Towards Social Video Verification to Combat Deepfakes via Deep Learning

by
Jacob Yu
James Tompkin, *Advisor*
Chen Sun, *Reader*

A Thesis submitted in partial fulfillment of the requirements for Honors
in the Department of Computer Science at Brown University

Providence, Rhode Island
May 2022

Acknowledgements

I would like to thank Professor James Tompkin for your patience, knowledge, and guidance throughout. It is your Computer Vision course that leads me onto this path.

This thesis would not have been possible either without my project partners, Yue Wang, Nick Huang, and Jayden Yi. We have come a long way and I have enjoyed every single moment.

I am also grateful to Eleanor Tursman, Alex Meyerowitz, Adam Pikielny, Harman Suri, and Lachlan Kermode who paved the way for the work in this thesis.

My sincere gratitude to Professor Chen Sun for being my thesis reader and offering invaluable suggestions.

Last but not least, a special shout-out to my friends, especially Cynthia Zhang, for your endless love and support.

Contents

Chapter 1

Introduction

Rarely can anyone deny the importance of deep learning nowadays: from virtual assistants like Siri to space exploration missions such as dark matter detection, deep learning is the cornerstone of various modern technologies. However, as well as it makes the world a better place it can also facilitate malicious acts. Deepfakes — fake videos of some individual(s) generated using deep learning — are among the most rampant ones. The recent deepfaked video of Ukrainian President Volodymyr Zelenskyy [1] asking his army to surrender is a notorious incident where deepfake was used to spread misinformation that could cause dire consequences.

Based on how the images or frames in a video are modified, deepfake creation can be roughly divided into three categories: replacement, reenactment, and editing. Replacement refers to the situation where the content of the source video is replaced with a fake one. Usually, it is the identity of the person in the video that is replaced with another, a process known as "face swap". Reenactment refers to the situation where one or more of the attributes of the person in the source video has been modified, or "reenacted". Different from replacement, reenactment involves more subtle facial or physical attributes manipulation than full identity swapping. These attributes include expression, gaze, pose, gesture, etc. Adjustment refers to any changes that do not modify the target's identity or attributes included in replacement and reenactment. For instance, the target's clothes [18], age, weight, or even the lighting condition [35] can be manipulated. Adjustment might not seem to cause as much damage compared to the first two categories of deepfakes, but this process could still help people achieve malicious purposes when false persona or contexts are created.

In this thesis, we focus on identity replacement and expression reenactment which are the most common forms of deepfakes. The former happens often when a video of a "dummy" person whose identity is replaced with that of a target person such that the target appears to take actions that were not taken by them. The latter is often used to make the person in the source video express emotions or say something that they did not mean to.

In light of the potential harm that deepfakes can cause, detection methods that classify videos as either real or fake have thus been developed to combat these video manipulations. In particular, deep learning based detection has been the most successful among all. Mainstream deep learning based detection methods can be grouped into two subcategories according to which aspect of the source video is examined.

The first group of methods looks for artifacts within video frames. Since current deepfakes are done on a frame-by-frame basis and not flawless, they can be detected by breaking down the video into frames and looking for visual artifacts in those frames. Then a classifier is used to separate real and fake videos. Nguyen et al. [25] combined VGG19 network and capsule networks with a dynamic routing algorithm for detecting pose-related manipulations in videos. Besides pose, artifacts are also commonly present in other features such as eyes, teeth, facial contours, etc. Matern et al. [22] extracted eyes and teeth feature vectors and used two classifiers, logistic regression and a small neural network, to detect fake frames.

The second group of methods focuses on temporal features across video frames. Instead of looking at individual frames, these methods examine the entire video sequence where artifacts in the frames often manifest themselves as temporal inconsistencies. A recurrent convolutional model (RCN) that integrates CNN and RNN was proposed by Sabir et al. [29] to identify temporal discrepancies. Likewise, Guera et al. [8] combined CNN with LSTM to capture temporal information. Physiological signal anomalies can also be used for detection. Li et al. [20] observed that a deepfaked individual has a significantly less blinking rate than a real one, so they came up with a threshold and classified videos into real and fake based on that. Caldelli et al. [2] studied the correlation between optical flow incoherence and the presence of deepfakes. They assumed that unnatural movements of facial features generated by deepfakes would give rise to abnormal motion patterns.

Despite considerable efforts that have been put into developing deepfake detection methods, we assume that with even more advanced technology future deepfakes will be indistinguishable from real video and thereby nullify current deepfake detection methods. Tolosana et al. [32] describe a need for improved detection methods as well as different settings that might exploit additional recordings available at capture time or afterward.

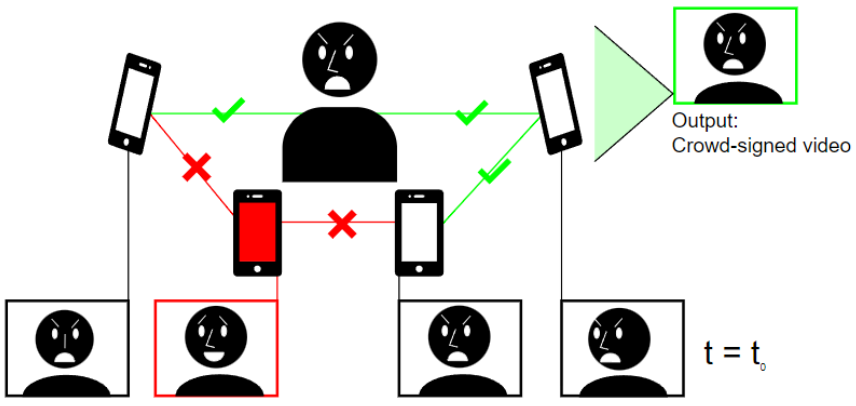


Figure 1.1: The social video verification setting [33].

Tursman et al. [33] posit a social verification setting (Figure 1.1) where a public event, such as a person speaking, is recorded by multiple devices within a certain time frame and some recordings may have been deepfaked. A reliable social verification system is supposed to correctly classify each video as real or fake through the social consensus of those recordings.

The task might seem trivial at first glance: human eyes are good at distinguishing different

appearances and expressions, so we ourselves can find the largest group of videos that are consistent with each other and those are exactly the real ones. But imagine, for instance, a group of reporters broadcasting a live press conference when some of their devices are hacked. A social video verification system will be able to confirm the validity of these videos in real time before any form of harm can be done. This, however, cannot be done by an average person. The high-level aim of this system is thus to prevent the circulation of fake videos by establishing a trustworthy version among different sources with great accuracy as well as speed.

Under such a setting, Tursman et al., Pikielny, and Meyerowitz (see Section 2.1 and Appendix) devised Machine Learning and graph based methods, achieving significantly better performances against a naive LSTM based classifier. But the classification accuracies of these methods are barely better than random guessing when they try to detect expression reenactment in videos.

One potential reason for the weak performances on expression reenactment is that these methods do not take full advantage of the high-level information such as semantic facial attributes (age, gender, emotion, etc.) in the source videos. We thereby leverage the power of deep learning that allows us to make use of those high-level information to propose a series of methods that attempt to build a more reliable social verification system.

Chapter 2

Related Work

2.1 Social Video Verification

In the preprocessing step of their approaches, Tursman et al. as well as Pikielny and Meyerowitz construct a matrix of feature vectors of the frames for each video from the cameras and perform PCA on those matrices individually. They then represent each camera using a vector of its frames' Mahalanobis distances between the projections from the feature vectors of those frames and the center of the PCA subspace.

Hierarchical Clustering To separate the cameras into real and fake subsets, Tursman et al. turn all cameras into a weighted binary tree using single-linkage hierarchical clustering. The ratio of the largest weight and the second largest weight in the tree is compared to an empirically derived threshold to determine if there is any fake. If there is, the tree is clustered again into two subtrees and cameras in the subtree that has a smaller sum of weights are classified as real.

Graph Cut Pikielny and Meyerowitz instead turn the cameras into a fully connected graph. They first estimate the distribution of the L_2 distances between all real camera vectors using a dip test and compute for each camera vector pair the probability that a random variable sampled from that estimated distribution is greater than or equal to the L_2 distance between the camera vector pair. Then they use these probabilities to assign weight to the edges in the graph and look for the maximum cut that separates the nodes into a real and a fake subset.

2.2 Generative Adversarial Networks (GANs)

First proposed by Goodfellow et al. [7] in 2014, GAN tries to produce outputs according to the distribution of training data. The backbone of a GAN consists of a generator G and a discriminator D: G tries to generate fake images from vectors in the latent space, a lower dimension space that represents training data, to fool D while D learns to differentiate between real images and fake ones produced by G in an "adversarial" process. Over the years, GANs have developed into one of the

most popular tools to generate high resolution and photorealistic images for humans, animals, and objects.

