Reproducibility report for "On Disentangling Spoof Trace for Generic Face Anti-Spoofing"

Anonymous Author(s)
Affiliation
Address
email

Reproducibility Summary

Scope of Reproducibility

From previous research(1), it has been demonstrated that face anti-spoofing problem can be treated as a denoising problem where spoof images can be disentangled into two parts – live counterpart and the noise. Based on that, this paper suggests that the disentanglement will be more appropriate if done in four parts - s, b, C, T where - s, b represent color range bias and balance bias, C denotes the smooth content patterns and, T is the high-level texture patterns. They claim that by leveraging these features, one can find the live counterpart of any image as well as the spoof counterpart of the live inputs by warping.

Methodology

We approached this challenge in the following manner. Firstly, we attempt to reproduce the paper in PyTorch while taking help from the paper, the authors, and the official implementation. This process took us around a month. Secondly, we verified different segments of the implementation (e.g., warping function) by using an input of the same value for both in our and their implementation. Thirdly, we train on MSU SiW Protocol-1 and OULU NPU Protocol-1 to verify whether we have succeeded to reproduce their results. Each epoch on a single V100 took around 12 hours. Later, we went on to propose several improvements over a few identified limitations of the original paper.

Results

We succeeded to match the ACER of the original paper to within 0.53% on both MSU SiW Protocol-1 and OULU NPU Protocol-1. In the original paper, the ACER of MSU-SiW Protocol is 0% and for us, it is 0.53% and the ACER of OULU NPU Protocol-1 is 1.9% and for us, it is 1.195%. Later, we moved our focus to generated images instead of the metrics due to a lack of computation resources. We, while following their implementation, fail to generate perfectly warped spoof images in many cases and detects unnecessary face details. We concluded that it's not always possible to get accurate spoof images. Later, we proposed a few techniques to

What was easy

As the paper was detailed and the official implementation was very easy to read and run, we were at ease while implementing the architecture of the paper in PyTorch, despite this being a complex one. Besides, we got help from the authors when we were perplexed, which made things easier.

What was difficult

The hyperparameters for producing the metrics are not given and said to be selected using the brute force method which made it really hard to reproduce.

Communication with original authors

After we read the paper and started implementing the code, we had a few confusions for which we mailed the author and got help from them. Later, we suggested an inconsistency between the paper and the implementation.

Submitted to ML Reproducibility Challenge 2020. Do not distribute.

The following section formatting is optional, you can also define sections as you deem fit.

1 Introduction

In recent years, the tendency to depend on biometric authentication is increasing. Of all bio-metric authentication technologies, face recognition is the most intuitive and effective. With the widespread use of face recognition brings into question the security of face-authentication systems and its robustness against various forms of malicious attacks. Sadly, these systems are susceptible to various forms of presentation attacks (PA), or more commonly known as spoofing. Mere printouts or recorded videos of faces presented in front of the biometric sensor are enough to fool many face-recognition systems.

In the past, hand-crafted features like HOG and LBP were used to tackle the problem of face anti-spoofing. In recent years, CNNs have been adopted as the preferred solution for this problem. CNNs, with techniques like depth supervision or pixel-wise binary supervision, has shown great potential. The paper(2) we reproduced proposes a Generative Adversarial Network based technique to find spoof traces from an image. This technique has shown great promise as the proposed method achieves state of the art (SOTA) performance in both inter and intra dataset testing across different research datasets.

2 Scope of reproducibility

Previously, it has been seen that, face anti spoofing problem can be treated as a denoising problem. Based on this finding, the paper we reproduced has extrapolated the following claims:

- Claims that the equation-1 will produce live counterpart of any image.
- Treats several cues as irrelevant and claims that it hinders the generalization.
- Claims that hard and rare(e.g., 3D face data) samples can be created using the spoof trace.

$$I_{hat} = I - (1 - s) * I - b - C - T \tag{1}$$

3 Methodology

Based on the claims from Section-2, we emphasize on the following methods to verify their claim and then go on to propose ways to tackle few problems that we found in their paper.

