8 and T9 offer more contribution towards genuine decisions. For the spoof instances, SLIME returns T1 and T10 as the top two explanations.

Table 4 shows the spectral explanations for genuine and spoof class prediction on the same six instances used in Table 3. There is not much difference in the magnitude of weights across the spectral segments. For both genuine and spoof decisions, it seems that M2 uses information across most of the spectral bands.

Using only the explanations for few confidently classified audio instances would not provide global understanding of the model behaviour. Therefore, in the next section we apply SLIME to every audio instance of the ASVspoof 2017 2.0 database and study the prediction explanation (weights) statistics to derive significant conclusions about what M2 has learned to make prediction.

4.2. Model-level explanations

Now we apply the SLIME algorithm across the entire training, development and evaluation subset. It is infeasible to show the distribution of weights across all the instances of the corpus, therefore we

Table 5. Distribution of temporal and spectral explanations generated from SLIME on the ASVspoof 2017 version 2.0 database. We take all the instances classified correctly with more than 70% confidence. Set **T**, **D** and **E** denote the training, development and evaluation subsets. **G**,**S** denote the genuine and spoof classes. T1-T10 and F1-F10 has the same meaning as in Table 3 and 4. The numbers represent the count statistics for the $top1$ explanations.

		What temporal information is the most critical?										*What spectral information is the most critical?*									
Set	**Class**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
T	**G**	13	12	60	215	0	9	2	132	**1047**	15	191	103	77	119	93	196	162	172	**285**	93
	S	**1208**	0	0	0	0	0	0	0	0	296	224	260	**410**	208	141	163	50	45	1	2
D	**G**	3	0	8	86	0	8	0	50	**521**	2	93	51	33	44	34	77	91	71	**122**	53
	S	**188**	0	0	0	0	0	0	0	0	167	60	44	**79**	30	23	44	21	37	15	2
E	**G**	49	13	13	116	0	9	0	98	**937**	30	100	93	96	124	82	126	170	190	**216**	52
	S	**1489**	0	0	0	0	0	0	0	0	**1668**	570	**647**	585	329	243	417	166	103	77	20

record only the *top1* explanation returned by SLIME and present the count statistics for *top1* explanations. This helps us derive a global understanding on the behaviour of M2 for genuine and spoof decision.

4.2.1. Explanations for the genuine decisions

In the training set, M2 correctly classifies 1505 out of 1507 genuine instances with more than 70% confidence. Table 5 (first row) shows the temporal and spectral explanations on these 1505 genuine instances. The temporal explanations suggest that though the majority vote for $top1$ appears to be T9, other temporal components also have some contribution in the prediction. To summarize, we observe that the first four (T1-T4) and the last three temporal segments (T8-T10) contribute most towards the genuine class decision.

One possible reason for such a spread in temporal explanations could be because of the 10 different variable length utterances used in the ASVspoof 2017 dataset. Further, inspecting the T9 segment of several genuine files in the training subset does not immediately reveal what M2 is detecting; however, we noticed that many files have non-speech (or silence) frames at their beginning. Therefore, M2 may be using both the speech and non-speech information across different temporal locations.

Looking at the spectral explanations (right hand side of Table 5) we find that M2 give importance to the information present across all frequencies (0-8 kHz).

To validate these findings, we repeat the above process across all the instances in the development and evaluation subsets. As shown in Table 5 (third and fifth rows), we observe a consistency in the genuine class explanations.

4.2.2. Explanations for the spoof decisions

In the training set, M2 correctly classifies 1504 out of 1507 spoof instances with more than 70% confidence. Table 5 (second row) shows the temporal and spectral explanations on these 1504 instances. The temporal explanation statistics suggest that the discriminative cue for replay spoofing detection appears either in the first or last 400 ms of the signal (T1 or T10 segments) and that M2 is not influenced by the information present in temporal regions T2-T9. On the spectral explanations, we observe that M2 is looking at the information present across all the frequencies. To validate these observations we repeat the process across all the instances in the development and evaluation subsets. As shown in the fourth and sixth rows of Table 5 we observe similar explanations.

Now the question is what cue is there in the first and last 400 ms of these instances? We inspect 50 spoof instances drawn randomly from the training subset and find that (1) the majority of instances do not have non-speech/silence in the first 400 ms (2) These instances have DTMF-like (dual-tone multi-frequency) tones with speech (29 out of 50) and without speech (7 out of 50) in the first 400 ms of the signal. The last 400 ms of the spoof instances have the same property found in (1) and (2). One reason for this could be due to raw samples copied for audio instances less than 4 seconds duration as pointed out in section 2.2.