In particular, StyleGAN [15, 16, 14] marks the state-of-the-art in human face generation. Inspired by style transfer, StyleGAN's generator architecture consists of a mapping network that draws styles from a random vector and a synthesis network that generates images by performing affine transformations on the set of styles as shown in Figure 2.1. StyleGAN allows more control over image synthesis by modifying the styles on different scales, producing high quality and diverse results.

Another feature of StyleGAN is its relatively disentangled latent space W. A perfectly disentangled latent space has linear subspaces that each control a factor of variation on the output from the generator. This means it is possible to modify a single attribute on the output by moving its latent code along the corresponding subspace. The importance of this feature will be explained later in Section 3.1.

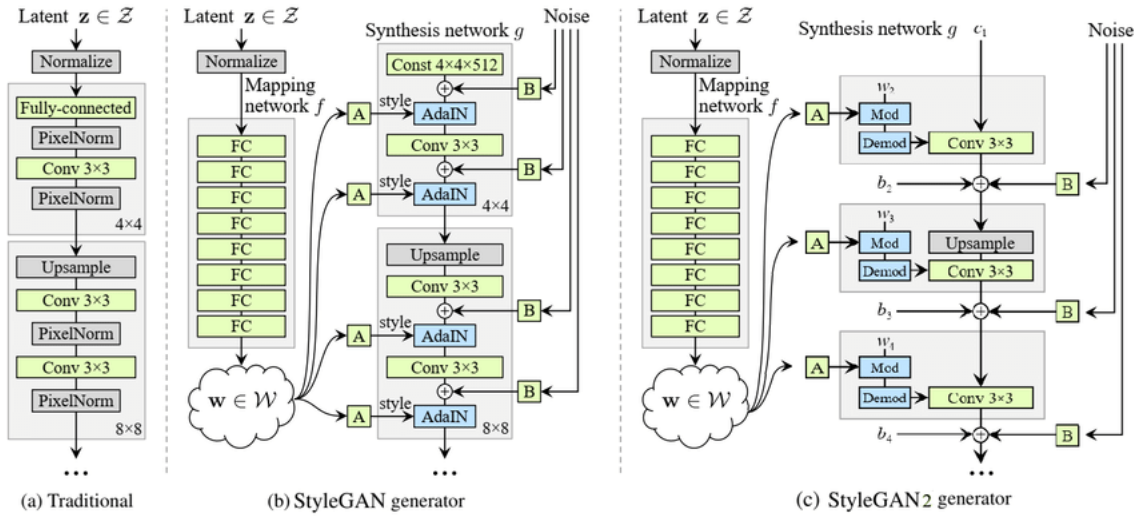


Figure 2.1: Structure of (a) PGGAN [13], (b) StyleGAN [15], and (c) StyleGAN2[16].

2.3 Human Face Novel View Synthesis

As a major component in mainstream face recognition methods, image synthesis for human faces has long been a topic of interest in the scientific world [21]. In particular, we have witnessed breakthroughs in terms of image quality and diversity in human face synthesis achieved by deep-learning-based approaches such as GANs.

In this thesis, we focus on novel view synthesis for human faces: given an input human face, a model needs to generate a photorealistic human face (high quality) viewed from a different angle while preserving the person's identity (high fidelity).

2D Human Face Novel View Synthesis Due to the high-quality face generation capability of GANs and/or their inherent latent space, many studies rely on them for latent space editing. This empowers their networks to manipulate a person's expression, age, pose, etc.

InterFaceGan by Shen et al [31] trained SVMs to find the hyperplane in StyleGAN latent space for each of the facial attributes and used vector arithmetic to achieve conditional attribute manipulation in Z space. Built upon InterFaceGan, Wang et al. [34] explored the Rate-Distortion tradeoffs between the image editability and reconstruction quality of StyleGAN's latent space and achieved high fidelity in both editing and reconstruction. On the other hand, GANspace by Härkönen et al. [11] performed unsupervised learning, mainly PCA, on the latent space in StyleGAN as well as feature layers in BigGAN to find the directions of some editable features.

3D Human Face Novel View Synthesis 3D image synthesis tries to capture the 3D shape or characteristics of the object in the input image before performing any edits.

Many of this kind of approaches incorporate GANs in their frameworks to leverage GANs' ability to generate and edit photorealistic, high-resolution results. FreeStyleGAN by Leimkühler et al. [17] constructs a camera manifold and a mapping between this manifold and StyleGAN latent space for free-viewpoint rendering. PiGAN by Chan et al. [4] adopts a sinusoidal representation network (SIREN) based implicit neural radiance field (NeRF) which produces interpretable and view-consistent 3D image synthesis. Apart from NeRF, other 3D representation techniques can also be part of the model architecture. Deng et al. [5] find in 3D space a reduced space consisting of several isosurfaces which they call radiance manifolds and predicts the color and occupancy for the intersection points between camera rays and the isosurfaces. StyleSDF by Or-EL et al. [26], on the other hand, uses an SDF volume renderer to learn explicit 3D geometry and a style-based generator that outputs high-resolution 2D images.

There are also non-GAN-based deep learning methods. Deng et al. [6] use CNN-based networks to predict the coefficient for a 3D Morphable Model (3DMM) that uses PCA bases for identity, expression, and texture and Spherical Harmonics for lumination. The learned 3DMM model is then able to reconstruct a 3D human face given an input image.

Chapter 3

Methods

Under the social video verification setting, we have a set of N cameras $c_1, c_2, \ldots, c_N$ that have captured the same person in the same time interval. The subset of all the cameras that produce real videos and those that produce fake videos are denoted by R and F respectively. The jth frame in the video captured by camera c_i is denoted by $I_{c_i,j}$. Note that if $|R| \leq |F|$ and F consists of a single type of fakes, the consensus reached by the cameras may lead to an incorrect conclusion. Therefore, we additionally assume that $|R| > |F|$.

3.1 Frontalization

Frontalization refers to the process of turning a side-view image I of a person into a frontal-view image $\hat{I}$ while preserving all attributes of the person. In an ideal case, all the frontalized frames with the same timestamp in the videos from cameras in R should be identical and also different from the frontalized frames in the videos from cameras in F. That is,

$$\hat{I}_{x,j} = \hat{I}_{y,j} \neq \hat{I}_{z,j} \quad \forall j \text{ and } x, y \in R \text{ and } z \in F$$

We can then use the pixel-wise difference between each frontalized frames to determine whether they are real or fake. Even if the frontalization is not perfect, it can still be a practical method as long as the difference between $\hat{I}_{x,j}$ and $\hat{I}_{y,j}$ is smaller than that between $\hat{I}_{x,j}$ and $\hat{I}_{z,j}$.

Naive Latent Code Edit The latent space in many GAN models can be exploited for face attribute editing. As mentioned earlier, StyleGAN has a relatively disentangled latent space W where there potentially exists a subspace associated with the viewing angle in the output. If we find such a subspace, we can manipulate the latent code $w_{c_i,j}$ in that subspace corresponding to frame $I_{c_i,j}$ to remove variations in the viewing angle. Three approaches are thereby proposed to achieve this goal.

(1) **Hyperplane Projection**

Apart from the less entangled latent space of StyleGAN, Shen et al. [31] further argue that for any binary attribute (e.g. smile, pose, eyeglasses, etc.) there exists a hyperplane in the latent

space such that all latent codes are highly separable on that attribute by the hyperplane. They proved their argument by successfully training a set of SVMs that can separate testing data according to pose (equivalent to viewing angle), smile, age, gender, and eyeglasses with high accuracies.

Using the norm vector of the pose boundary found by Shen et al., we perform an orthogonal projection of latent code $w_{c_i,j}$ onto the boundary for all i and j in an attempt to neutralize the viewing angle, i.e., frontalize all outputs.

(2) **PCA Projection**

Given that the set of N cameras capture a person from varying angles, we hypothesize that the viewing angle accounts for the largest variation in $I_{c_1,j}, \ldots, I_{c_N,j}$ as well as in the corresponding latent codes $w_{c_1,j}, \ldots, w_{c_N,j}$ for all j. We also know that the first principal component (PC1) in PCA represents the direction along which the data show the maximum variance. So to eliminate that variance, we can perform an orthogonal projection of $w_{c_i,j}$ onto the vector orthogonal to PC1 from PCA of latent codes $w_{c_i,j}$ for all i. In fact, that vector is the second principal component (PC2).

