



NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

CHALLENGES OF USING INCONSISTENT HEAD POSES TO CLASSIFY DEEPFAKES

by

Kevin D. Lutz

March 2021

Thesis Advisor:
Second Reader:

Robert L. Bassett
Ross J. Schuchard

Approved for public release. Distribution is unlimited.

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			<i>Form Approved OMB No. 0704-0188</i>	
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instruction, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington headquarters Services, Directorate for Information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188) Washington, DC 20503.				
1. AGENCY USE ONLY (Leave blank)		2. REPORT DATE March 2021	3. REPORT TYPE AND DATES COVERED Master's thesis	
4. TITLE AND SUBTITLE CHALLENGES OF USING INCONSISTENT HEAD POSES TO CLASSIFY DEEPFAKES			5. FUNDING NUMBERS	
6. AUTHOR(S) Kevin D. Lutz				
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) Naval Postgraduate School Monterey, CA 93943-5000			8. PERFORMING ORGANIZATION REPORT NUMBER	
9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES) N/A			10. SPONSORING / MONITORING AGENCY REPORT NUMBER	
11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.				
12a. DISTRIBUTION / AVAILABILITY STATEMENT Approved for public release. Distribution is unlimited.			12b. DISTRIBUTION CODE A	
13. ABSTRACT (maximum 200 words) Political leaders, military leaders, and the general public gather information from images and video to make decisions. Media can be spread instantaneously throughout the world at low cost and anonymously using social media, allowing small groups to gain powerful influence. This leaves the United States vulnerable to deception by media forgery. The problem of media forgery is not new, but the recent advances in machine learning have led to the development of DeepFakes, which are more sophisticated and difficult to identify. DeepFakes are especially dangerous because they can be used to change the identity of a person in a video or image. Detecting DeepFakes has been the focus of academic research, and several techniques have been developed and show promising results on available data sets, but the suitability of these algorithms for deployment within the Department of Defense or any other critical environment is an open question. This research provides a decisive answer to this question for a promising recent analytic that uses inconsistent head poses to detect when an image is manipulated.				
14. SUBJECT TERMS DeepFake, deep learning, machine learning			15. NUMBER OF PAGES 85	
			16. PRICE CODE	
17. SECURITY CLASSIFICATION OF REPORT Unclassified	18. SECURITY CLASSIFICATION OF THIS PAGE Unclassified	19. SECURITY CLASSIFICATION OF ABSTRACT Unclassified	20. LIMITATION OF ABSTRACT UU	

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

**CHALLENGES OF USING INCONSISTENT HEAD POSES TO CLASSIFY
DEEPFAKES**

Kevin D. Lutz
Lieutenant, United States Navy
BS, Tennessee Technological University, 2014
BS, Tennessee Technological University, 2014

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

**NAVAL POSTGRADUATE SCHOOL
March 2021**

Approved by: Robert L. Bassett
Advisor

Ross J. Schuchard
Second Reader

W. Matthew Carlyle
Chair, Department of Operations Research

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Political leaders, military leaders, and the general public gather information from images and video to make decisions. Media can be spread instantaneously throughout the world at low cost and anonymously using social media, allowing small groups to gain powerful influence. This leaves the United States vulnerable to deception by media forgery. The problem of media forgery is not new, but the recent advances in machine learning have led to the development of DeepFakes, which are more sophisticated and difficult to identify. DeepFakes are especially dangerous because they can be used to change the identity of a person in a video or image. Detecting DeepFakes has been the focus of academic research, and several techniques have been developed and show promising results on available data sets, but the suitability of these algorithms for deployment within the Department of Defense or any other critical environment is an open question. This research provides a decisive answer to this question for a promising recent analytic that uses inconsistent head poses to detect when an image is manipulated.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1	Introduction	1
1.1	Motivation	1
1.2	Military Relevance.	2
1.3	Technical Background	3
2	Review of the Analytic	7
2.1	Review of the Analytic	7
2.2	Pinhole Camera Model	8
2.3	3D Head Pose Estimation	14
2.4	Classification	23
3	Problematic Assumptions	31
3.1	Landmark Position Estimation	31
3.2	Head Pose Estimation Problem	37
3.3	Flipped Pose	42
3.4	Generalizable Results	43
3.5	Conclusion.	46
4	Results	47
4.1	Classification Performance on the UADFV Data Set	47
4.2	Descriptions of the Data Sets	55
4.3	Train Test Splits.	57
4.4	Conclusion.	59
	List of References	61
	Initial Distribution List	65