4.2.3. Explanations for misclassification

We now analyse why a genuine test instance is misclassified as spoof and vice-versa. We hypothesize that a test utterance is misclassified if it does not exhibit its own class attribute but shows the attributes of the competing class. First, we look at why M2 misclassifies a genuine instance. We take all the genuine instances misclassified as spoof with more than 50% confidence[6] in the development and evaluation subsets and generate temporal and spectral explanations. We show the results in the first two rows of Table 6. We find that these genuine instances do not have the genuine class attributes, rather show the attributes of a spoof class ($top1$ corresponds to either T1 or T10 only), which explains the reason for misclassification.

Finally, we look at the spoof audio instances in both the development and evaluation subsets that were successful in fooling M2 with more than 50% confidence. We find 269 (out of 950) such instances in the development subset and 2088 (out of 12008) spoof examples in the evaluation subset. We show the explanations obtained from SLIME in the last two rows of Table 6. We find that these spoof audio instances do not show the attributes of the spoof class but appear to exhibit genuine class properties ($top1$ explanation distributed across T1-T10).

5. INTERVENTIONS

SLIME identifies where M2 is looking to discriminate genuine and spoof instances. We now perform two intervention experiments to test the significance of this analysis.

5.1. Intervention I: Break the system

Here, our primary goal is to break M2 from an attacker's perspective. In other words, we aim to increase the false alarm rate by manipulating correctly classified spoof instances so that M2 judges them as genuine.

[6]We chose confidence more than 50% as there are very few instances with more than 70% confidence in the development subsets. We use the same threshold across the evaluation subsets.

Table 6. Spectral and temporal explanations for all the **misclassified** genuine (**G**) and spoof (**S**) instances with more than 50% confidence in the development (**D**) and evaluation (**E**) subsets. F1-F8 has the same meaning as in Table 4 and T1-T10 as in Table 3.

		What temporal information is the most critical?										*What spectral information is the most critical?*									
Class	**Set**	**T1**	**T2**	**T3**	**T4**	**T5**	**T6**	**T7**	**T8**	**T9**	**T10**	**F1**	**F2**	**F3**	**F4**	**F5**	**F6**	**F7**	**F8**	**F9**	**F10**
G	**D**	4	0	0	0	0	0	0	0	0	3	1	2	1	2	0	0	0	1	0	0
	E	0	0	0	0	0	0	0	0	0	2	0	1	1	0	0	0	0	0	0	0
S	**D**	0	0	5	18	0	7	1	14	**224**	0	3	4	3	7	10	16	14	19	**21**	13
	E	62	34	28	221	0	23	2	172	**1540**	6	150	123	93	140	167	196	213	235	**311**	206

Table 7. Intervention I: breaking the system. Demonstrating the effect on two spoof instances each in the training, development and evaluation set.

Genuine class probability %			
Subset	**Instance id**	**before**	**after**
Train	$T\_1002189$	0.21	0.80
	$T\_1001687$	0.22	0.80
Dev	$D\_1000884$	0.21	0.85
	$D\_1000889$	0.25	0.83
Eval	$E\_1004999$	0.18	0.8
	$E\_1008476$	0.19	0.81

We randomly take two spoof audio instances from each of the training, development and evaluation subsets that have been classified correctly with more than 70% confidence and replace their first and last 400 ms (T1 and T10) by a T1[7] segment of the most confidently classified genuine signal in the training set ($T\_1000780$). We then submit them to M2 to see if they can pass as genuine. Table 7 shows that M2 now misclassifies them with high confidence. When we repeat this procedure for all the correctly detected spoof instances in the development and evaluation subsets, we observe a dramatic increase in the EER from 7.6% to 34.1% and from 10.6% to 29.7% respectively (first and second rows of Table 9).

Table 8. Demonstrating the effect of intervention II: protecting the system. Training subset has no misclassified instances.

Genuine class probability %			
Subset	**Instance id**	**before**	**after**
Dev	$D\_1001544$	0.81	0.23
	$D\_1000803$	0.78	0.4
Eval	$E\_1003144$	0.83	0.25
	$E\_1001926$	0.82	0.29

5.2. Intervention II: Protect the system

Here our goal is to protect M2 from a researcher's perspective (or an ASV system administrator). In other words, we aim to reduce the EER by manipulating misclassified spoof instances so that M2 judges them correctly. Since the training subset does not have any misclassified spoof instances (EER is 0.0%) we randomly take two spoof instances each from the development and evaluation subsets that were successful in fooling M2. From section 4.2.3, we know that M2 detects a spoof signal correctly if spoof attribute (DTMF tone and/or speech) appears in the first or last 400 ms (i.e T1 or T10 segment), we hypothesize that T1 and/or T10 of these four spoof instances do not have such attributes. We generate temporal explanations for these four instances and find that the $top1$ explanation does not favor T1 or T10.