(3) **Dimension Replacement**

Instead of using vector arithmetic, we can directly make changes to the latent codes that would lead to desired outputs. All $w \in W$ are 512-dimensional vectors. We hypothesize that certain dimension(s) or a combination of some dimensions account for the variations in the viewing angle of the generated outputs. Under the assumption that the viewing angle accounts for the largest variation, we aim to find the dimensions that have the largest variance or standard deviation and replace them with a neutral value.

Specifically, we calculate the standard deviation $\sigma_1, \ldots, \sigma_{512}$ for each of the 512 dimensions where σ_k is computed for dimension k across all $w \in W$ and the mean $\bar{\sigma}$ as well as the standard deviation $\hat{\sigma}$ of all σ's. Then we find the set of dimensions S with $\sigma_i \geq \bar{\sigma} + \hat{\sigma}$, i.e., one standard deviation or more from the mean, and for each dimension $d \in S$ we replace the value with the mean value in dimension d across all w's.

Network-based Latent Code Edit Multi-Layer Perception (MLP) serves as the mapping network in a number of architectures [15, 16, 4] to perform vector transformation. We aim to find such a mapping $M : w \to \hat{w}$ that transforms the latent code w from a human face image to the latent $\hat{w}$ that corresponds to the frontal view of that human face. We designed an MLP-based network (Figure 3.1). Some prominent features of our network include the following:

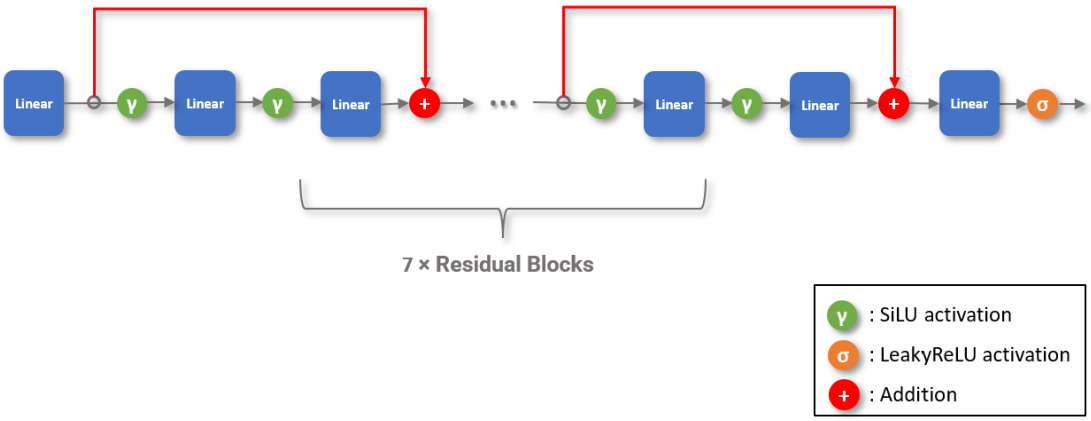


Figure 3.1: Structure of our network.

(1) **Residual Blocks**

Inspired by ResNet [10], we use multiple residual blocks in our MLP. A residual block (Figure 3.2) is able to learn the identity mapping, i.e. $f : x \to x$, so it can be "skipped" by the network. Hence, instead of manually tuning the total number of layers, the network itself learns which layers are beneficial for the model accuracy and dynamically sets the number of active layers.

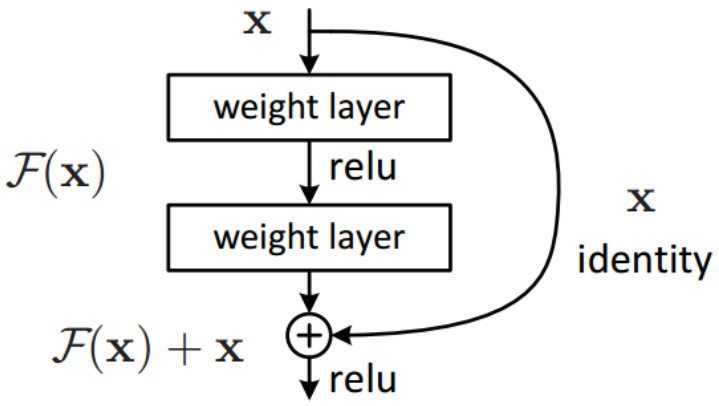


Figure 3.2: Structure of an example residual block [10].

(2) **Gradient Penalty**

Gradient penalty is a technique used in [9] to enforce the Lipschitz constraint under which the gradients of the network have a unit norm almost everywhere. We apply gradient penalty to our network such that the mapping function M transforms the input latent code w as smooth as possible to $\hat{w}$. This is expected to make M learn how to gradually turn a side-view latent code into a frontal-view one.

(3) **Latent to Image Feature Space Conversion**

During training, we discovered that converting the latent codes to feature vectors in the synthesis network of StyleGAN2 (Figure 2.1(c)) in our loss function helps improve the network performance. The latent codes in latent space W are the coarsest representations of the generated images, while those feature vectors comprise representations in all levels from coarse to fine. Loss function computed using feature vectors thus places more constraint on both the high-level and low-level details of the generated images, improving the image fidelity with respect to the identity of the input side-view image.

The loss function of the network is the weighted sum of the mean-squared-error(MSE) between the feature vectors and the gradient penalty

$$\mathcal{L}(G) = MSE(G(w^*)), G(\hat{w})) + \gamma \cdot gradient\ penalty$$

where w^* is the latent code for the ground-truth frontal view image, γ is the gradient penalty weight, and G is the function that takes in a latent code and outputs the set of corresponding feature vectors in the synthesis network of StyleGAN2.

Frontal View Synthesis by SOTA models As discussed in Section 2.3, 2D and 3D state-of-the-art (SOTA) models are able to generate high quality and high fidelity human faces given an input human face. This means that they may be able to achieve good frontalization results. Depending on the availability of source code and computational complexity, we performed frontalization using two of the SOTA models.

(1) **High-Fidelity GAN Inversion (HFGI)**

GAN inversion refers to the process of reconstruction of the input image by the GAN generator. Proposed by Wang et al. [34], HFGI aims to tackle the problem of accuracy and editability trade-off in GAN inversion: reconstruction accuracy increases at the expense of lower editing performance, i.e., the network's ability to edit the person's facial attributes in the input images suffers. Wang et al. argue that the inferior reconstruction accuracy is due to the low dimensional latent codes which are unable to encode image-specific details such as background and illumination. But increasing their dimension leads to less interpretable latent codes as well as a higher chance of an overfitting model, reducing the editing performance. In order to make the latent codes aware of the image-specific details without sacrificing the editing performance, the authors come up with a novel framework as shown in Figure 3.3.

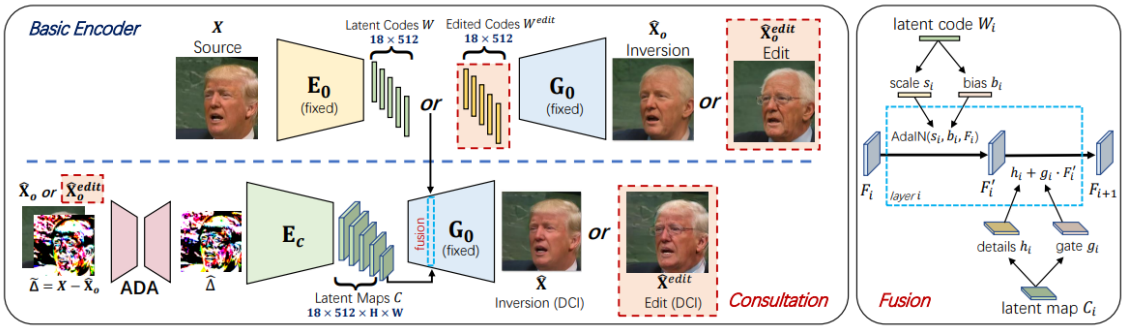


Figure 3.3: Structure of HFGI [13].

The loss in image-specific details $\tilde{\Delta}$ between the input X and the naive inversion result $\hat{X}_0$ or the naive inversion result from the edited latent code $\hat{X}_0^{edit}$ gets projected to a high-dimensional latent map C through the consultation encoder E_c. The latent codes W of the input or the edited latent codes W^{edit} are then combined with C through layer-wise fusion, a process called Distortion Consultation Inversion (DCI), to generate high-accuracy and editable reconstruction.

HFGI is thus able to edit the viewing angle of any frame X by using the hyperplane for the viewing angle in the StyleGAN latent space found by Shen et al. [31] to get W^{edit} and passing it through the network to get $\hat{X}^{edit}$.