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

Figure 1.1	The process of training autoencoders and then exchanging decoders to generate a DeepFake	4
Figure 2.1	Visualization of the analytic applied to a real and fake image. . .	8
Figure 2.2	Basic pinhole camera model.	9
Figure 2.3	Similar triangles used to derive projection into the image plane. .	11
Figure 2.4	Facial landmarks broken into central and outer subsets	15
Figure 2.5	3D average face model.	15
Figure 2.6	Figure illustrating approximation of focal width using field of view.	17
Figure 2.7	Sample frames from the UADFV data set.	24
Figure 2.8	ROC curves showing the original classification results.	26
Figure 3.1	Photos illustrating dependence of facial landmark position estimates	32
Figure 3.2	Figure depicting the progression of landmark position estimation at separate levels of the cascade.	35
Figure 3.3	Photos illustrating that the estimated position of the outer landmarks are affected by manipulations of the inner facial region.	36
Figure 3.4	Photos of Nicolas Cage illustrating the effects of the optimization algorithm terminating in a local minimum.	38
Figure 3.5	Figures illustrating transformation from world to camera coordinates.	40
Figure 3.6	3D plots illustrating the effects of the optimization algorithm terminating in a local minimum.	41
Figure 3.7	Figure illustrating the concept of a flipped head pose.	42
Figure 3.8	ROC curves showing the original classification results.	43

Figure 3.9	Photos illustrating the effect of blurring on estimated facial landmark position.	44
Figure 3.10	ROC curve showing the results of using the original model as a Nicolas Cage detector.	45
Figure 4.1	ROC curves showing the results of our experiment replicating the results presented by the author on both the UADFV and the subset of the DARPA GAN challenge data set.	48
Figure 4.2	Boxplot summarizing 30 experiments with random train test splits from the UADFV data set.	48
Figure 4.3	Barplot showing the number of frames requiring pose corrections in the UADFV data set.	50
Figure 4.4	Figure illustrating the concept of a flipped head pose.	50
Figure 4.5	3D Plots illustrating the distribution of the 3 dimensional translation vector feature in the UADFV testing set.	51
Figure 4.6	Boxplots illustrating the change in the distribution of the z -component of the translation vector feature after the poses are corrected in the UADFV data set.	52
Figure 4.7	Plots showing the distribution of the cosine distances presented by the authors is misleading.	54
Figure 4.8	ROC curve showing the effect of correcting the head poses on classification performance on the UADFV data set.	55
Figure 4.9	ROC curves presenting the results of our experiments on the DFDC and Face Forensics ++ data sets.	59

List of Tables

Table 2.1	Summary of coordinate systems	9
Table 2.2	Area under the ROC curve based on videos and frames	25
Table 4.1	Table summarizing the number of frames in each to the train test splits for our experiments.	57
Table 4.2	AUC scores for various models. The “Provided” model is pretrained on UADFV and provided by the authors. The “Uncorrected” model is trained on the training frames from Table 4.1, without the head pose correction. The “Corrected” model is also trained on the frames in Table 4.1, but uses the corrected head poses.	59

