Voice Cloning Detection: Deep4SNet Counter to SiF-DeepVC

Thomas Snipes* and Shuang Lin†

*Computer Science & Engineering
Lehigh University
Bethlehem, PA, USA
tps323@lehigh.edu

†Cognitive Science
Lehigh University
Bethlehem, PA, USA
shl224@lehigh.edu

***Abstract*—Recent advancements in voice cloning technology have raised concerns regarding the vulnerability of AI-synthesized voices to detection by state-of-the-art voice clone detectors. A notable study introduced SiF-DeepVC, a method employing hand-crafted noise above the human hearing frequency range to camouflage AI-synthesized voices, successfully bypassing leading voice clone detectors. In response, this paper presents a survey application utilizing the Deep4SNet model as a baseline against SiF-DeepVC in voice clone detection. Deep4SNet, originally trained on the H-Voice, is a convolutional neural network (CNN) approach and a recent model to be bypassed by SiF-DeepVC. Our study aims to counter the hand-crafted SiFs specifically under the Deep4SNet CNN model. Two contributions are made: firstly, exploring the countermeasures against camouflage using SiF-DeepVC trained data, and secondly, replicating the original Deep4SNet model architecture to create a generalized model capable of countering the camouflage across various datasets. The methodology involves data preparation, filtering, feature extraction, modeling, and metrics evaluation. The network architecture follows a shallow CNN design, leveraging dropout layers for regularization and sigmoid activation function for binary classification. Results indicate that models trained on SiF-DeepVC data perform better against the camouflage but lack generalizability. Notably, the proposed brown models, trained on a combination of H-Voice and SiF-DeepVC data, achieve high performance while maintaining generalizability. Challenges were met, including the absence of detailed architecture in the Deep4SNet source code and anomalies in testing metrics, attributed to IDE bugs. Overall, this study provides insights into the effectiveness of Deep4SNet-based models against SiF-DeepVC camouflage and highlights the importance of generalizability in voice clone detection systems.**

I. Introduction

A recent novel study implemented hand-crafted noise above the human-hearing frequency range to camouflage AI-synthesized voices [5]. Their method, SiF-DeepVC, successfully bypasses three state-of-the-art voice clone detector models: RawNet2 [6], Deep4SNet [3], and Farid et al [1]. During a standard voice cloning, the model generates speaker-irrelative features (SiFs) above the human-perceivable frequency range, and SiF-DeepVC aims to replace those SiFs [5].

A human voice recording has information concentrated in the range of 300 Hz to 3400 Hz, so SiF-DeepVC merges the frequencies above 4000 Hz in synthetic recordings with the noise above 4000 Hz from real recordings [5]. Their method reduces the volume of frequencies below 4000 Hz in the real recording, leaving mostly the SiFs. They also reduce the volume of frequencies above 4000 Hz in the generated voices, leaving mostly the voices. Then, they can merge the real SiFs into the fake voices, creating a camouflaged synthetic voice to confuse the detection models [5]. To prevent the camouflage and bypassing of recent voice detectors, we have decided to dive into Deep4SNet and aid in the defense against voice hijackers.

A. Our Contributions

Our project is a survey application utilizing the existing Deep4SNet as a baseline against SiF-DeepVC in voice clone detection. We aim to counter the hand-crafted SiFs, specifically under the Deep4SNet CNN model. Deep4SNet is a computer vision approach, and the newest model that was bypassed by SiF-DeepVC [3], [5]. The model was originally trained on the fake voice histograms provided by Imitation and DeepVoice (H-Voice) [2]. Our contributions to this process will be completed in two stages. First, we explore how easy it is to counter the camouflage if the model is simply trained on the camouflaged data. In the second stage, we aim to replicate the original Deep4SNet model and use that to create a generalized model that can also counter the camouflage.

B. Related Works

Similar to Deep4SNet, RawNet2 is a convolutional neural network that outputs speaker embeddings [6]. It is an improvement upon RawNet, a CNN introduced in 2019 that had some shortcomings. RawNet applies its initial convolutional layer directly to the raw speech waveform, with all filter parameters learned automatically. The architecture incorporates residual blocks to extract frame-level representations, leveraging skip connections for training deeper classifiers and capturing more discriminative information. These blocks utilize either long short-term memory or gated recurrent units to aggregate utterance-level representations. For verification tasks, RawNet employs either a b-vector classifier or a deep neural network backend with concatenation and multiplication techniques. However, the unconstrained nature of the first layer's parameters, learned automatically, may lead to slow learning and noisy outputs, particularly with sparse training data.