(2) **Pi-GAN**

Periodic Implicit Generative Adversarial Networks (pi-GAN) [4] is a 3D-aware image synthesis model. It is 3D-aware in the sense that it learns the underlying 3D representation of the generated image, and specifically, pi-GAN uses a neural radiance field (NeRF) [23] representation for this purpose (Figure 3.4). Given a 3D coordinate x, a viewing direction d, and a set of frequencies γ's and phase shifts β's (produced by the latent code z and the mapping network), pi-GAN outputs the density σ and the color c. σ and c are subsequently used in neural rendering to generate the 2D image. The network also uses a sinusoidal representation network (SIREN) with feature-wise linear modulation (FiLM) which can represent fine details better and thereby produce sharper images than previous works such as GRAF by Schwarz et al. [30].

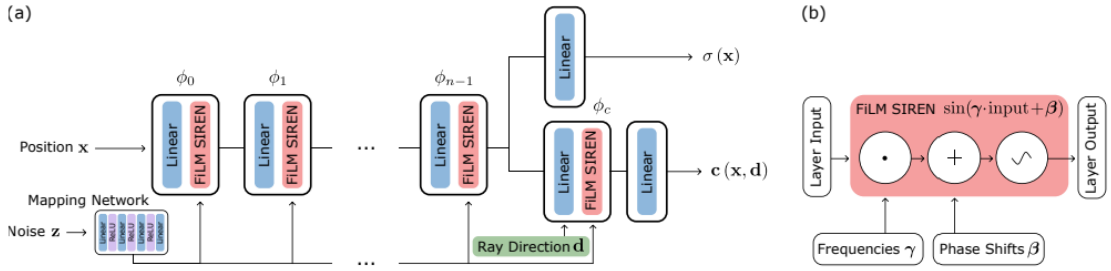


Figure 3.4: Structure of pi-GAN [4].

Given a source image, we can perform an optimization step by freezing all parameters other than γ and β to find the set of γ's and β's with which pi-GAN generates the image that best matches the source image. Once the radiance field for the source image is learned, we can perform frontalization by rendering images along the frontal direction.

3.2 3DMM-based Classification

As mentioned in Section 2.3, Deng et al. [6] propose a CNN-based method that predicts 3D Morphable Model (3DMM) coefficients for 3D face reconstruction. Its overall architecture is shown in Figure 3.5.

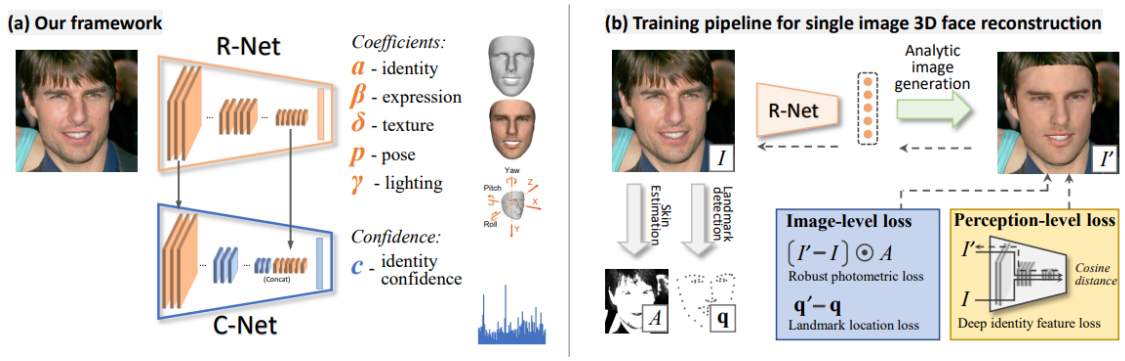


Figure 3.5: Structure of the 3D face reconstruction method by Deng et al. [4].

Based on the Basel face model [27] and the FaceWareshouse model [3], this method regresses the identity α, the expression β, and the texture δ, i.e., the coefficients for a 3DMM face model

$$S = S(\alpha, \beta) = \bar{S} + B_{id}\alpha + B_{exp}\beta$$
$$T = T(\delta) = \bar{T} + B_t\delta$$

where $\hat{S}$ and $\hat{T}$ are the average face shape and texture while B_{id}, B_{exp}, and B_t are the PCA bases of identity, expression, and texture respectively. It also regresses the face pose p from the perspective camera model and the Spherical Harmonics coefficients γ for lighting.

Given a set of video frames $I_{c_1,j}, \ldots, I_{c_N,j}$ from the set of N cameras, we extract identity coefficient vector $\alpha_{i,j}$ and expression coefficient vector $\beta_{i,j}$ for each frame $I_{c_i,j}$. Since in our problem setup, $|R| > |F|$, i.e., there are more real videos than fake videos, we can find the outliers among all videos using respective coefficients.

For instance, with real and fake expression data, we compute the mean expression coefficient vector $\bar{\beta}_{i,j}$ for any given i, j and the distances from each $\beta_{i,j}$ to the mean, $d_{1,j}, \ldots, d_{N,j}$, normalized by dividing the max distance such that the range is $[0, 1]$. For each camera i, we calculate the mean distance of all $d_{i,j}$'s, $\bar{d}_i$. These mean distances are compared against a threhold t. If $\bar{d}_i \leq t$, camera i is classified as real; otherwise it is classified as fake. The same process is applied on fake identity data. Additionally, the way to determine t is explained in Section 4.2.2.

Chapter 4

Experiments

4.1 Settings

4.1.1 Dataset

We use two datasets that both contain multi-view videos of human speakers and are used in different methods discussed in Section 3.

Tursman et al. [33] dataset The dataset consists of videos of 24 speakers set against a black background curtain. Six cameras are set up in an arc of 65° facing a seated speaker. To induce natural head and facial movement, speakers are prompted to answer random questions.

DeeperForensics [12] dataset The dataset consists of videos of 100 speakers set against a dark green background curtain. Seven cameras are set up in an arc of 120° facing a seated speaker. Speakers are requested to speak naturally with 8 target expressions.

4.1.2 Fake Data Generation

Fake Identity data We retrain a model [24] on CelebA-HQ [13] and FFHQ [15] datasets from scratch based on FaceShifter by Li et al. [19] and modify the identity of 15 individuals in each video frame from the DeeperForensics dataset.

Fake Expression Data We mainly consider the scenario where the visemes are altered, i.e. the speaker's mouth movement is modified such that they appear to say something else. To generate the fake viseme data, we take both datasets and use LipGAN by Prajwal et al. [28] that takes a video of a human speaker and arbitrary audio to generate the edited face with corresponding visemes.

4.1.3 Image to Latent

Before analyzing and manipulating the latent codes for frontalization, we need to convert the source frame to the corresponding latent code in StyleGAN latent space. We use the inverse projection method by Karras et al. [16] that performs an optimization step to find the latent code which

generates the output image that best matches the source frame. In particular, we get rid of the noise regularization in the optimization so that inverse projection deterministically outputs a latent code given a source frame.

4.2 Results

4.2.1 Frontalization

Naive Latent Code Edit We experiment with different methods described in Section 3.1 on the DeeperForensic dataset. We use synchronized real videos of the same subject to see if these methods can perform proper frontalization. Five consecutive frames from each video are included as inputs to make sure frontalization does not produce drastically different results across frames from the same video. One typical set of results from the experiments is presented below.

(1) **Original Frames**

Figure 4.1: Frames from cam1 (*top*), cam2 (*middle*), and cam5 (*bottom*) of the same subject.

(2) **Hyperplane Projection**

Figure 4.2: Frames from cam1 (*top*), cam2 (*middle*), and cam5 (*bottom*) of the same subject.

(3) **PCA Projection**

Figure 4.3: Edited frames from cam1 (*top*), cam2 (*middle*), and cam5 (*bottom*) of the same subject.

(4) **Dimension Replacement**

Figure 4.4: Edited frames from cam1 (*top*), cam2 (*middle*), and cam5 (*bottom*) of the same subject.

It can be observed that PCA projection achieves the best frontalization results in terms of both angle adjustment and identity preservation. This means our hypothesis that the viewing angles of the ith frames from all cameras account for the largest variation in their corresponding latent codes is valid. However, the frontalization by PCA projection still does not adequately preserve the identity of the subject in the original frames. Overall, naive latent code edit is unable to perform ideal frontalization.

Network-based Latent Code Edit To confirm if our MLP-based network can potentially perform frontalization as intended, we first train the network on a constrained frontalization task: given a side-view image of a person from a pre-determined angle, frontalize the image.

Using the Adam optimizer with an initial learning rate 5×10^{-5} and a cosine annealing schedule, we train the network for 400 epochs with different gradient penalty weight γ. Training data come

from video frames captured by one of the side-view cameras along with those captured by the frontal-view camera as training labels in the Tursman et al. dataset. Test data come from video frames captured by the camera that has a similar side-view angle in the DeeperForensics dataset. The training and test results are shown in Figure 4.5 rows 1-6.