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AUC	area under the curve
Celeb-DF	Celebrity DeepFake
CNN	convolutional neural network
DARPA GAN	DARPA MediFor GAN Image/Video Challenge
DFDC	DeepFake Detection Challenge
DFDD	DeepFake Detection Data Set
DLT	Direct Linear Transformation
DOD	Department of Defense
EM	expectation maximization algorithm
FF++	Face Forensics ++
FOV	field of view
GAN	generative adversarial network
KDE	kernel density estimates
LM	Levenberg-Marquardt
mm	millimeters
NPS	Naval Postgraduate School
px	pixels
RBF	Radial Basis Function
ROC	Receiver Operating Characteristic

SVM	support vector machine
UADFV	University of Albany DeepFake Video
USN	U.S. Navy

Executive Summary

The evolution of the internet and satellite technology has closed the gap between people around the world. This connectivity leaves us vulnerable to attack in new ways such as information campaigns that target our elections, or seek to destabilize vulnerable regions around the world. While the United States homeland still maintains the geographic advantage provided by the oceans on our borders, the significance is minimized in this new era of warfare in the information domain.

These new vulnerabilities are described in the 2017 National Security Strategy as Information Statecraft, which are efforts by competitors to weaponize information to attack the values and institutions that underpin free society (Trump 2017). These efforts become more dangerous as competitors integrate information from personal and commercial sources with intelligence collection and data analytic capabilities based on artificial intelligence and machine learning to target and enhance their misinformation techniques (Trump 2017).

DeepFakes will become weapons in this new domain of Information Statecraft, giving competitors the ability to insert synthetically generated faces using deep learning techniques to change the identity of an individual in a video, or manipulate their facial features. DeepFakes may be used to target political and military leaders in an attempt to disrupt the public's confidence in our leaders. The United States must develop the ability to process large amounts of media, and identify DeepFakes reliably in order to defend against these threats. Several DeepFake classification analytics have been developed and show great promise in an academic setting, but for many the performance does not generalize to other data sets or newer generations of DeepFakes. We must develop a more complete understanding of the performance of these analytics before they will be suitable for use in the field.

Our research develops a deeper understanding of the analytic that Yang et al. (2019) propose that classifies DeepFakes using inconsistent head poses. We provide in depth analysis of the landmark estimation algorithm along with the head pose estimation procedure used by the analytic, that expose weaknesses such as overtraining on the facial structure of individuals along with systematic errors in the head pose optimization process. The problems we uncover cause us to question multiple assumptions that are fundamental to the design of

the analytic, and undermine our confidence in the generalizability of its performance. We justify our conclusions by conducting experiments on four new data sets that showcase the challenge of classifying DeepFakes in realistic scenarios.

References

Yang X, Li Y, Lyu S (2019) Exposing deep fakes using inconsistent head poses. ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 8261–8265 (IEEE).

Trump DJ (2017) National Security Strategy of the United States of America. Technical report, Executive Office of The President Washington DC Washington United States.

Acknowledgments

I would like to thank my family, particularly my parents and brother, for their support throughout my education and career. Their support has given me the opportunity to pursue my interests, leading me to a rewarding career, and their guidance has helped me make the most of it.

I would also like to thank my advising team, Dr. Robert Bassett and LTC Ross Schuchard, for all the effort supporting me through this project, along with all the teachers and mentors who have organized and facilitated great projects that have allowed me to build my skills and deepen my interest in the STEM fields throughout my academic career.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:

Introduction

“It is now undeniable that the homeland is no longer a sanctuary. America is a target, whether from terrorists seeking to attack our citizens; malicious cyber activity against personal, commercial, or government infrastructure; or political and information subversion”

—2018 National Defense Strategy

This thesis evaluates the effectiveness of classifying DeepFakes using inconsistent head poses with the goal of understanding how the analytic works on a variety of data sets, and exposing challenges that must be overcome prior to its implementation. Specifically, we identify several challenges fundamental to the operation of the analytic: the training process can create a bias causing the analytic to classify individuals based on how they appear in the training set, the optimization algorithm is not suited for head pose estimation, and estimated positions of facial landmarks are not independent. We systematically analyze the methods used within the analytic to expose the source of these problems, which may serve to as the foundation for future work to correct the analytic or influence the development of future analytics.

