Trends and Applications of Computer Vision: Face swapping at the limit - Related work write-up

Giovanni Lorenzini 223715 | Diego Planchenstainer 223728 | Riccardo Sassi 220237 | Riccardo Ziglio 224285

1. Introduction

The availability of new and powerful deep fake generating models could expose people to new threats, such as impersonation or other kinds of frauds. Thus, in order to protect the public from criminals, it is of vital importance to detect deepfakes. For this purpose, we want to assess the capability of generative models by applying them, in particular FSGAN [13, 14], to the challenges proposed by Perov *et al*. [16] pushing the systems to the limit.

2. Related works

The next sections address the most relevant related works, in particular GOTCHA, from which the challenges to test deepfake generator are taken. Since the objective is to push the networks to the limit for detecting flaws in the system, is essential that the general pipeline of deepfake generators is explained clearly. The main components are usually a face detection network, a face segmentation and swapping network, then the blending and color correcting network. These components will be inspected for each publication.

2.1. FSGAN

FSGAN is a deep learning-based approach to face swapping and face reenactment both in images and videos. The peculiarity of this approach is that it is subject agnostic. So, unlike other state of the art models, it can be applied to different pairs of faces without requiring specific training on those faces. Also, this model is end-to-end trainable and produces photo realistic, temporally coherent results [14]. In the following sections we will concentrate on the second version of FSGAN [14] since it improves the previous work [13].

This source is important for our topic since it is one of the Real Time DeepFakes (RTDF) generation pipelines tested in the reference paper about GOTCHA [16]. Moreover, our initial experiments, inspired by the human-introduced occlusion-like challenges of GOTCHA [16], were tested on the FSGAN pipeline using the Colab notebook made available by the authors on GitHub.

2.1.1 System overview

The objective of FSGAN is to synthesize a new image based on the target image, such that the face of the target is seamlessly replaced by the face of the source while preserving the same pose and expression [14]. The FSGAN pipeline, shown in Fig. 1, is formed of three major components: the reenactment generator G_r together with the segmentation network G_s, the face inpainting generator G_c and the blending generator G_b.

All generators are based on the archtiecture of pix2pixHD [22], while the segmentation network G_s is based on the U-Net architecture [19]. Each generator is trained with a loss function composed by a perceptual loss, to capture the fine details of the face texture, and a reconstruction loss to reconstruct the low-frequency image content. To further improve the realism of the synthesized image, the pix2pixHD adversarial objective is also applied to the loss function.

2.1.2 FSGAN pipeline

The FSGAN approach is composed by three main steps:

- Reenactment and Segmentation: in the first step, the reenactment generator G_r takes as input the source image I_s and a sequence of facial landmarks encoding the facial expression of the target image I_t. As output, G_r generates the reenacted image I_r such that the source face (F_s) has the same pose and expression as the target face (F_t). To align the source and target expression, the authors applied a face landmarks transformer network for interpolating between face landmarks without relying on 3D information. Instead, to align the source and target pose, an appearance map embedding the head pose of the source face was used to smoothly interpolate between head pose views.

  Once the pose and expression of F_s and F_t are aligned,

G_r computes the segmentation mask of the reenacted image S_r. Then, the segmentation net G_s, computes the segmentation mask of the face and hair of F_t that will be used in the next step.
- Inpainting: due to partial occlusions in the source face, i.e. due to hair, the reenacted image I_r may have some missing face parts that cannot be rendered on F_t. To avoid artifacts in the face swapping results, the face inpainting generator G_c is applied. G_s uses the S_t segmentation mask to inpaint the missing parts, producing the complete reenacted face F_c.
- Blending: the final phase of the pipeline is the blending of the complete face F_c with the target face F_t, obtaining the final face swapping result. The blending operation takes into account the different skin tones and lighting conditions thanks to the novel Poisson blending loss introduced in the paper.

Compared to other SoA deepfakes face swapping methods, FSGAN achieves better results while not being trained on any of the tested subjects.