The performance of the model is measured by the pixel-wise difference between the output images and the corresponding ground-truth images. Alternatively, a good performance can be indicated by no visual difference between the output and ground-truth images. We can see that without gradient penalty, the network cannot even generate photorealistic images despite performing relatively well on the training data. Among the networks that were trained using gradient penalty, the one with gradient penalty weight $\gamma = 1$ has the best performance on the training data. But overall, performance on the test data is far worse than that on the training data. This suggests that the network might have memorized each input-output pair without truly learning how to frontalize an image.

This problem may be solved by using training data that consist of video frames from multiple side-view cameras. This makes memorizing input-output pairs much more difficult and thereby forces the network to learn the frontalization method which can be applied to images that are not from the training data. At the same time, it lifts the constraint on the frontalization task, which means the network could take images from any viewing angle.

Using four side-view cameras in Tursman et al. dataset, we train the network under the same settings as before with $\gamma = 1$. The training and test results are shown in Figure 4.5 row 7 and Figure 4.6. We can see that the network is indeed able to generate photorealistic frontal-view images from a random viewing angle and preserve some attributes of the person in the ground-truth image such as expressions and areas around the eyes. However, the identity of the person is not preserved and completely different identities are generated from the same person in the ground-truth images. In other words, our network fails to capture all relevant features of the person in the input image, and thus the frontalization performed by our network has good quality but lacks fidelity.

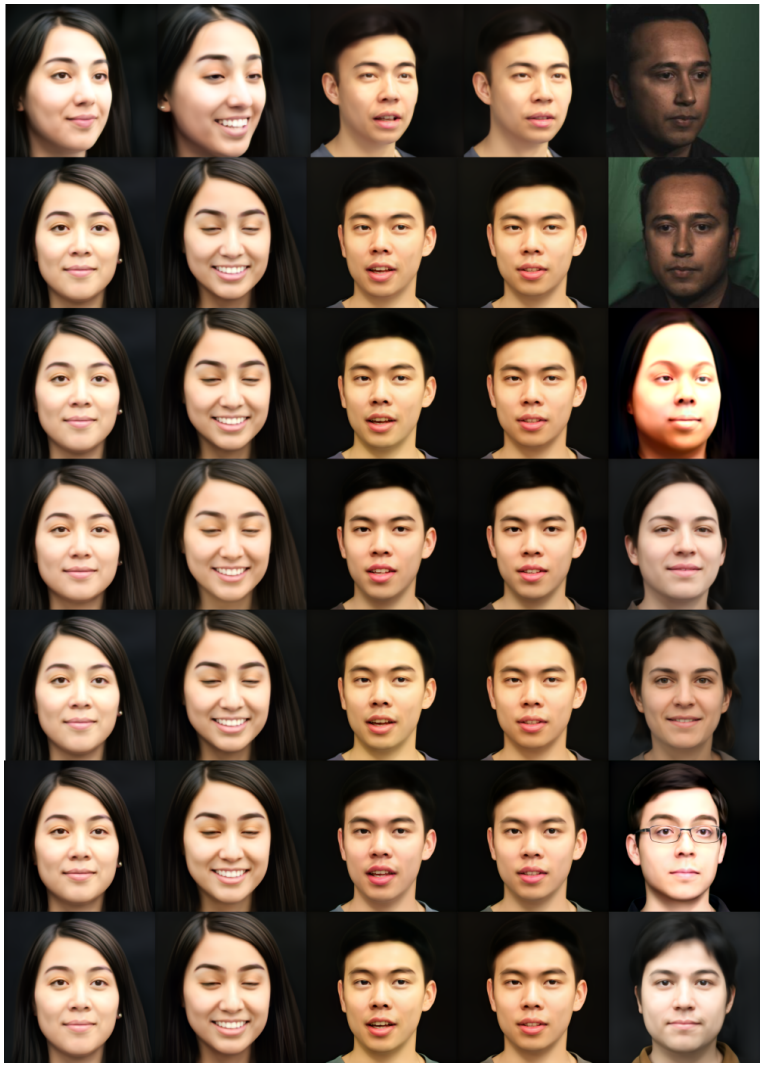
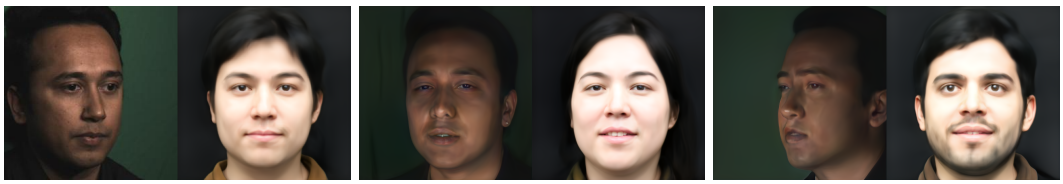


Figure 4.5: Rows 1-7 (*from top to bottom*): ground-truth side-view images, ground-truth frontal-view images, output images without gradient penalty, output images with $\gamma = 100$, output images with $\gamma = 1$, output images with $\gamma = 0.01$, output images with multi-view training data. Columns 1-4 (*from left to right*): video frames of subjects in training data. column 5: video frame of subject in test data.



(a) (b) (c)

Figure 4.6: (a)(b)(c): Ground-truth side-view images of the same subject (*left*) and output images from our network (*right*).

Frontal View Synthesis by SOTA models We leverage the novel view synthesis power of the two SOTA models mentioned in Section 3.1, HFGI [34] and pi-GAN [4], to edit the viewing angle. Frontalization results on the Tursman et al. dataset are shown below.

(1) **HFGI**

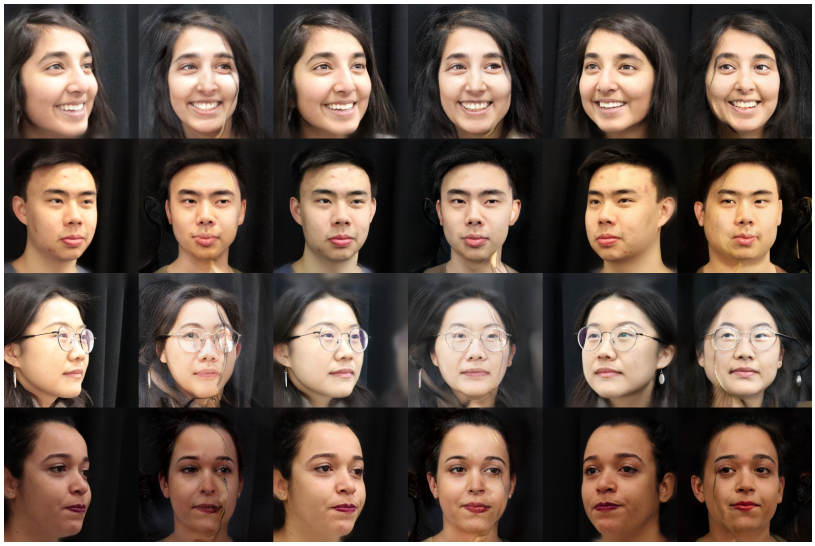


Figure 4.7: Columns 1,3,5 (*from left to right*): Ground-truth frames captured by different cameras at the same time from Tursman et al. dataset. Columns 2,4,6: Frontalized outputs from HFGI.

The results from HFGI show noticeable visual artifacts, especially when the subject wears eyeglasses. The frontalized outputs from original frames with varying viewing angles for the same individual also differ considerably.

(2) **Pi-GAN**

Figure 4.8: Column 1 (*from left to right*): Ground-truth side-view images. Columns 2-5: Novel views generated by pi-GAN.

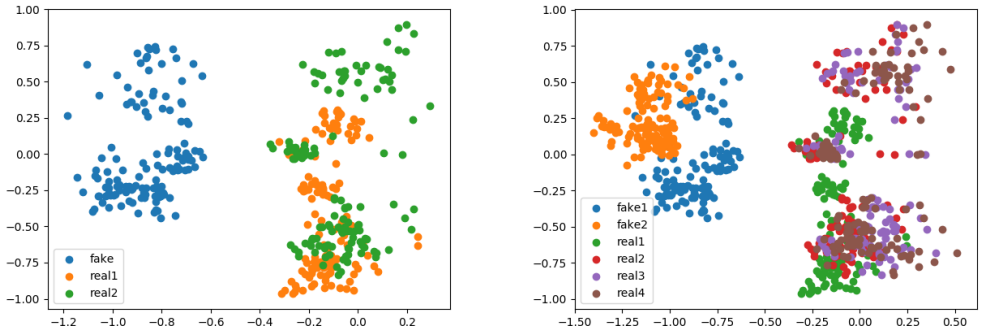
Due to our machine's limited computational capacity, we were only able to set the output resolution to 64×64 for a reasonable training time. But the fact that the outputs are not photorealistic, especially around the eye areas, is not caused by the low resolution. It is instead because the 3D coordinate x and the viewing direction d required by pi-GAN are difficult to

know exactly. Our estimation was most likely not good enough for the network to produce high quality results.