To solve these problems, SincNet, another CNN, is introduced. SincNet addresses challenges in neural network design by incorporating a unique first convolutional layer composed of band-pass filters parametrized as sinc functions, operating directly on raw waveforms. This constrained layer, with learnable cut-in and cut-off frequencies, fosters the learning of a more meaningful filterbank structure and outputs.

RawNet2, an evolution of RawNet1 introduced in 2020, merges the strengths of RawNet1 and SincNet. It adopts the first layer architecture of SincNet while retaining RawNet1's residual blocks and GRU layers. Additionally, RawNet2 introduces filter-wise feature map scaling via a sigmoid function, acting as an attention mechanism to derive more discriminative representations. It also increases the embedding dimension from 128 to 1024. Experimental results on various databases demonstrate the effectiveness of RawNet1 and RawNet2 in automatic speaker verification.

II. Methodology

In this project, we have explored several models that were affected by the SiF-DeepVC bypass. We focus on Deep4SNet and its histogram inputs, working mainly on the Google Colab IDE.

A. Deep4SNet

The backbone of our model follows the Deep4SNet CNN architecture [3]. The Deep4SNet model takes histograms, converted from real and synthesized speech audios, as its input. Deep4SNet is a convolutional neural network solution with image augmentation and dropout. The model's architecture is trained on 2092 histograms of original and fake recordings, and cross validated with 864 histograms. For Imitation-based recordings, Deep4SNet boasts a precision of 0.997 and recall of 0.997. For Deep Voice recordings, they had a precision of 0.985 and recall of 0.944. Overall the model has an incredible global accuracy of 0.985. The results seem to indicate that it is generally successful in detecting synthesized voice clones.

B. Histograms

The dataset used to train the original model is H-Voice [4], but we will be training our Deep4SNet model on the SiF-DeepVC data set [3]. H-Voice provided their histogram architectures and methods; plotting in Matlab with 2^{16} bins [2]. We format the SiF-DeepVC audio data into histograms following the H-Voice's methods. However, due to our computing limitations, we must resort to a lower number of bins. The number of bins are reduced from 2^{16} bins to 2^{8} bins.

C. Google Colab IDE

The integrated development environment (IDE) we have chosen is Google Colab. Unlike the incompatibility issues with other IDEs, Colab's TensorFlow version

Source	**Files**	**Label**
FoR	4500	Real
Farid et al.	2995	Fake
RawNet2	1505	Fake
Deep4SNet	1000	Fake

TABLE I: Audio Files From SiF-DeepVC

Source	**Files**	**Label**
Original	3268	Real
Imitation	3260	Fake
Deep Voice	144	Fake

TABLE II: Audio Files From H-Voice

is compatible with the original Deep4SNet model, TensorFlow version 2.15.0 and Keras version 2.15.0. We have attempted to work on Lehigh University's remote MAGIC GPU, but the newer Python and TensorFlow versions disallows loading in the original Deep4SNet. Therefore, Google Colab was the option we chose.

III. Approach

Our project is split into five major steps. First, Data Preparation, in which we download, load, and sample the data. Second, Filtering, where we filter spectrograms of the audio files to remove SiFs above the frequency of 4000 Hz. Third, Feature Extraction is the stage we transform the regular and filtered audios into histograms. Fourth, in Modeling, we create three variations of model architectures. One baseline Deep4SNet, one with increased dropouts, and one with deeper layers. In our fifth step, Metrics, we measured our model performances with methods based on the Deep4SNet paper [3] and the SiF-DeepVC paper [5], utilizing accuracy and false positive rate.