2.2. DeepFaceLab

DeepFaceLab [17] is a deepfake generator that provides the tools necessary to perform photorealistic face swapping. With respect to FSGAN [14], which is a many-to-many technique, DeepFaceLab needs to be trained for any pair of source and destination faces, as it is a one-to-one technique.

The main characteristics of DeepFaceLab are:
- Convenience: a command line tool for using DeepFaceLab is provided, making as easy and as productive as possible the face swapping task.
- Extensibility: DeepFaceLab is composed by many modules that are interchangeable so that it's possible to select the best and most suitable ones.

In our opinion this source is important because it is ”the most sophisticated deepfake generator” [16], and therefore difficult to detect using automated systems like GOTCHA. Looking at the results obtained by GOTCHA, it is possible to see that DeepFaceLab is the most resilient to the proposed challenges. Also, the knowledge of the pipeline and its building blocks allow us to understand better how the attack surface is composed.

2.2.1 Pipeline

DeepFaceLab pipeline is composed as follows:
- Extraction (Fig. 2):
  - Face detection: from the images in the *src* and *dst* folder an algorithm to find the faces is applied. The default algorithm used by DeepFaceLab is S3FD [24].
  - Face alignment: to align the face a similarity transformation matrix is obtained using the method proposed by Umeyama [21]. To produce it the facial landmarks are needed, which are extracted either with 2DFAN [2] or PRNet [4].
  - Face segmentation: TernausNet [7] is used for segmenting the face, to remove items like hair, hands or glasses that may cover the face. If the result is not good enough, XSeg [17] can ben used. It needs a few manually segmented faces to train and then can automatically segment all others images.
- Training: the authors have provided two network structures: DF (Fig. 3) and LIAE (Fig. 4). Both of them can be trained without a stricly matched facial expression between *src* and *dst*, but LIAE is even more powerful by employing a more complex structure. This capability is given by the shared *inter* and *encoder*. The training process has some nice features, such as it can weight different parts of the face differently and that it uses a mixed loss to improve speed and clarity.
- Conversion (Fig. 5): the final part involves color transfer, blending and sharpening. Color transfer algorithms like RCT [8] and IDT [18] are used to match the skin color of the face with the skin color of the rest of the body. The face is blended using the Poisson blending technique [15]. At the end, a super-resolution neural network is used for sharpening the image.

2.3. GOTCHA

Previous research has conventionally studied deepfakes in the offline, non-interactive setting. These techniques rely on detecting unnatural artifacts in the image frequency domain [10], eye features [5], inner and outer face features [3], expressions [11] or biological signals [1]. For this reason, GOTCHA [12], a RTDF detection system for live video interactions, was built. GOTCHA presents to a suspected RTDF a sequence of challenges, with the objective of interfering with the deepfake pipeline, introducing visible artifacts and degradation.

2.3.1 Quality Metric

In order to assess degradation, we assume the availability of a quality metric, defined by the function $Q : \mathbb{R}^{H\times W\times C} \to [0, 1]$, where H, W and C are dimensions of a single video frame. Here, Q could either be a trained ML model or any other image statistic(s) that indicates “realism”; the higher the value assigned by the metric, the higher the realism quality of the frame.

2.3.2 Challenges and RTDF weaknesses

Challenges are performed either actively or passively, and are designed to exploit weaknesses of RTDF generators pipeline. Active challenges require the user to complete some physical tasks. Instead, passive challenges use some digital manipulations of the webcam feed. In Fig. 6 examples of the task are shown. Challenges are divided in five categories:

- Occlusion: any circumstance where the full frontal face is not visible due to the presence of an occluding object or head movement.
- Facial expression: introduction of facial deformations due to distinctive (but natural) facial expressions.
- Facial distortion: introduction of unnatural deformations of the face images that can be performed either actively by the subject or passively via video manipulation.
- Surroundings: challenges that alter the subject's background and surrounding ambience.
- Additional details: any extra information to the video feed that can be extracted and analyzed automatically through the webcam, like adding noise, steganography or feed duplication.