4.2.2 3DMM-based Classification

To confirm the method by Deng et al. [6] could potentially work, we first analyze if the frames from the real and fake videos are separable using the relevant 3DMM coefficients. As mentioned in Section 3.2, the coefficients are for the PCA bases used by the 3DMM face model, so we could focus on the coefficients for the first two principal components which account for the largest variations in the corresponding attribute. We thereby extracted identity coefficients $\alpha_{i,j}$ and expression coefficients $\beta_{i,j}$ for each frame $I_{c_i,j}$ and then made 2D scatter plots of the first two coefficients for each attribute.

From Figure 4.9 and Figure 4.10, we can see that the identity coefficients for the frames from real and fake identity videos are perfectly separable while the expression coefficients for the frames from real and fake identity videos are overlapping each other. Although not separable, the distributions of the latter do differ between those from real videos and those from fake videos. This means that it is still possible to distinguish between real and fake expression data.



(a) (b)

Figure 4.9: 1st (x-axis) and 2nd (y-axis) identity coefficient of frames from (a) 2 real videos and 1 fake identity video (b) 4 real videos and 2 fake identity videos of the same subject.

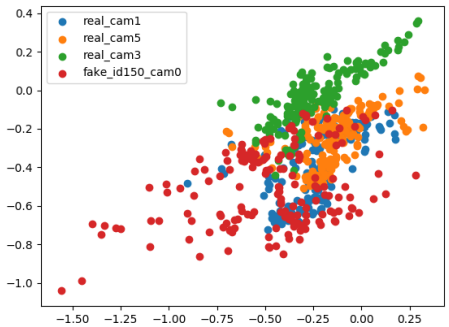


Figure 4.10: 1st (x-axis) and 2nd (y-axis) expression coefficients of frames from 3 real videos and 1 fake expression video of the same subject.

We then devise the approach described in Section 3.2. Before mixing fake identity and expression data in F, we first deal with an easier situation where only one type of fake data is present. This is also in line with the assumption in previous works, which makes makes performance comparison and analysis more convenient. To find the optimal threshold t, we

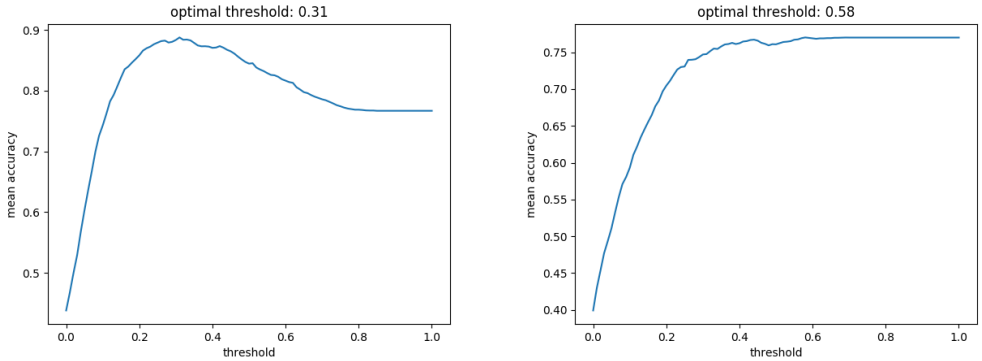
(1) Randomly select a subject from the DeeperForensic dataset and a set of real cameras R and another set of fake identity or expression cameras F for that subject such that $|R| > |F|$ and $|F| \geq 0$.

(2) Compute the mean distance $\bar{d}_i$ for the identity or expression coefficient vectors of all frames from each camera i.

(3) For each threshold in $[0.00, 0.01, 0.02, \ldots, 0.99, 1.00]$, classify all cameras and compute the classification accuracy $= \frac{\text{number of correctly classified cameras}}{|R|+|F|}$.

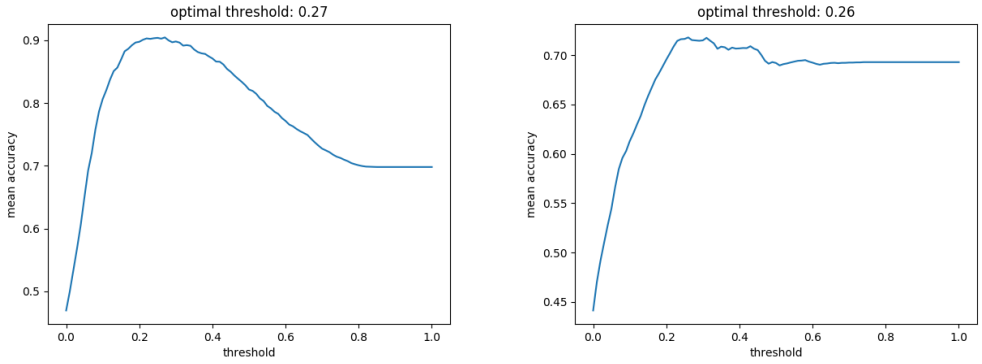
and repeat all steps multiple times to get the average accuracy for each threshold (Figure 4.11). Note that in the social video verification setting, true positive rate and true negative rate are equally important since misclassifying either real or fake would cause comparable confusion. Classification accuracy is thus the first metrics we use for evaluation.

We also notice that the optimal thresholds become smaller (Figure 4.12) when $|F|$ is limited to be nonzero, i.e. $|F| > 0$. This difference arises from the case when $|F| = 0$ where the classification accuracy for larger thresholds would be 1, as long as the threshold is greater than the max mean distance to the mean coefficient vector of all real cameras. This explanation is supported by the fact that there is a significant decrease in mean accuracy for the larger threshold values when $|F| > 0$.



(a) (b)

Figure 4.11: threshold v.s. accuracy plot with $|F| \geq 0$ when (a) fake identity data can be present (b) fake expression data can be present.



(a) (b)

Figure 4.12: threshold v.s. accuracy plot with $|F| > 0$ when (a) fake identity data can be present (b) fake expression data can be present.

To see how well the optimal thresholds perform under different real and fake camera combinations, we calculate their classification accuracies and compare them with those from other methods. Apart from previous works introduced in Section 2.1, we also include a baseline approach: given that there are more real cameras than fake ones, one way to "cheat" the system is to classify all cameras as real and it would be consistently better than random guessing. Results are shown in Table 4.1 and Table 4.2.

(#real, #fake) / **Method**	(6,0)	(6,1)	(5,2)	(4,3)
Baseline	**1.000**	0.857	0.714	0.571
Hierarchical Clustering	0.724	0.975	0.891	0.818
Graph Cut	0.930	0.919	0.962	**0.890**
Our Method with $t = 0.31$	0.870	**0.989**	**0.971**	0.735
Our Method with $t = 0.27$	0.778	0.962	0.959	0.756

Table 4.1: Accuracies of different methods when fake identity data can be present.

(#real, #fake) / **Method**	(6,0)	(6,1)	(5,2)	(4,3)
Baseline	**1.000**	**0.857**	0.714	0.571
Hierarchical Clustering	0.490	0.644	0.209	0.065
Graph Cut	0.110	0.275	0.501	0.572
Our Method with $t = 0.58$	**1.000**	0.849	**0.719**	0.578
Our Method with $t = 0.26$	0.767	0.751	0.716	**0.707**

Table 4.2: Accuracies of different methods when fake expression data can be present.

Compared to previous works and the baseline approach, our 3DMM-based classification performs very well in detecting fake identity data when $|R| >> |F|$, but not so much when $|R| \approx |F|$. This may suggest that the variance of identity coefficient vectors causes the mean distances of real and fake cameras to get mixed up sometimes, especially when more fake cameras are present, hurting the classification accuracy.

On the other hand, 3DMM-based classification significantly outperforms previous works in detecting fake expression data. But with $t = 0.58$, our method has nearly identical accuracies as the baseline approach. This is mainly because the difference in mean distances of real and fake cameras on expression coefficients is generally smaller than that on identity coefficients. A large threshold like 0.58 would often classify all cameras as real so our method behaves almost the same as the baseline. With a smaller threshold, our method has relatively consistent performance across all camera combinations and beats the others by a large margin when $|R| \approx |F|$.