- Equation-1 was proposed to get live counterpart of any given input. Later on, this image is subtracted from the input image to get the final spoof trace. We follow the equation-1 and verify whether this claim is true or not.
- They claim that spoof trace helps to remove the outside cues which effect the generalization ability of the model. We use Fast Fourier Transformation based features as outside cues to verify the claim.
- Finally, while demystifying the above claims, we observed a limitation which is the reconstructed live counterpart of spoof images take up a different colored hue. We explored a few ways to solve this problem.

3.1 Experimental setup

3.1.1 Datasets

We used protocol-‘ from **MSU-SiW** and **OULU-NPU** to verify our claims. Below, the datasets are discussed:

OULU-NPU

OULU-NPU (3) is one of the most recent datasets in the face anti-spoofing dataset family. It consists of 55 subjects and all the recordings were done using six phones and in three different environments (session). The session was introduced to better simulate real world situations. To enforce generalization in the data, they followed "leave one out" approach where for training, five of six mobile phones are taken and the remaining one is used for testing. There were 3960 spoof videos (print and video replay combined) and 1980 bona fide videos. This makes it by far the most variant and definitely larger than MSU-MFSD (4) dataset. Also, there were 4 different protocols. The protocol names were set based on the level of difficulty. Following points show a brief description of OULU-NPU (3) protocols -

- Protocol I consists of a Test set which has different environments (session) than Dev and Train set.
- Protocol II ensures difference in instruments (smartphones) in the Test set as opposed to the Dev and Train set.
- Protocol III uses recording from different phones for Train and Test set. This helps to verify generalization capability of any network.
- Protocol IV is the hardest protocol of them all. It combines every constraints of the previous protocols. Also, with smaller subset of videos for training and evaluation.

MSU-SiW

The previous datasets before this (5) had lesser subjects compared to this one's 165 subjects where each subject has up to 8 bona-fide videos and 20 spoof videos. All the videos are captured in 30 frames per second on high definition Cameras – a Canon EOS T6 and a Logitech C920. The dataset provides print and replay attacks based on four sessions to capture the videos which included pose, illumination, distance and expression. Furthermore, to set a baseline for future studies, three protocols are created -

- **Protocol I** uses the first 60 frames of the videos for training and uses all the videos on the testing set. This would test if the model is generalized for contrasting face poses and expressions.
- **Protocol II** leverages the leave-one-out strategy to train on three mediums and test on the remaining one, to understand if the model is invariant to varying mediums.
- **Protocol III** tests the models performance on different forms of attack by training on one print attacks, testing on replay attacks and vice-versa.

3.1.2 Memory Usage

We had access to a single Tesla V100 and a single Tesla K80 for training. For verification purposes, we used the v100 as we required to train using batch size of 8 to be consistent with the original paper. For the separate extensions, we performed all our experiments on a single Tesla K80.

3.1.3 Implementation

We implemented most of our codes in **PyTorch**. However, there is a functionality which is not available in PyTorch, for that, we adapted a code snipped from TensorFlow to make the code faster in terms of that specific functionality. To do this, we converted the PyTorch tensor to a TensorFlow tensor for that block and later, converted it back to PyTorch tensor. Besides, to get the warping matrix of the spoof keypoint to live keypoint, we followed the official implementation. Furthermore, we compared several code blocks of ours with theirs whenever we had doubts. Our implementation: https://github.com/gazeai/STDN-PyTorch

3.1.4 Resources

We mostly followed the original paper and the github repository provided by the authors to reproduce our version. During the implementation phase, we found inconsistencies between the paper and the official implementation. These were resolved by contacting the authors. Besides, while implementing, we had support from PyTorch discuss forum.

4 Our Contribution

4.1 Fast Fourier Transformation as Cue

As per their paper, cues of spoof are too varied in real life scenario and spoof detection as a binary classification task would take those cues into account to make the final decision. But in reality, those cues might be irrelevant to spoof. For example illumination. This hinders the generalization ability of a model. Often the illumination and environment present in a research dataset contains much less variations than a real life scenario. To deal with this, they use only spoof trace to make the final decision. This has shown to prove good generalization ability as it doesn't take those spoof cues into account.