Since eyes are an essential feature for detectors, eye occlusion would be critical for **face detection**, **landmark detection** and **face alignment**, which are fundamental steps for deepfake generators. Moreover, the **face-swapping module** stores most of the source-specific knowledge. Challenges based on data diversity, or digital feed manipulation (e.g., inserting a mole, or illuminating with structured light), changing emotional expressions and distortions work very well. Also, occlusion and face distortion degrade the quality of the **segmentation module**. Other critical steps are managing smoothening, color difference, facial artifacts (e.g. freckles, beards), positioning, and scaling.

2.3.3 Workflow

In this section, all the steps of GOTCHA are described (Fig. 7). The authors consider ξ as the weighted cumulative anomaly score. With ξ_C, they mean the weighted cumulative score at task number C.

The steps are the following:

- **STEP 1**: Assign the score $\xi \leftarrow 0$ and select a threshold T.
- **STEP 2**: Select challenges C from $\mathfrak{C}_\beta$ and order them with an utility function.
- **STEP 3**: Ask the participant *par* to complete the next challenge $c_i \in C$. Initialize the timer.
- **STEP 4**: Capture the video I_{par,c_i} until timeout.Verify that the subject(s) performs the task correctly. If it fails, go back to **STEP 2**.
- **STEP 5**: Assign score I_{par,c_i} using an anomaly detector Q with confidence p_i.
- **STEP 6**: Increment ξ by $log(p_i)*$ Q($[I_{par,c_i}]$). if $\xi > T$, $par =$ **imp**. Declare **FAIL**.
- **STEP 7**: if $i < |C|$, return to **STEP 2**.
- **STEP 8**: if $i = |C|$ and $\xi < T$, $par =$ **src**. Declare **PASS**.

2.3.4 Challenge Verification

Once a challenge is issued, a human evaluator (or an ML-based module) can verify whether the participant indeed adhered to the challenge by observing the video response. E.g., if the challenge is to look side-to-side, then verification involves asserting a minimum change in yaw angle. It is important to note that challenge verification is specific to a given challenge and is independent of anomaly detection. Also, it does not immediately evaluate any particular artifacts.

2.3.5 Anomaly Detection and Defence Amplification

In this situation, two hypothesis are defined: H_0 (legitimate video) and H_1 (manipulated video). Now, it is possible to express the confidence p_i as:

$$p_i = \mathcal{L}(H_1|[I_{par,c_i}])$$

and the cumulative weighted score ξ of the video feed is:

$$\xi_k = \sum_{i=1}^{k} \log p_i * \overline{Q}([I_{par,c_i}])$$

Here, I_{par,c_i} is the response video, and $\overline{Q}$ is the average quality metric across all relevant frames in the response video.

2.3.6 GOTCHA components overview

Anomaly Detection: the assignment of anomaly score can be performed either via a human scorer or by an ML module. In the latter, a ML module was trained on 600 faces derived from FaceForensics dataset [20]. Two linear neural network heads (for classification and regression) were trained in a self-supervised way using binary cross-entropy and LPIPS [23] as loss functions, respectively. **RTDF Generation Pipelines**: the authors evaluated the effectiveness of GOTCHA against four modern RTDF generation pipelines: (i) a lightly trained DeepFaceLab model (LDFL), (ii) a highly trained Deep- FaceLab model (HDFL) [17], (iii) Face Swapping Generative Adversarial Network (FS-GAN) [14], and (iv) Latent Image Animator (LIA) [9]. The results of total score of these RTDF generation pipelines are shown in Fig. 8. Looking at the graphs, it seems LIA scores better than the other methods. Actually, this is due to the fact that, during the challenges, the pipeline of LIA is completely cleared and the video shows the real face of the impersonator. This could be cirtical in case there was only a ML-base module auditing the score.

2.4. Detection of real-time deepfakes in video conferencing with active probing and corneal reflection

Another interesting active approach is the one proposed by Hui *et al*. [6] where an image with high contrast, e.g. a black triangle on a white background, is shown on the screen. Since the eyes of a real person reflect the displayed image, the authors tried to search the pattern to classify whether the person in front of the camera is genuine or not. We think that it is important to report this approach since it proposes another challenge that could enhance GOTCHA.