Overall, our method shows a solid improvement in classification accuracy over previous works, but it still fails to outperform the naive "all real" approach in every situation. This is especially the case with fake expression data, a problem we had foreseen since fake expression is by nature more subtle and thus harder to detect than fake identity.

Chapter 5

Discussion

5.1 Limitations and Future Work

Among different methods proposed in this thesis, 3DMM-based classification shows the most promising results. To potentially improve its classification accuracy, we will look for better means to calculate distances that take advantage of the fact that some coefficients correspond to more crucial principal components from the PCA bases. Metrics such as precision and recall should also be included for a more holistic performance evaluation. Most importantly, we plan to adapt our method for the general case where both identity and expression manipulations can be present.

On the other hand, perfect or even decent frontalization is a challenging, if not impossible, task to accomplish, since it is trying to recover complete information (frontal view) from partial information (side view). Even SOTA models are unable to produce satisfactory results. Our own MLP-based network suffers from overfitting and there is no panacea to tackle the issue. With millions of trainable parameters, training and tuning our model can be time-consuming. Nonetheless, we will try out different ways to enhance its robustness such as adjusting the model size or applying more constraints in training.

Another important aspect to consider is the overall social video verification setting. The assumption that there are always more real videos than fake ones is not very applicable in a real-world context — videos that are edited in different ways may well outnumber unedited videos. Ideally, our model needs to maintain its performance under any situation as long as the real videos have a higher count than any set of videos that are edited in the same way.

5.2 Conclusion

Results of the experiments in this thesis demonstrate that deepfake detection under the social video verification setting is a non-trivial task. The proposed deep learning based methods are able to combat deepfakes to various extents, but there has not been one that excels in every possible situation. Hopefully, this will not be the case by the time when, sooner or later, deepfakes become flawless.

Bibliography

[1] Bobby Allyn. Deepfake video of zelenskyy could be 'tip of the iceberg' in info war, experts warn. https://www.npr.org/2022/03/16/1087062648/deepfake-video-zelenskyy-experts-war-manipulation-ukraine-russia, March 2022.

[2] Roberto Caldelli, Leonardo Galteri, Irene Amerini, and Alberto Del Bimbo. Optical flow based cnn for detection of unlearnt deepfake manipulations. *Pattern Recognition Letters*, 146:31–37, 2021.

[3] Chen Cao, Yanlin Weng, Shun Zhou, Yiying Tong, and Kun Zhou. Facewarehouse: A 3d facial expression database for visual computing. *IEEE Transactions on Visualization and Computer Graphics*, 20(3):413–425, 2014.

[4] Eric Chan, Marco Monteiro, Petr Kellnhofer, Jiajun Wu, and Gordon Wetzstein. pi-gan: Periodic implicit generative adversarial networks for 3d-aware image synthesis. In *arXiv*, 2020.

[5] Yu Deng, Jiaolong Yang, Jianfeng Xiang, and Xin Tong. Gram: Generative radiance manifolds for 3d-aware image generation. In *arXiv*, 2021.

[6] Yu Deng, Jiaolong Yang, Sicheng Xu, Dong Chen, Yunde Jia, and Xin Tong. Accurate 3d face reconstruction with weakly-supervised learning: From single image to image set. In *IEEE Computer Vision and Pattern Recognition Workshops*, 2019.

[7] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks, 2014.

[8] David Güera and Edward J Delp. Deepfake video detection using recurrent neural networks. In *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pages 1–6. IEEE, 2018.

[9] Ishaan Gulrajani, Faruk Ahmed, Martín Arjovsky, Vincent Dumoulin, and Aaron C. Courville. Improved training of wasserstein gans. *CoRR*, abs/1704.00028, 2017.

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[11] Erik Härkönen, Aaron Hertzmann, Jaakko Lehtinen, and Sylvain Paris. Ganspace: Discovering interpretable gan controls. In *Proc. NeurIPS*, 2020.

[12] Liming Jiang, Ren Li, Wayne Wu, Chen Qian, and Chen Change Loy. Deeperforensics-1.0: A large-scale dataset for real-world face forgery detection. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 2889–2898, 2020.

[13] Tero Karras, Timo Aila, Samuli Laine, and Jaakko Lehtinen. Progressive growing of gans for improved quality, stability, and variation, 2017.

[14] Tero Karras, Miika Aittala, Samuli Laine, Erik Härkönen, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Alias-free generative adversarial networks. In *Proc. NeurIPS*, 2021.

[15] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks, 2018.

[16] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of StyleGAN. *CoRR*, abs/1912.04958, 2019.

[17] Thomas Leimkühler and George Drettakis. Freestylegan: Free-view editable portrait rendering with the camera manifold. 40(6), 2021.

[18] Kathleen Lewis, Srivatsan Varadharajan, and Ira Kemelmacher-Shlizerman. Vogue: Try-on by stylegan interpolation optimization. 01 2021.

[19] Lingzhi Li, Jianmin Bao, Hao Yang, Dong Chen, and Fang Wen. Faceshifter: Towards high fidelity and occlusion aware face swapping, 2020.

[20] Y. Li, M. Chang, and S. Lyu. In ictu oculi: Exposing ai created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7, Dec 2018.

[21] Zhihe Lu, Zhihang Li, Jie Cao, Ran He, and Zhenjun Sun. Recent progress of face image synthesis. pages 7–12, 11 2017.

[22] Falko Matern, Christian Riess, and Marc Stamminger. Exploiting visual artifacts to expose deepfakes and face manipulations. In *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, pages 83–92, 2019.

[23] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: Representing scenes as neural radiance fields for view synthesis. In *ECCV*, 2020.

[24] Mindslab-AI. Faceshifter unofficial implementation, 2021.

[25] Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos, 2018.

[26] Roy Or-El, Xuan Luo, Mengyi Shan, Eli Shechtman, Jeong Joon Park, and Ira Kemelmacher-Shlizerman. StyleSDF: High-Resolution 3D-Consistent Image and Geometry Generation. *arXiv preprint arXiv:2112.11427*, 2021.

[27] Pascal Paysan, Reinhard Knothe, Brian Amberg, Sami Romdhani, and Thomas Vetter. A 3d face model for pose and illumination invariant face recognition. In *2009 Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pages 296–301, 2009.

[28] Prajwal K. Renukanand, Rudrabha Mukhopadhyay, Vinay P. Namboodiri, and C.V. Jawahar. A lip sync expert is all you need for speech to lip generation in the wild. In *Proceedings of the 28th ACM International Conference on Multimedia*. ACM, October 2020.

[29] Ekraam Sabir, Jiaxin Cheng, Ayush Jaiswal, Wael AbdAlmageed, Iacopo Masi, and Prem Natarajan. Recurrent convolutional strategies for face manipulation detection in videos, 05 2019.

[30] Katja Schwarz, Yiyi Liao, Michael Niemeyer, and Andreas Geiger. GRAF: generative radiance fields for 3d-aware image synthesis. *CoRR*, abs/2007.02442, 2020.

[31] Yujun Shen, Jinjin Gu, Xiaoou Tang, and Bolei Zhou. Interpreting the latent space of gans for semantic face editing. In *CVPR*, 2020.

[32] Ruben Tolosana, Ruben Vera-Rodriguez, Julian Fierrez, Aythami Morales, and Javier Ortega-Garcia. Deepfakes and beyond: A survey of face manipulation and fake detection. *Information Fusion*, 64:131–148, December 2020.

[33] Eleanor Tursman, Marilyn George, Seny Kamara, and James Tompkin. Towards untrusted social video verification to combat deepfakes via face geometry consistency. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, June 2020.

[34] Tengfei Wang, Yong Zhang, Yanbo Fan, Jue Wang, and Qifeng Chen. High-fidelity gan inversion for image attribute editing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2022.

[35] Hao Zhou, Sunil Hadap, Kalyan Sunkavalli, and David Jacobs. Deep single-image portrait relighting. pages 7193–7201, 10 2019.

Chapter 6

Appendix

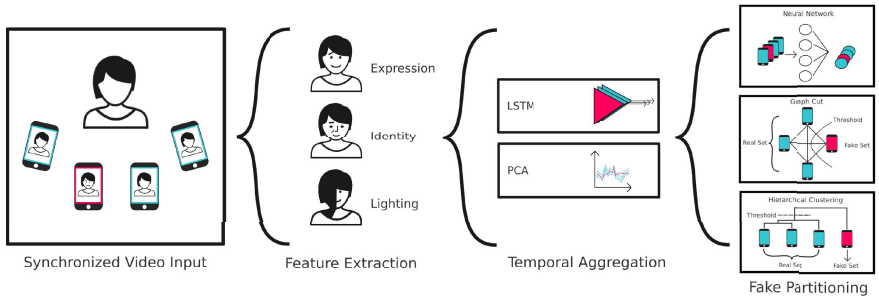


Figure 1: Overview of our scenario and evaluation setup. A set of synchronized video captures contains fakes across expression, identity, or lighting. We compare metrics with different partitioning approaches to determine which cameras belong to the real and fake sets.

high-dimensional clusters, and detect fakes if they fall outside these clusters [3]. Such approaches require an existing corpus of data for a particular speaker. Agarwal et al. match phoneme and viseme pairs to detect expression or audio forgeries [4], and Hosler et al. consider mismatched video and audio emotion [13].