1.1 Motivation

DeepFakes are media that have been altered completely or in part using deep learning techniques, and can potentially be used to interfere with elections and diminish the public’s confidence in our military leaders. As the field of machine learning continues to develop, DeepFakes are becoming more sophisticated, realistic, and difficult to classify. Researchers are developing new and more intricate methods for classifying DeepFakes, but their performance outside of the academic setting must be well understood before they can be implemented by the Department of Defense (DOD). Yang et al. (2019) propose an analytic to classify DeepFakes based on inconsistencies in the estimated head poses. This thesis takes a critical look at this analytic by breaking down the methods used to process the image and

develop features for classification. We conduct tests to evaluate the analytic's performance on new data sets that reflect the challenges of classifying media posted in an online setting. Our research reveals weaknesses in the methods used to develop the features, and show that these weaknesses result in poor performance on new data sets.

1.2 Military Relevance

The evolution of the internet and satellite technology has closed the gap between people around the world. This interconnectedness has unquestionably leads to great things across the globe such as widespread access to news and education sources, and has minimized the barriers to conducting business on an international scale. These connections also leave us vulnerable to attack in new ways such as information campaigns targeting our elections as reported by Shane and Mazzetti (2018), and destabilizing vulnerable regions around the world as documented by Taub and Fisher (2018). While the United States homeland still maintains the geographic advantage provided by the oceans on our borders, the significance is minimized in this new era of warfare in the information domain.

These new vulnerabilities are described in the 2017 National Security Strategy as Information Statecraft, which are efforts by competitors to weaponize information to attack the values and institutions that underpin free society (Trump 2017). These efforts become more dangerous as competitors integrate information from personal and commercial sources with intelligence collection and data analytic capabilities based on artificial intelligence and machine learning to target and enhance their misinformation techniques (Trump 2017).

DeepFakes will become weapons in this new domain of Information Statecraft giving competitors the ability to insert synthetically generated faces using deep learning techniques to change the identity of an individual in a video, or manipulate their facial features. DeepFakes may be used to target political and military leaders in an attempt to disrupt elections or diminish the public's confidence in military leaders. The United States must develop the ability to process large amounts of media, and identify DeepFakes reliably in order to defend against these threats.

Several DeepFake classification analytics have been developed and show great promise in an academic setting, but for many the performance does not generalize to other data sets or

newer generations of DeepFakes. We must develop a more complete understanding of the performance of these analytics before they will be suitable for use in the field.

1.3 Technical Background

This section provides an introduction to the term DeepFake, and then focuses our attention on the face swap style DeepFake that Yang et al. (2019) designed the analytic to classify. We discuss the steps of creating a face swap DeepFake using an autoencoder as well as present common post-processing techniques used to finish the creation process. This section concludes with a discussion of different DeepFake classification techniques demonstrating various strategies for identifying DeepFakes. These introductions provide context for the review of the analytic in the following sections.

1.3.1 Introduction to DeepFakes

In recent years, applications of deep learning to image manipulation have led to the emergence of *DeepFakes*, which lower technological barriers to creating high-quality manipulations. One of the most common tools for generating DeepFakes is the autoencoder-based face swap; see for example Petrov et al. (2020); Anonymous (2020a); Noon van der Silk (2019); Gareth Dunstone (2021); Anonymous (2020b). Though the term “DeepFake” is sometimes used as a universal descriptor for any synthetic media generated wholly or partially with deep learning, in this thesis we use it to describe images which are created with an autoencoder-based face swap. Though minor differences exist between implementations, we outline the generation of autoencoder-based DeepFakes in this section.