We however, tried incorporating Fast Fourier Transformation (FFT) into the final decision making process. In figure-1, we can see the results. It changes color tone to pink which is also visible in the main implementation of the paper[Figure-2]. However, it performs worse than the main implementation as we can see that, unlike the main implementation, the FFT implementation introduces a checkerboard texture in the generated images.

Figure 1: Generated outputs after using FFT as an outside cue. Row-1: a. Live input, b. Live counterpart of a, 3. Spoof counterpart of a. Row-2: a. Spoof input, b. Live counterpart of a, 3. Spoof trace of a

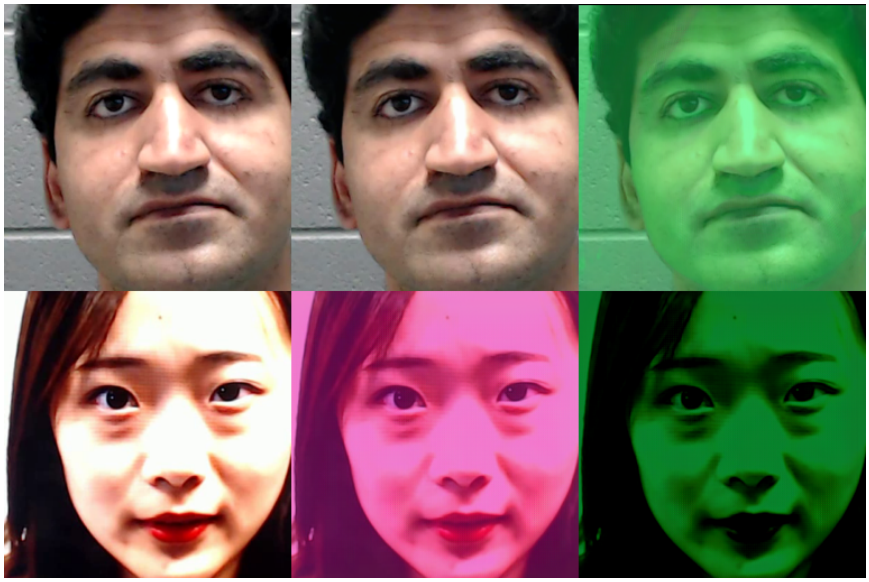

Figure 2: Generated outputs after the PyTorch Implementation of the official implemntation. Row-1: a. Live input, b. Live counterpart of a, 3. Spoof counterpart of a and, Row-2: a. Spoof input, b. Live counterpart of a, 3. Spoof trace of a

4.2 MinMax Normalization

In Equation-1 , I is the input image within the range of $[0, 1]$ and, *s, b, C, T* ranging within [-1, 1] according to the official implementation of the paper. If we think of a scenerio where s=b=C=T=-1, the reconstructed image I_{hat} will fall into the range of [4, 5] and on the other hand, if s=b=C=T=1, the range will be [-4, -3] which will produce unrealistic traces. This we can see in Figure-2 where the images add a pink and green hue in the faces and traces. To solve this problem, our initial solution was to normalize the reconstructed image I_{hat} using equation-2 which ranges from $[0, 1]$.

$$I_{hat} = \frac{I_{hat} - \min I_{hat}}{\max I_{hat}} \tag{2}$$

From Figure-3, we can see that the problem of changing hue has been solved, However, as we are normalizing the reconstructed images using minmax normalization, the reconstructed live counterpart of spoof inputs tend to distort as the model starts to converge.