[7]Main motivation here is to ensure that T1 and T10 (of the spoof instances) would not have any spoof attributes (section 4.2.2) after the intervention. Best option was to pick a genuine audio whose first 400 ms would contain mostly non-speech/silence. That is why we pick T1 of $T\_1000780$ (confident genuine instance) which satisfies the criteria.

Table 9. EER% before and after the two intervention on M2.

Intervention	Dev	Eval
Initial EER	7.6	10.6
I: Break the system	34.13	29.76
II: Protect the system	5.9	7.8

Now, we remove raw samples from the beginning of these four instances to ensure that the speech signal occurs within the first 400 milliseconds. We chose the amount of samples to remove based on the original duration of the audio signal. For example, if the duration is between 3 to 4 seconds we remove the first 1200 ms samples. We then submit them to M2 to see if they can be now detected as spoof. Table 8 shows that M2 now classifies them correctly as spoof with high confidence. When we repeat this process across all the misclassified spoof instances in the development and evaluation subsets, we observe a reduction in the EER from 7.6% to 5.9% and from 10.6% to 7.8% respectively (first and third rows of Table 9).

Though intervention II did not completely reduce the EER of M2 to 0% on the development and evaluation subsets, it shows the potential of our analysis work, and demonstrates how knowledge gained from such model explanations can help improve the detection performance. Upon closer analysis, we find that out of 269 misclassified spoof instances we intervened in the development subset, M2 detects only 8 instances as spoof with high confidence while large number of instances were detected spoof with low confidence. We find a similar observation on the evaluation subset. Out of 2088 misclassified spoof instances, only 88 instances were detected as spoof with high confidence. This explains the reason for a small change in the EER. Further investigation is required to gain deeper understanding about this discrepancy, which we leave as our future work.

6. DISCUSSION AND CONCLUSION

In this paper we implemented and analysed a CNN-based replay spoofing detection system (M2) adapted from the state-of-the-art CNN $LCNN_{FFT}$ using the updated version 2.0 ASVspoof 2017 database. System M2 shows comparable performance to $LCNN_{FFT}$. We use the SLIME algorithm to generate class explanations from spectral and temporal perspectives. Our analysis shows that M2 uses the first few milliseconds of the audio signal to make class prediction. We further demonstrated the significance of our analysis and findings by pre-processing the test signals which led to a predictable change in the EER on both the development and

evaluation subsets.

Though these systems, including the state-of-the-art $LCNN_{FFT}$, seem to be successful in discriminating between genuine and spoofed speech, our analysis shows that to some extent they could be exploiting cues from the database which are unrelated to the problem. This raises a question about the integrity and trustworthiness of such systems. Further, variability of patterns of signals (presence and absence of non-speech frames in the beginning) within a class makes the problem difficult on this database. For example not all spoof instances have a speech onset in the first few milliseconds of the audio signal and not all genuine instances have non-speech signals in the start.

Our analysis shows how spoofing detection performance is correlated to the first few samples of the audio signals. This suggests that a replay spoofing detection system built on this dataset needs to use a pre-processing step that detects non-speech samples before and after the speech onset/offset making sure that dataset artefacts are removed completely so that the models would then actually start to exploit factors of interest (acoustic environment, playback device, recording device properties etc.) from the underlying training data for replay spoofing detection.

Our future work aims to extend this analysis to investigate: (1) how explanations vary across the ten different phrases of the corpus, (2) how explanations vary across different types of replay conditions/configurations of the ASVspoof 2017 corpus, (3) whether speaker information (fundamental frequency, speaking styles) is used by M2 for making class decisions, and (4) different temporal and spectral segmentations (example, 40 temporal segments each of 100 ms) that may help us derive more deeper understanding of the explanations. Finally and most importantly, we would also look into methods to address the issues found in the ASVspoof version 2.0 database either through an automatic preprocessor or permanently cleaning the audio signals.

7. REFERENCES

[1] D. A. Reynolds, "Speaker identification and verification using gaussian mixture speaker models," *Speech Communication*, vol. 17, no. 1, pp. 91–108, 1995.