2.4.1 Our results

We used a 1080p standard webcam to acquire the video and a 24" monitor to display the geometrical figure. From what we have observed, the reflection on the eye are very small and so in our opinion it cannot be used to discriminate if face swapping is performed. On the other hand, using a bigger monitor (32") and the rear camera of a smartphone (high resolution) the geometrical figure reflected on the retina can clearly be seen, but this is not the typical setup used in video conferencing.

References

[1] Giuseppe Boccignone, Sathya Bursic, Vittorio Cuculo, Alessandro D'Amelio, Giuliano Grossi, Raffaella Lanzarotti, and Sabrina Patania. Deepfakes have no heart: A simple rppg-based method to reveal fake videos. In *International Conference on Image Analysis and Processing*, pages 186–195. Springer, 2022. 2

[2] Adrian Bulat and Georgios Tzimiropoulos. How far are we from solving the 2d & 3d face alignment problem? (and a dataset of 230, 000 3d facial landmarks). *CoRR*, abs/1703.07332, 2017. 2

[3] Xiaoyi Dong, Jianmin Bao, Dongdong Chen, Ting Zhang, Weiming Zhang, Nenghai Yu, Dong Chen, Fang Wen, and Baining Guo. Protecting celebrities from deepfake with identity consistency transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9468–9478, 2022. 2

[4] Yao Feng, Fan Wu, Xiaohu Shao, Yanfeng Wang, and Xi Zhou. Joint 3d face reconstruction and dense alignment with position map regression network. *CoRR*, abs/1803.07835, 2018. 2

[5] Hui Guo, Shu Hu, Xin Wang, Ming-Ching Chang, and Siwei Lyu. Eyes tell all: Irregular pupil shapes reveal gan-generated faces. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2904–2908. IEEE, 2022. 2

[6] Hui Guo, Xin Wang, and Siwei Lyu. Detection of real-time deepfakes in video conferencing with active probing and corneal reflection. 4

[7] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation. *CoRR*, abs/1801.05746, 2018. 2

[8] Vladimir Iglovikov and Alexey Shvets. Ternausnet: U-net with VGG11 encoder pre-trained on imagenet for image segmentation. *CoRR*, abs/1801.05746, 2018. 2

[9] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping. *arXiv preprint arXiv:1912.13457*, 2019. 3

[10] Francesco Marra, Diego Gragnaniello, Luisa Verdoliva, and Giovanni Poggi. Do gans leave artificial fingerprints? In *2019 IEEE conference on multimedia information processing and retrieval (MIPR)*, pages 506–511. IEEE, 2019. 2

[11] Ghazal Mazaheri and Amit K Roy-Chowdhury. Detection and localization of facial expression manipulations. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1035–1045, 2022. 2

[12] Govind Mittal, Jiraphon Yenphraphai, Chinmay Hegde, and Nasir Memon. Gotcha: A challenge-response system for real-time deepfake detection. *arXiv preprint arXiv:2210.06186*, 2022. 2

[13] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGAN: Subject agnostic face swapping and reenactment. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 7184–7193, 2019. 1

[14] Yuval Nirkin, Yosi Keller, and Tal Hassner. FSGANv2: Improved subject agnostic face swapping and reenactment. In *IEEE Transactions on Pattern Analysis and Machine Intelligence*. IEEE, 2022. 1, 2, 3

[15] Patrick Pérez, Michel Gangnet, and Andrew Blake. Poisson image editing. *ACM Trans. Graph.*, 22(3):313–318, jul 2003. 2

[16] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. Deepfacelab: A simple, flexible and extensible face swapping framework. *CoRR*, abs/2005.05535, 2020. 1, 2

[17] Ivan Perov, Daiheng Gao, Nikolay Chervoniy, Kunlin Liu, Sugasa Marangonda, Chris Umé, Mr. Dpfks, Carl Shift Facenheim, Luis RP, Jian Jiang, Sheng Zhang, Pingyu Wu, Bo Zhou, and Weiming Zhang. Deepfacelab: A simple, flexible and extensible face swapping framework. *CoRR*, abs/2005.05535, 2020. 2, 3