Figure 3: Generated outputs after Minmax Normalization is used. Row-1: a. Live input, b. Live counterpart of a, 3. Spoof counterpart of a and, Row-2: a. Spoof input, b. Live counterpart of a, 3. Spoof trace of a

4.3 Hyperbolic tangent on Spoof-Trace

Although MinMax Normalization solves the problem of unnatural image range partially, another problem arises which is – if we normalize the reconstructed images within $[0, 1]$ range, we might fail to capture the traces where spoof images tend to be darker than live images. To solve this problem, we used equation-4 on the the trace we will get from equation-3. This ensures that the trace will always fall within the limit of $[-1, 1]$ and helps to solve the problem of less bright spoof images.

$$G(I) = I - I_{hat} \tag{3}$$

$$G(I)' = \frac{e^{G(I)} - e^{-G(I)}}{e^{G(I)} + e^{-G(I)}} \tag{4}$$

From Figure-4, we can see that this solution succeeds to find the spoof trace – which is a striped pattern – as well as maintains the color of the image as well. However, like all other above mentioned technique, this also fails to generate perfect outputs as the live counterpart has some jitter in it.

Figure 4: Generated outputs after hyperbolic tangent function is used in the spoof traces. Row-1: a. Live input, b. Live counterpart of a, 3. Spoof counterpart of a and Row-2: a. Spoof input, b. Live counterpart of a, 3. Spoof trace of a

5 Spoof Detection Results

From Table-1, it can be seen that, our implementation for MSU-SiW Protocol-1 matches their ACER close to 0.53%. On the other hand, our implementation on OULU NPU Protocol-1 outperforms their results by almost 0.3% which shows the robustness of our implementation.

	Official results			**Our implementation**		
Protocol	APCER	BPCER	ACER	APCER	BPCER	ACER
MSU-SiW protocol 1	0.0	0.0	0.0	1.0	0.05	0.53
OULU-NPU protocol 1	0.8	1.3	1.1	1.3	1.09	1.195

Table 1: Performance comparison between their and our implementation.

However, as discussed previously, our implementation based on their official repository demonstrates a discolored hue and fails to generate perfect spoof and live images.

6 Discussion

During the time of our study of the paper, we have observed many aspects of the paper. Some of these strengthen the paper and some of these are just mere scopes to improve the paper. In the following subsections, we will discuss them.

6.1 What worked

Firstly, we would like to discuss our observation of the original paper. We found the paper to match the implementation with the paper a great deal. The things that did not match are just the improvements that the authors are continuously working hard for.

We found the claim that – live inputs will produce blackened spoof trace and spoof input will will highlight the spoof textures – to be true. Also, the claim that outside cues hinder the generalization of face-anti-spoofing to be true as well. For this we took Fast Fourier Transformed features as an outside cue and the results were not as effective as the original one which we showed in Figure-1.

Secondly, in our extended work, we used $tanh function$ on spoof-traces to set the limit within $[-1, 1]$. This made the synthetic spoof images more realistic, images shown in Figure-4.

6.2 What did not work

Although, we managed to reproduce close to 0.53%, for the original paper, after a certain time of training, the reconstructed live and spoof images tend to be of more Pink and green hue. We believe, this problem existed because of the traces from equation-3 to not being within $[-1, 1]$ range.

Finally, the problem that we could not solve and seemed to be a big problem is that – whenever there is a side face in one of the inputs, the warping function doesn't work properly which is shown in Figure-5. It creates some form of stretching mark. Besides, whenever there are any extraneous object on the face, the spoof trace network considers them as a part of spoof trace. For instance, spectacles or any kind of birth mark in faces [Figure-6].

Figure 5: Warping doesn't work well when Sideface is used as an input. Row-1: a. Live input, b. Live counterpart of a, 3. Spoof counterpart of a and, Row-2: a. Spoof input, b. Live counterpart of a, 3. Spoof trace of a