The steps required to swap the face from a source actor and place it on the body of a target actor using an autoencoder are as follows. First, images must be preprocessed by extracting faces from them. This step requires facial detection, alignment, and segmentation steps. Face detection itself is a mature technology, and many tools exist for automating it, with the result that faces can be detected across multiple images quickly. Examples include the `get_frontal_face_detector` method from the `dlib` package King (2009), and the `face_cascade` classifier from the `opencv` package Bradski (2000). Facial alignment can also be conducted quickly, because estimating the optimal affine transforms between two sets of 2D points is straightforward. Li et al. (2020) proposes an automated method for

segmentation by selecting the facial region in the convex hull of a set of facial landmarks, though this process yields poor results when the subject has hair or accessories obscuring part of the face, so it can require manual intervention.

After preprocessing, an autoencoder network is adversarially trained on the aligned faces, where a single encoder network is used for both actors but separate decoders are used for the source and target actor. After training, the swap is performed by encoding the features of the target actor, and then decoding the features with the source decoder. Figure 1.1 illustrates the process of training and then swapping decoders in order to create DeepFakes.

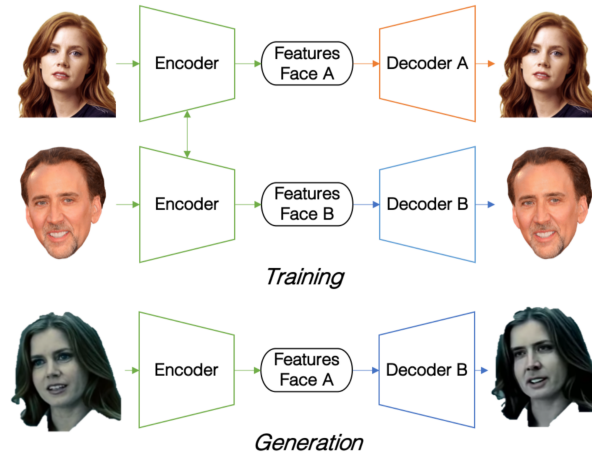


Figure 1.1. The process of training autoencoders and then exchanging decoders to generate a DeepFake. Source: Güera and Delp (2018).

Postprocessing is an important final step to eliminate visual artifacts where the swapped inner region meets the outer portion of the face. Common techniques for doing so include Poisson blending, where the DeepFake's color gradient is taken to be a linear combination of the original and swapped portions of the face, for details see Zhang et al. (2020). Petrov et al. (2020) uses an alternate technique of blurring the entire face (using Gaussian blur, for example) and then apply a super-resolution network to reconstruct a realistic face from the blurred one. An impressive property of this approach is that a properly trained super-resolution network can reconstruct minor details, such as moles or wrinkles, on the generated face.

1.3.2 Classification Techniques

Motivated by the convincing nature of modern DeepFakes, many methods have been proposed to detect them from authentic images. These methods can be partitioned by the features they use to detect the manipulated image. Though not specific to DeepFake detection, methods which are designed to detect image splicing, where a part of a source image is placed into a target image, can also be applied for this purpose. For example, Nguyen et al. (2019a) trains an autoencoder to partition a face into spliced and unmanipulated regions. Similarly, Huh et al. (2018) leverages image metadata to detect inconsistencies and generate a splice mask for various regions of an image. Wu et al. (2019) take a different perspective by formulating a hypothesis test to detect splicing at each pixel, using Z-scores constructed for various neighborhoods of the pixel as a test statistic. Zhao et al. (2020) detect spliced regions using steganalysis features unique to the processing of individual cameras by comparing these features in two different regions of an image.

Another large class of techniques trains a classifier to detect images generated by either convolutional or generative adversarial networks. For example, Guarnera et al. (2020) uses an expectation maximization algorithm (EM) to detect patterns of correlation among pixel neighborhoods. The correlation patterns are used to classify whether an image was generated by a convolutional network. Similarly, Wang et al. (2020) shows that images generated by a large suite of convolution-based generators (including DeepFakes) can be detected by training a ResNet model on images from only a single generator, which suggests that these images exhibit common structure. Considering GANs instead of convolutional networks, Yu et al. (2019) considers the residual of an autoencoder-based image reconstruction as a “GAN fingerprint” and uses it to classify GAN-generated from authentic images. Along similar lines, Zhang et al. (2019) uses a discrete Fourier transform to detect artifacts of the upsampling procedure used by many GAN-based image generators.