[2] K. A. Lee, B. Ma, and H. Li, "Speaker verification makes its debut in smartphone," *IEEE Signal Processing Society SLTC Newsletter*, 2013.

[3] Z. Wu, S. Gao, E. S. Cling, and H. Li, "A study on replay attack and anti-spoofing for text-dependent speaker verification," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2014 Asia-Pacific*, Dec 2014, pp. 1–5.

[4] Z. Wu, N. Evans, T. Kinnunen, J. Yamagishi, F. Alegre, and H. Li, "Spoofing and countermeasures for speaker verification: A survey," *Speech Communication*, vol. 66, pp. 130 – 153, 2015.

[5] Z. Wu, J. Yamagishi, T. Kinnunen, C. Hanili, M. Sahidullah, A. Sizov, N. Evans, M. Todisco, and H. Delgado, "ASVspoof: The Automatic Speaker Verification Spoofing and Countermeasures Challenge," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 588–604, June 2017.

[6] Y. W. Lau, M. Wagner, and D. Tran, "Vulnerability of speaker verification to voice mimicking," in *Proceedings of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing, 2004.*, Oct 2004, pp. 145–148.

[7] J. Gaka, M. Grzywacz, and R. Samborski, "Playback attack detection for text-dependent speaker verification over telephone channels," *Speech Communication*, vol. 67, pp. 143 – 153, 2015.

[8] J. Antonio, V. López, A. Miguel, A. Ortega, and E. Lleida, "Spoofing detection with DNN and one-class SVM for the ASVspoof 2015 challenge," in *Interspeech*, 2015.

[9] C. Zhang, C. Yu, and J. H. L. Hansen, "An investigation of deep-learning frameworks for speaker verification antispoofing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 11, no. 4, pp. 684–694, June 2017.

[10] P. Nagarsheth, E. Khoury, K. Patil, and M. Garland, "Replay Attack Detection using DNN for Channel Discrimination," in *Interspeech*, 2017, pp. 97–101.

[11] G. Lavrentyeva, S. Novoselov, E. Malykh, A. Kozlov, K. Oleg, and V. Shchemelinin, "Audio Replay Attack Detection with Deep Learning Frameworks," in *Interspeech*, 2017, pp. 82–86.

[12] B. L. Sturm, "A Simple Method to Determine if a Music Information Retrieval System is a "Horse"," *IEEE Transactions on Multimedia*, vol. 16, no. 6, pp. 1636–1644, Oct 2014.

[13] B. Chettri and B. L. Sturm, "A Deeper Look at Gaussian Mixture Model Based Anti-Spoofing Systems," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Apr. 2018.

[14] B. Chettri, B. L. Sturm, and E. Benetos, "Analysing replay spoofing countermeasure performance under different conditions," in *2018 IEEE International Workshop on Machine Learning for Signal Processing (MLSP)*, September 2018, accepted.

[15] H. Delgado, M. Todisco, Md. Sahidullah, N. Evans, T. Kinnunen, K.A Lee, and J. Yamagishi, "ASVspoof 2017 Version 2.0: meta-data analysis and baseline enhancements," in *Speaker Odyssey*, 2018.

[16] G. Montavon, W. Samek, and K.-R. Müller, "Methods for Interpreting and Understanding Deep Neural Networks," *Digital Signal Processing*, vol. 73, no. Supplement C, pp. 1–15, 2018.

[17] S. Mishra, B. L. Sturm, and S. Dixon, "Local Interpretable Model-Agnostic Explanations for Music Content Analysis," in *International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[18] M. T. Ribeiro, S. Singh, and C. Guestrin, "Why Should I Trust You?": Explaining the Predictions of Any Classifier," in *Proc. Knowledge Discovery and Data Mining(KDD)*, 2016.

[19] T. Kinnunen, Md. Sahidullah, H. Delgado, M. Todisco, N. Evans, J. Yamagishi, and K. A. Lee, "The ASVspoof 2017 Challenge: Assessing the Limits of Replay Spoofing Attack Detection," in *Interspeech*, 2017, pp. 2–6.

[20] X. Glorot and Y. Bengio, "Understanding the difculty of training networks," in *13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010, vol. 9, pp. 249–256.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[22] François Chollet et al., "Keras," https://keras.io, 2015.

[23] N. Brümmer and E. D. Villiers, "The Bosaris Toolkit: Theory, Algorithms and code for surviving the New DCF," *arXiv preprint arXiv:1304.2865*, 2013.