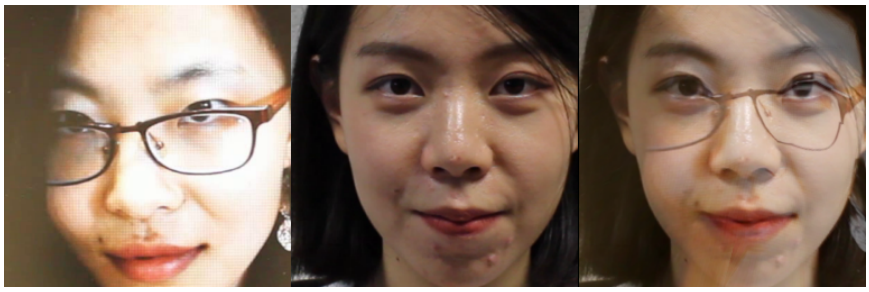

Figure 6: Spectacles are detected as a part of spoof trace. a. Spoof input, b. Live Input, 3. Spoof counterpart of live where spoof trace is from b.

6.3 What was easy

Despite the complicated technique, we found the paper to be understandable because of the diagrams. Moreover, we were able to match most of the techniques with code snippets from the official implementation. This helped us to get a clear perspective of what the author tried to say. Besides, when we found some mismatches – such as: six channel input in the code – we contacted the author and they helped us clear the confusion.

Besides, all the hyperparameters for training the technique were given which saved us from brute-forcing the training hyperparameters.

We advise anyone who wants to re-implement this paper is just to read the code and the paper concurrently and write things up and draw diagrams to make things easier.

6.4 What was difficult

To get the metrics, the paper expects us to depend on a lot of hyperparameters. However, these were not explicitly mentioned anywhere in the paper. We believe that this part of the paper can be highly improved to make it less hyperparamer-prone. However, as a temporary solution, we followed their suggestion and used brute-force method to get to the best solution.

Furthermore, the paper is an extremely complicated one to read, not because of its language, but because of the technique. The technique demands to have three training steps and each time with different types of requirements. It made things really difficult for us to re-implement it. To make sure of correctness, we divided the steps into small sections and renamed them in a way that we can identify these sections easily. Later, we verified these separate code snippets to ensure the reproduction. This way we managed to overcome this problem.

6.5 Future works

While the original paper introduced us to a briliant approach to tackle face-anti-spoofing, there are several scopes which can be addressed in future:

- The color and textures are not yet up to the mark as the color becomes less vibrant and the texture tends to show artifact effects.
- The warping for the side angle is a scope to work on.
- We can use a self attention so that we can ignore any specific marks on faces.

6.6 Communication with original authors

While reading the paper as well as the implementation, we had a few confusions for which we got help from the authors:

1. We couldn't connect the mention of using L2-regularizor in section 3.1 with the implementation. The authors helped us by pointing out to the line number of the code.
2. The authors continuously update the code base with improvements. We found two such inconsistencies we started implementing the code base. The author solved our query and later we integrated the improvements into our codebase.
3. We were unsure how early spoof regressor was working in the code base. The authors ensured us with this as well.

References

[1] A. Jourabloo, Y. Liu, and X. Liu, "Face de-spoofing: Anti-spoofing via noise modeling," in *Proceedings of the European Conference on Computer Vision (ECCV)*, pp. 290–306, 2018.

[2] Y. Liu, J. Stehouwer, and X. Liu, "On disentangling spoof traces for generic face anti-spoofing," in *In Proceeding of European Conference on Computer Vision*, (Virtual), August 2020.

[3] Z. Boulkenafet, J. Komulainen, L. Li, X. Feng, and A. Hadid, "Oulu-npu: A mobile face presentation attack database with real-world variations," in *2017 12th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2017)*, pp. 612–618, IEEE, 2017.

[4] Z. Zhang, J. Yan, S. Liu, Z. Lei, D. Yi, and S. Z. Li, "A face antispoofing database with diverse attacks," in *2012 5th IAPR international conference on Biometrics (ICB)*, pp. 26–31, IEEE, 2012.

[5] Y. Liu*, A. Jourabloo*, and X. Liu, "Learning deep models for face anti-spoofing: Binary or auxiliary supervision," in *In Proceeding of IEEE Computer Vision and Pattern Recognition*, (Salt Lake City, UT), June 2018.