A. Step 1: Data Preparation

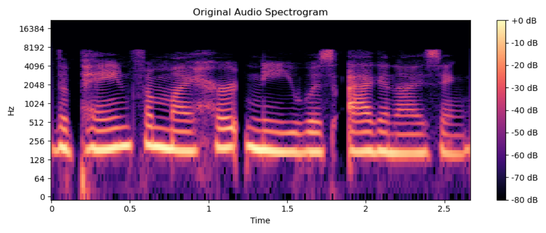
We use original human recordings from the Fake-or-Real (FoR) Validation dataset and cloned fake voices by SiF-DeepVC (SiF) for Farid et al., Deep4SNet, and RawNet2 [5]. All 24640 audio files, provided by SiF-DeepVC, are in .wav format and pre-labeled by their folder names. We sample 9000 files; 4500 fakes from Farid et al. and RawNet2 and 4500 reals from FoR. We also sampled 1000 files from their method against Deep4SNet to test our model's effectiveness against their camouflage. The SiF dataset is split into three sets following the general 70:30 rule: 70% training, 15% validation, and 15% test.

In order to emulate the original Deep4SNet model, we also implemented the H-Voice dataset, which was what their model trained on. The H-Voice

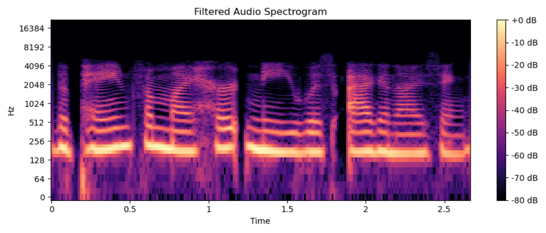
B. Step 2: Filtering

As mentioned in the introduction, SiF-DeepVC accounts for the fact that the human voice ranges from 300 - 3400 Hz by silencing anything below 4000

Hz, so we decided to implement a filter in hopes of combating their approach. Our filter, applied before converting the .wav files to histograms, blocks out all frequencies above 4000 Hz. By doing so we ignore their handcrafted SiFs, so they cannot trick our model. Please note that this filtered data is treated as a completely different dataset that we use for training, validation, and testing independently.

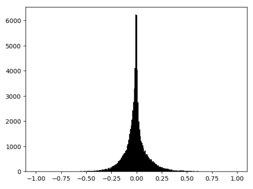


(a) Original Spectrogram

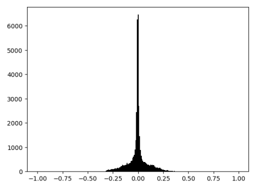


(b) Filtered Spectrogram

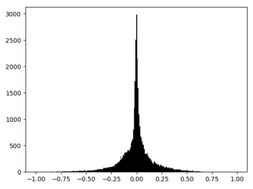
Fig. 1: Spectrograms Filtering



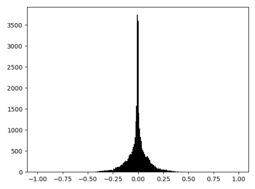
(a) Real from FoR



(b) Fake targeting Farid et al.



(c) Real from FoR



(d) Fake targeting RawNet2

Fig. 2: Histograms of SiF-DeepVC Voices

C. *Step 3: Feature Extraction*

The histograms of H-Voice are created in 2^{16} bins, but we cannot afford such expensive computations [2]. Therefore, we resorted to 2^8 bins, reducing the number of bins from 65536 to 256. Additionally, we colored our histograms black like the H-Voice histograms [4]. Ultimately, we reproduce our histograms as similar to the H-Voice set as possible.

D. *Step 4: Modeling*

We train our Deep4SNet model on the SiF-DeepVC's fake voice clones built against the other two methods, Fraid et al. and RawNet2, as well as real recordings from FoR. The fakes built against Deep4SNet were not trained upon as they will overfit the model. Instead, we use it to test the effectiveness of our counter model.

To be thorough, we decided to build four separate models based on Deep4SNet's architecture. One model nearly identical to Deep4SNet, one with multiple layers of dropout and one with another double the convolutional layers.

Deep4SNet reported better overall accuracy when combining the data pre-processing step of flipping the histograms and a single dropout layer of 0.2. Since their global accuracy of 0.985 seemed incredibly high and almost impossible, we decided to add multiple layers of increasing dropout to combat the potential issue of the model overfitting. While the model is not very deep compared to typical CNNs, increasing the dropout especially in the deeper layers is helpful since they are more prone to overfitting. Moreover, we thought increasing the dropout would generalize the model and make it more accurate when run on the test data.