[18] François Pitié, Anil C. Kokaram, and Rozenn Dahyot. Automated colour grading using colour distribution transfer. *Computer Vision and Image Understanding*, 107(1):123–137, 2007. Special issue on color image processing. 2

[19] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. *CoRR*, abs/1505.04597, 2015. 1

[20] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. Faceforensics++: Learning to detect manipulated facial images. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1–11, 2019. 3

[21] S. Umeyama. Least-squares estimation of transformation parameters between two point patterns. *IEEE Transactions on*

Pattern Analysis and Machine Intelligence, 13(4):376–380, 1991. 2

[22] Ting-Chun Wang, Ming-Yu Liu, Jun-Yan Zhu, Andrew Tao, Jan Kautz, and Bryan Catanzaro. High-resolution image synthesis and semantic manipulation with conditional gans. *CoRR*, abs/1711.11585, 2017. 1

[23] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 586–595, 2018. 3

[24] Shifeng Zhang, Xiangyu Zhu, Zhen Lei, Hailin Shi, Xiaobo Wang, and Stan Z. Li. S^3fd: Single shot scale-invariant face detector. *CoRR*, abs/1708.05237, 2017. 2

Appendix

The images cited in the report were placed here to respect the imposed page limit.

A.1. FSGAN

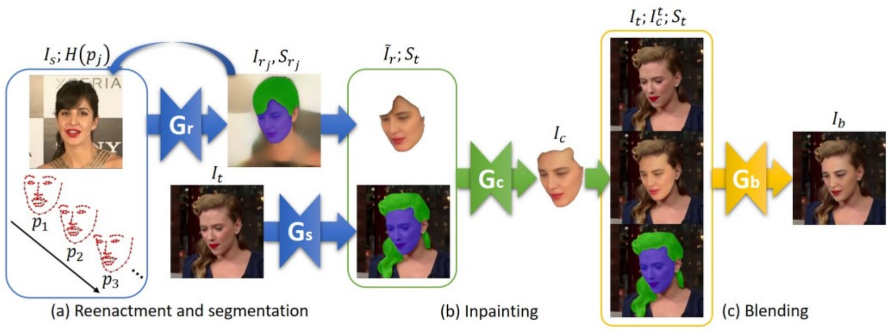

Figure 1. FSGAN pipeline; here can be noted the different operations performed by different generators.

A.2. DeepFaceLab

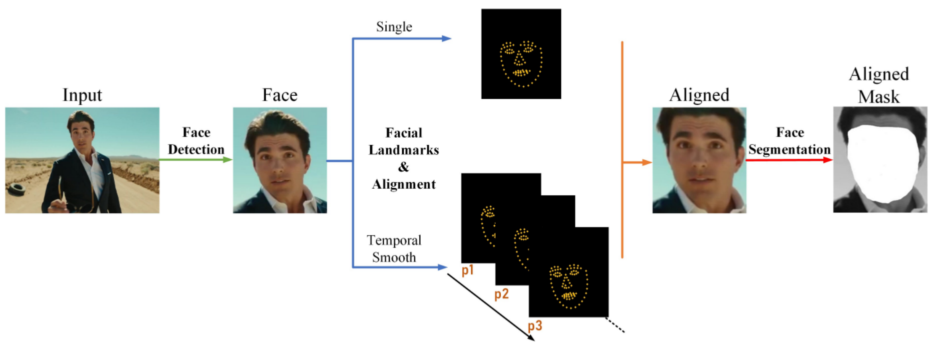

Figure 2. Extraction phase in DeepFaceLab.