Detection can be aided by creating and publicly releasing data. Khodabakhsh et al. create a dataset of manipulated images—Fake Faces in the Wild [19]. Rossler et al. create a dataset of fake videos called FaceForensics++ [35], and determine that expression transfer methods like Face2Face are most difficult for humans to detect—this motivates the need to detect subtle manipulations in image content. These datasets are all single view, so are insufficient for the social verification problem. Both Tursman et al. [42] and Jiang et al. [16] introduce datasets captured with multiple cameras with sets of diverse actors in a close-up shot setting; we use these to evaluate detection methods.

One final consideration for detection methods is interpretability. Verdoliva [43] describes that neural network detection methods are hard to interpret, which is insufficient for scenarios requiring the interpretation of detected manipulated video, e.g., in legal contexts.

3 Method

Social video verification assumes a set of C capturing cameras $c_1,...,c_i,...,c_C$ viewing the same event in real time, with a subset of real videos R and faked videos F such that $|R| > |F|$ (Fig. 1). The task is to identify and separate the set of real videos from fake videos.

We assume that frame-accurate synchronization is provided by communication between cameras via peer-to-peer wireless networks [5]. Given the setting, it is important for any detection method to operate efficiently in compute and communication size, i.e., to not require the transfer of video frames between devices, such that C can be large. Our goal is to extract a compact set of features *per camera* that can then be used to efficiently cluster cameras, such that video frames do not have to be shared between cameras. In Sec. 4, we will describe experiments with specific features for facial expression, identity, and lighting.

Given a set of features per frame $\{\vec{m}_{c_i,1},\vec{m}_{c_i,2},...,\vec{m}_{c_i,n}\}$, where each camera c_i captures a video of n frames, we consider how to integrate temporal information to capture variance differences between views via functions over windows in time (Section 3.1). Then, we describe methods to separate our cameras into real and fake subsets (Section 3.2).

3.1 Describing Features over Time

Oftentimes, clues over time can reveal a fake. Sometimes, features extracted from images may show temporal instability. For both of these reasons, we describe methods for time.

For each camera c_i, we define an $f \times n$ matrix M_{c_i} that stores the set of features with f components each extracted over a window size of n frames:

$$M_{c_i} = \begin{bmatrix} \vec{m}_{c_i,1} \\ \vec{m}_{c_i,2} \\ \vdots \\ \vec{m}_{c_i,n} \end{bmatrix} = \begin{bmatrix} m_{1,1} & m_{2,1} & \dots & m_{f,1} \\ m_{1,2} & m_{2,2} & \dots & m_{f,2} \\ \vdots & & & \vdots \\ m_{1,n} & m_{2,n} & \dots & m_{f,n} \end{bmatrix}.$$

Classical PCA To represent variation compactly as a signal over time per camera c_i, one approach proposed by Tursman et al. [42] is to perform classical PCA on each of the matrices $M_{c_1},...,M_{c_C}$. At each frame, we represent the frame as the distance from a feature's projected position in the PCA subspace for M_{c_i} to the center of that same PCA subspace. We calculate this length using the Mahalanobis distance metric, which can be used to find outliers in multivariate comparisons. We define this $n \times 1$ distance vector for c_i as $\vec{d}_i$, where:

$$\vec{d}_i = \sqrt{\sum_{cols} (M_{c_i}^{proj} - A)^T cov(M_{c_i})^{-1} (M_{c_i}^{proj} - A)} \quad (1)$$

$$M_{c_i}^{proj} = (M_{c_i} - \bar{M}_{c_i})E \quad (2)$$

Intuitively, quantifying and comparing feature variance over n frames intends to ignore absolute appearance factors. For example, to detect unusual face motion, this approach ignores the pose of the head but still captures whether face parts are moving in similar ways.

Given the occasional noise in feature representations, we also compare to a robust PCA approach [32] (with details in supplemental material). This is significantly more expensive to compute but can accommodate outliers within M.

3.2 Separating Real and Fake Camera Subsets

Given a set of distances D, we wish to separate the real and fake subsets. We assume the scenario where fakes in F are manipulated in a consistent way, and so should group similarly.

Hierarchical clustering We include a baseline approach followed by Tursman et al. [42]. Given a set of vectors $\vec{d}_1, \vec{d}_2, ..., \vec{d}_C$, we obtain a weighted binary tree t by single-linkage hierarchical clustering, using the L^2 distance between pairs of tree leaves to repeatedly join the minimal distance clusters. Motivated by the idea that real and fake camera clusters should be far from each other, we consider a variant that computes the sum L^2 feature distance to *all other* leaf nodes.

$$d_i = \sum_{c_j \neq c_i} \sqrt{\sum_{k=0}^{n} (\vec{d}_{c_j,k} - \vec{d}_{c_i,k})^2} \quad (3)$$

Given a clustering, we use the final tree's shape and costs to determine which, if any, of the inputs are faked by inconsistency with the majority (Fig. 1). For a tree t with k connections and a distance weight w per connection, we calculate:

$$\phi = w_k / w_{k-1}, \quad (4)$$

where ϕ represents the ratio of the cost of the ultimate and penultimate connections in t. If ϕ is below this threshold, then we say that there are no fakes present. If ϕ is greater than a set threshold τ, we

say that there is inconsistency within our input set of cameras, and at least one fake is present. To find which cluster is the real cluster, we re-cluster the tree into two sets and pick the cluster with the smallest overall distances. We set τ between 1.05–1.9 empirically based on labeled datasets and features.

Graph cut with distribution-reweighted edge distances. We also compare to an approach based on a graph weighted by the probability that real and fake cases will lie in different modes. For each timestep, we model each camera as a node in a fully connected graph, to be cut into the two real and fake subgraphs. We expect edge distances to have separate distributions—larger distances are likely to result from two nodes belonging to different real/fake subgraphs. We analyze the distribution of all feature distances. If the distribution is bimodal, then we anticipate a fake; conversely, if the distribution is unimodal, then we do not.

We use a dip test [12] to find the probability that our data is unimodal, and use it to split our set of distances into subset distributions. Since the within-class-subgraph distribution, L, will have smaller distances, it will be the leftmost distribution discovered by the dip test. Then, we find the probability that each edge distance d could have taken on its value or greater assuming it belongs to L:

$$p = P(L > d) \tag{5}$$

We want edges in our graph to have a high weight when p is low—edges that connect real and fake nodes should have the largest weight. As such, each edge weight E is set according to:

$$E = \frac{1}{p^2} \tag{6}$$

Finally, given the distribution-reweighted edge distances, we find the maximum cut in the graph to partition the nodes into a real and fake set.

To handle isolating the zero fake case, we use the distributions analyzed in previous steps. First, across all possible cuts, we consider the likelihood of the max cut occurring, p_m. Then, we combine it with the probability of the feature-distance distribution being unimodal, p_u. Our time instance is then given a value $T = p_m \times p_u$. If $p_m p_u$ is above a threshold τ, then we consider the set of videos to all be real. As before, we set τ between 0.01 and 0.10 empirically.

3.3 End-to-end Learned Separation

Finally, we describe a baseline learning approach that separates cameras into real and fake subsets. This method uses *generic* visual features. First, we run a ResNet18 model pretrained on ImageNet upon each image and extract features from the last layer of the network. These features are flattened into a 32,768-dim vector and fed into an LSTM network with 1,024 hidden units to aggregate information over time. The hidden state of the LSTM is then taken as a summary vector at each timestep and fed into a layer of 1000 fully-connected neurons. These output to a set of logits, one per camera, that output a binary classification of real or fake. Within this model, the analogy to the other methods is that the LSTM acts as a temporal aggregator (Sec. 3.1), and the fully-connected layer acts to separate the real and fake subsets (Sec. 3.2), except that both are trained simultaneously.

4 Experiments

Our experiments consider falsifications that alter identity (Sec. 4.1), expression (Sec. 4.2), and lighting (Sec. 4.3). The classic 'face swap' setting of deep fakes is the same as our identity setting. The expression and lighting experiments show more subtle manipulation while preserving identity.