Our other model variation is constructed with double the convolutional layers. Since all of our data is formatted as two-dimensional histograms, we do not have differently shaped or unique objects, there is not much of a need for additional convolutional layers in theory, but we wanted to see its performance on each dataset in case there were any surprising improvements or our theory of less accurate results was correct.

E. *Step 4: Metrics*

The original Deep4SNet paper measured their model performance with accuracy, precision, and recall. We have taken their influence and simply measured our models with its test accuracy.

Alongside the test accuracy, we have computed the false positive rate (FPR) for testing.

$$FPR = \frac{FP}{FP + TN}$$

FPR measures the ratio of falsely classified negative instances to the total actual negative instances. In other words, it represents the proportion of fake voices that are wrongly classified as real. FPR was also used in the SiF-DeepVC paper, so we can compare our model improvements further [5].

In terms of the target set, we measured its accuracy alongside with the true positive rate (TPR).

$$TPR = \frac{TP}{TP + FN}$$

TPR measures the ratio of correctly classified positive instances to the total actual positive instances. It simply represents the proportion of real voices that are correctly classified as real. We have chosen to use TPR for measuring the target set because FPR would provide 0% as the target set holds only fake voices.

IV. Network Architecture

Unlike other computer vision networks, the number of convolutional and pooling layers is significantly lower. A more shallow network is ideal since we are only using histograms and do not have to identify oddly shaped objects or deal with noisy images. The architecture consists of three convolutional and pooling layers followed by a flatten layer, hidden layer, and then the output layer. The equation below represents the output size of a convolutional block where *n* and *m* correspond to the number of rows and columns of an input image, respectfully, *f* represents the filter size, *p* is the size of the padding, *s* corresponds to the size of the stride, and n_f is the number of filters.

$$\left(\frac{\mathrm{n}+2\mathrm{p}-f}{\mathrm{s}}+1\right)\times\left(\frac{\mathrm{m}+2\mathrm{p}-f}{\mathrm{s}}+1\right)\times n_f$$

A dropout is included in the hidden layer to avoid overfitting.

Each convolutional layer contains a set of trainable parameters, comprising the weights of the filters and their associated bias terms. In the initial convolutional layer, there are 32 filters, each operating on RGB channels, resulting in a total of 896 weights, including the 3x3 weights for each filter and a single bias parameter per filter.

The chosen loss function, binary crossentropy, quantifies the disparity in entropy between two sequences of data—specifically, between the known labels and the predicted labels. This type of loss function is particularly effective for addressing binary classification tasks.

The chosen optimizer, RMSprop, is a method that dynamically adjusts the learning rate by scaling it with the square root of the average gradient of the mini-batch. ReLU (Rectified Linear Unit) activation functions are employed for the convolutional and hidden layers. ReLU is preferred for its balance between performance and computational efficiency, as it discards negative values while allowing positive values to pass through.

Lastly, the activation function for the last neuron is sigmoid. A popular and effective function for binary classification problems.

V. Results

In these three tables, the pink colored sections are H-Voice trained models that should perform poorly against the SiF-DeepVC data since the SiFs are built specifically against the Deep4SNet detection software. The green colored sections are SiF-DeepVC trained models that should perform well on both the regular and filtered data. Lastly, the brown colored sections are our generalized models that we created to see if their accuracy and FPR would hold up in a broader setting. They are made up of a combination of the H-Voice data and both the regular and filtered SiF-DeepVC data.

Each table shows results of the same models tested on different datasets. Table III is tested on the regular SiF-DeepVC dataset, while Table IV is tested on the filtered version of the SiF-DeepVC dataset. On the otherhand, Table V tests on the H-Voice data, which the original Deep4SNet model was trained on.

A. Pink

Looking at Table III, the original H-Voice model uploaded directly to our program performs exactly as expected, and our own H-Voice implementation almost mirrors it exactly. Our H-Voice test set accuracy is only 2.04% less than that of the original model, our false positive rates for the test set and true positive rates for the target set are identical to that of the original. In contrast, however, our H-Voice target accuracy is zero. This is a good sign because it means that the SiF-DeepVC target dataset, that was built specifically to trick Deep4SNet, was impossible for it to correctly identify a majority of the time. Now the two other models, the one with dropout layers and the one with additional convolutional layers performed mostly as expected. The deeper model was less accurate against the test set and the TPR was lower for the target dataset, but the target accuracy is baffling. Double the number of convolutional layers is a significant change in the architecture of the model, but it seems improbable that it would correctly identify all the custom made anti-Deep4SNet data.