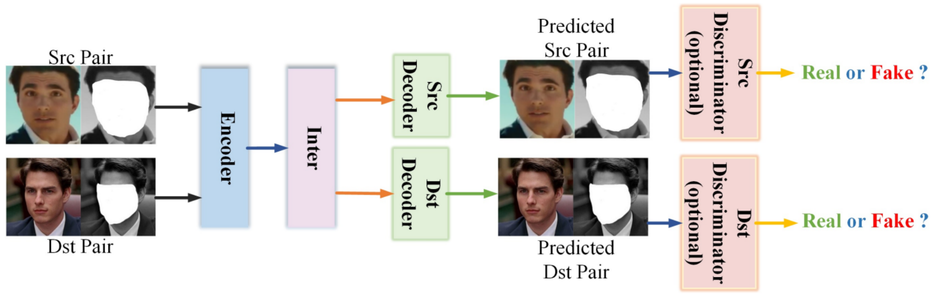

Figure 3. DF structure of DeepFaceLab.

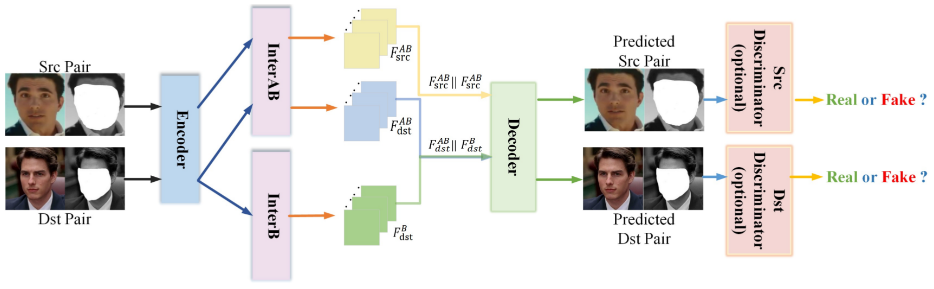

Figure 4. LIAE structure of DeepFaceLab.

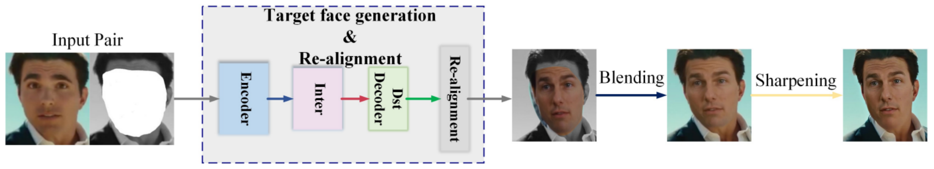

Figure 5. Conversion phase in DeepFaceLab.

A.3. GOTCHA

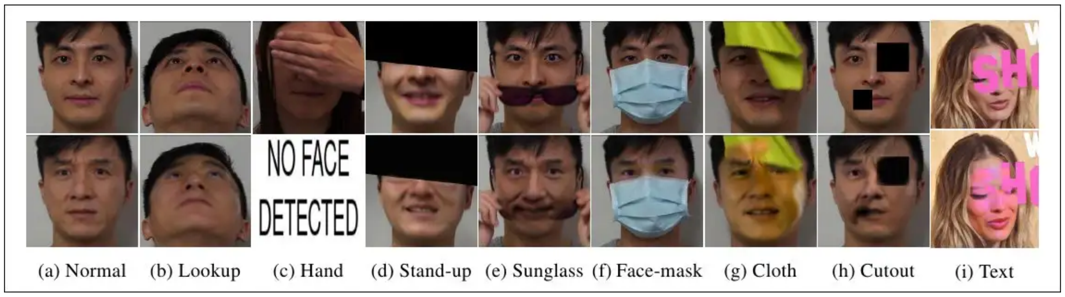

Figure 6. Example of active (b, c, d, e, f, g) and passive (h, i) challenges. The first row is the original video, while the second one is a RTDF. It is clear how these challenges can ruin the deepfake pipeline.