In Table IV, we see how the performance was affected after filtering out the generated SiFs. These results are a bit odd and unexpected. The accuracy for four of the models, including the original H-Voice, is lower while the deeper model's accuracy increased by 2.15% in comparison to its accuracy in Table III. Continuing with unexpected results, the FPR of all of our models is incredibly high. Incorrectly identifying almost every real histogram, especially with the SiFs filtered out, is very strange. The target accuracies and TBRs remained relatively the same as the unfiltered SiF-DeepVC data, but the dropout model struggled. This seems to suggest that the dropout was more harmful than anything in this situation. In our attempt to prevent overfitting, we caused excessive regularization that resulted in the model underfitting the data. The model most likely struggled to capture important patterns and necessary features of the filtered histograms. This is surprising since filtering out the SiFs should have increased the performance of all of our H-Voice variation models in all categories.

Lastly, in Table V, our H-Voice shows itself to be as accurate as expected when testing against its own dataset. While the original H-Voice model does outperform in both categories, our models were respectable.

TABLE III: Test on Data - SiF-DeepVC - Regular

Model	Accuracy - Test	FPR - Test	Accuracy - Target	TPR - Target
Original-HVoice	53.78%	52.38%	9.00%	17.71%
Our-HVoice	50.74%	52.38%	0.00%	17.71%
Our-HVoice-Dropout	50.74%	52.38%	0.00%	15.55%
Our-HVoice-Deep	49.26%	52.38%	100.00%	15.55%
Our-SiF-Regular	93.85%	53.08%	94.00%	17.71%
Our-SiF-Regular-Dropout	93.48%	52.03%	94.80%	17.71%
Our-SiF-Regular-Deep	93.93%	48.87%	98.30%	17.71%
Our-SiF-Filtered	64.52%	52.38%	62.40%	15.55%
Our-HVoice_SiF-Regular	53.11%	52.38%	99.30%	15.55%
Our-HVoice_SiF-Filtered	56.00%	52.38%	99.00%	15.55%

TABLE IV: Test on Data - SiF-DeepVC - Filtered

Model	Accuracy - Test	FPR - Test	Accuracy - Target	TPR - Target
Original-HVoice	51.04%	49.81%	5.70%	21.50%
Our-HVoice	48.52%	99.86%	0.10%	21.50%
Our-HVoice-Dropout	48.59%	98.92%	0.00%	0.02%
Our-HVoice-Deep	51.41%	98.92%	100.00%	17.71%
Our-SiF-Regular	50.74%	90.06%	11.30%	0.02%
Our-SiF-Regular-Dropout	52.96%	93.66%	8.40%	0.02%
Our-SiF-Regular-Deep	50.15%	91.35%	14.00%	0.02%
Our-SiF-Filtered	94.44%	44.96%	98.30%	21.50%
Our-HVoice_SiF-Regular	67.33%	17.43%	92.60%	21.50%
Our-HVoice_SiF-Filtered	70.30%	23.20%	92.10%	21.50%

TABLE V: Test on Data - H-Voice

Model	Accuracy - Test	FPR - Test
Original-HVoice	97.61%	42.92%
Our-HVoice	54.07%	47.35%
Our-HVoice-Dropout	93.18%	48.89%
Our-HVoice-Deep	54.07%	98.92%
Our-SiF-Regular	56.71%	28.54%
Our-SiF-Regular-Dropout	55.38%	20.13%
Our-SiF-Regular-Deep	54.78%	0.66%
Our-SiF-Filtered	49.28%	73.23%
Our-HVoice_SiF-Regular	95.93%	45.35%
Our-HVoice_SiF-Filtered	98.68%	44.03%

Our H-Voice imitation had a low accuracy but decent FPR, the dropout model almost matched the original H-Voice in accuracy and FPR. Dropout did not have a negative effect like it did when testing against the SiF-DeepVC dataset. This can be most likely attributed to the fact that any model trained on certain data should perform well when testing against a set of the same data. The additional layers in the deeper model proved to be counter intuitive as we expected in this case. Since there are not many features to learn from a two-dimensional histogram, the model suffers from overfitting. The model is overly complex and when tested against unseen data, cannot generalize its knowledge.