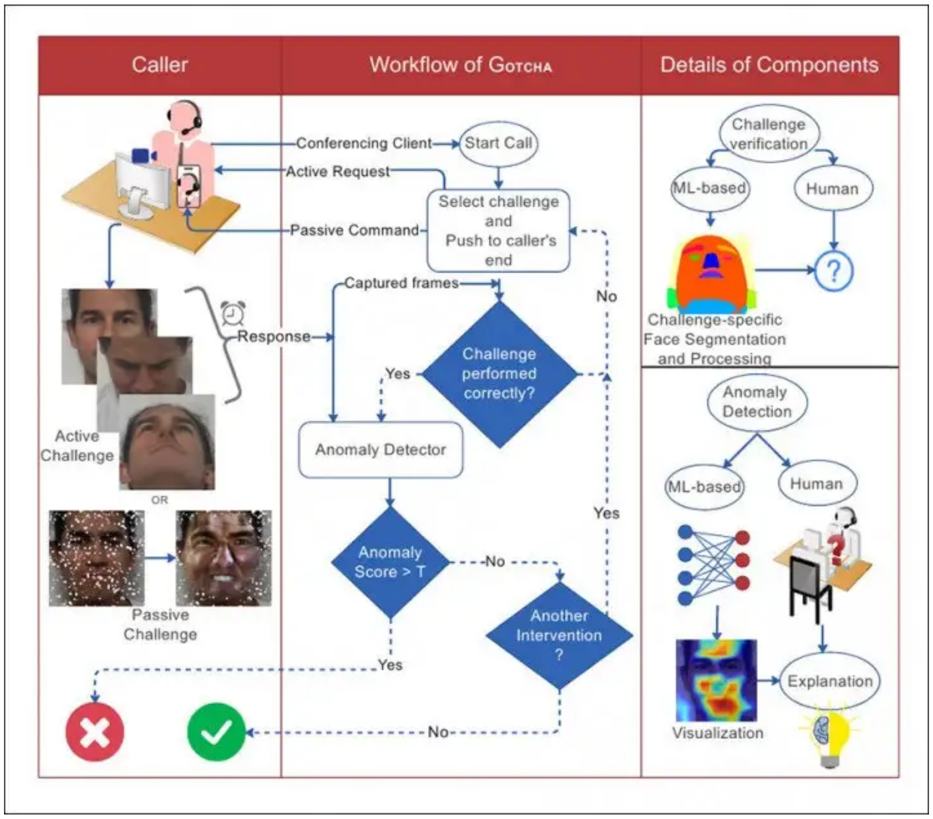

Figure 7. The figure illustrates the interaction of a caller with the GOTCHA pipeline. Bold lines in the middle column are data flows, and dotted lines are decision flows. Here, the active challenge is to look down and look up, while the passive challenge is adding snowflakes via a secure device (e.g., a smartphone). T is a threshold to limit false positives.

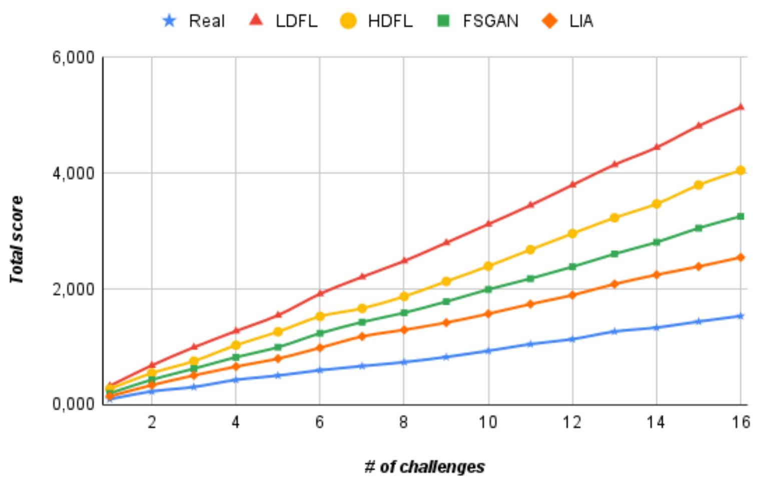

Figure 8. A plot of cumulative weighted score vs. cascade progression ($k \in [1, ..., |C|]$) for each pipeline. Genuine participants and impersonators walk through a cascade C consisting of 14 challenges, progressively increasing their cumulative weighted anomaly scores ξ. The manipulated feed results in a higher slope than the genuine one. Thus, challenges provide enough signal for the anomaly detector to classify between feeds and lead to a more robust decision as the cascade progresses.