B. Green

In the green sections of Table III, we can observe that the models trained on SiF-DeepVC dataset have increased accuracy as compared to models trained on H-Voice data. It also kept a relatively close FPR. When tested against the target sets, we see that they counter the camouflage well. However, the filtered models did not see as great of an improvement as the unfiltered models. Observing carefully into the Our-SiF-Regular variations, we see that applying more dropout does not increase performance. Meanwhile, modeling with deeper layers did increase performance slightly.

Observing Table IV, we see the green models performing worse when compared to Table III. However, the performance of the regular and filtered models seem to have switched. This means the models trained on specific datasets do not perform well under different scenarios, a low generalizability.

In Table V, we can see that the green models performed poorly as compared to the pink models trained on H-Voice. It makes sense since they were not designed to withstand a different scenario like the H-Voice data.

C. Brown

From Table III, we observe that the brown models did not perform well in the test set of SiF-DeepVC. However, it bypassed the camouflaged target set near perfectly. We also see a slight increase in accuracy in the filtered model, signifying the filter aided in bypassing the camouflage.

Table IV, the brown models performed best on the test set, as compared to the green and pink models. However, not as good as the Our-SiF-Filtered model since that one was designed to perform well in this situation. In the end, it bypassed the target set's camouflage well, but not as well as Table III.

In Table V, we see the brown models achieving high performance with the filtered variation surpassing the regular. This explains the lack of manipulated SiFs in the H-Voice dataset, because the filtered model analyzes the lower frequencies more.

VI. Discussion

It seems that by training the Deep4SNet model on other handcrafted SiF examples, it can prevent the effects of camouflage in handcrafted SiFs, even if it's crafted specifically to bypass the model itself. However, our model's low accuracy in H-Voice means that it is too specialized for camouflaged audio. From observing the H-Voice histograms, we notice that some of their audios are at a lower frequency. Sometimes, not even exceeding 200 Hz. This means our model is not used to dealing with lower frequencies, since SiFs are mostly in higher frequencies, which might have led to the lower accuracy when testing with H-Voice histograms. In the end, we found that the SiF and H-Voice models perform well in their own scenarios, but do not generalize well. Our goal to bypass the SiF-DeepVC's camouflage was a success because the brown models have maintained high performance while achieving generalizability.

A. Roadblocks

The Deep4SNet source code only provided a pre-trained model and not its detailed architecture. It can be assumed that SiF-DeepVC is directly tested on their pre-trained model, trained from H-Voice. Despite the lack of source code, the model was not difficult to build and the instructions were written clearly in the Deep4SNet paper [3].

In the testing stage to grasp accuracy, FPR, and TPR, we observed extreme values in some models. These extreme values were bugs in the IDE because they cannot be altered no matter how many times the testing was ran. We suggest overlooking these values as they may provide no value.

References

[1] Ehab A. AlBadawy, Siwei Lyu, and Hany Farid. Detecting ai-synthesized speech using bispectral analysis. In *CVPR Workshops*, 2019.

[2] Dora M. Ballesteros, Yohanna Rodriguez, and Diego Renza. A dataset of histograms of original and fake voice recordings (h-voice). *Data in Brief*, 29:105331, 2020.

[3] Dora M. Ballesteros, Yohanna Rodriguez-Ortega, Diego Renza, and Gonzalo Arce. Deep4snet: deep learning for fake speech classification. *Expert Systems with Applications*, 184:115465, 2021.

[4] Dora Maria Ballesteros L, Yohanna Patricia Rodriguez, and Diego Renza. H-voice: Fake voice histograms (imitation+deepvoice), 2020.

[5] Xin Liu, Yuan Tan, Xuan Hai, Qingchen Yu, and Qingguo Zhou. Hidden-in-wave: A novel idea to camouflage ai-synthesized voices based on speaker-irrelative features. In *2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE)*, pages 786–794, 2023.

[6] Hemlata Tak, Jose Patino, Massimiliano Todisco, Andreas Nautsch, Nicholas Evans, and Anthony Larcher. End-to-end anti-spoofing with rawnet2. In *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6369–6373, 2021.