Some methods for classifying authentic from manipulated images use deep learning, but do not focus on specific features of the DeepFake generation process. Instead, they train the model using authentic and manipulated images and rely on the model to develop its own features to classify the images. For example, Afchar et al. (2018) uses a lower number of layers in its architecture, and Nguyen et al. (2019b), who uses a capsule network as opposed to more traditional convolutional layers.

A final class of techniques uses features derived from knowledge that the image contains a human face. Li et al. (2018a) classifies manipulated videos by noting the lack of natural blinking patterns. More recent DeepFakes circumvent this by including images with a blinking subject in the training data. Matern et al. (2019) detect manipulated images using inconsistencies in the eye and teeth regions, where lack of detail is common. Agarwal et al. (2019) consider the problem of classifying videos that contain a certain individual, such as a public figure. The analytic captures the individual’s facial mannerisms using a correlation matrix of facial actions, such as eyebrow raising and chin raising, across frames in a video, and uses this correlation matrix as features in an SVM to determine authentic from manipulated videos. In follow-up work, Agarwal et al. (2020) uses a convolutional network to capture facial mannerisms, which avoids the labor-intensive process of hand-crafting the correlation-based features. Finally, Ciftci et al. (2020) estimates blood volume changes that occur due to a rhythmic heartbeat, and uses this information to distinguish authentic from synthetic videos. The analytic we consider in this paper falls in this final class of techniques that leverage the fact that the image contains a human subject, because its method for distinguishing real from manipulated images utilizes the structure of the human face.

CHAPTER 2:

Review of the Analytic

In this chapter we give an overview of how the analytic works, and provide some introductions to the methods and tools the analytic uses to process images for classification. The first section introduces how the analytic proposes to classify deepfakes. This is followed by a section dedicated to the development of an equation to approximate the action of the camera from the pinhole camera model, and explains the approximations that are necessary for the analytic to apply the equation. The third section explains how the facial landmarks are approximated and how they are used to develop the head pose estimates. The chapter concludes with a discussion of how the analytic is trained, including a subsection on the support vector machine classifier.

2.1 Review of the Analytic

The goal of this section is to provide a broad overview of the analytic to establish the proper context for the following sections. This includes a demonstration of how the analytic is trying to distinguish between real and DeepFake images. We provide a basic summary of how the analytic functions and discuss why we expect this analytic could succeed.

The analytic takes advantage of the fact that only the inner region of the face is manipulated in an autoencoder based face swap by comparing the head pose estimated based on central landmarks to the head pose estimated based on the full set of landmarks. Yang et al. (2019) claim that if the image is manipulated, only the central facial landmarks are affected and the two pose estimates may differ, while in a real image the two poses should align. A support vector machine classifier is trained using the head pose features to make the classification.

Figure 2.1 is included in the original paper by Yang et al. (2019) to illustrate the authors' claim that inconsistent head poses can be used to classify DeepFakes even if these inconsistencies are not distinguishable by the human eye. Image (a) is a real image and the red dots are the estimated position of the central facial landmarks. The red vector represents the orientation of the head relative to the camera. Image (b) is the same real image where the blue dots are the estimated positions of all the facial landmarks, and the blue vector

represents the orientation of the head relative to the camera. Image (c) shows a comparison of the vectors from images (a) and (b), and they closely align indicating that the image is not manipulated. The dots and vectors in images (d), (e), and (f) represent the same features as in the previous images, but the image is manipulated. Image (f) shows a comparison of the two orientation vectors and they do not align, indicating that the image is a DeepFake. In the following sections we take a closer look at each step that is required in the execution of the analytic.

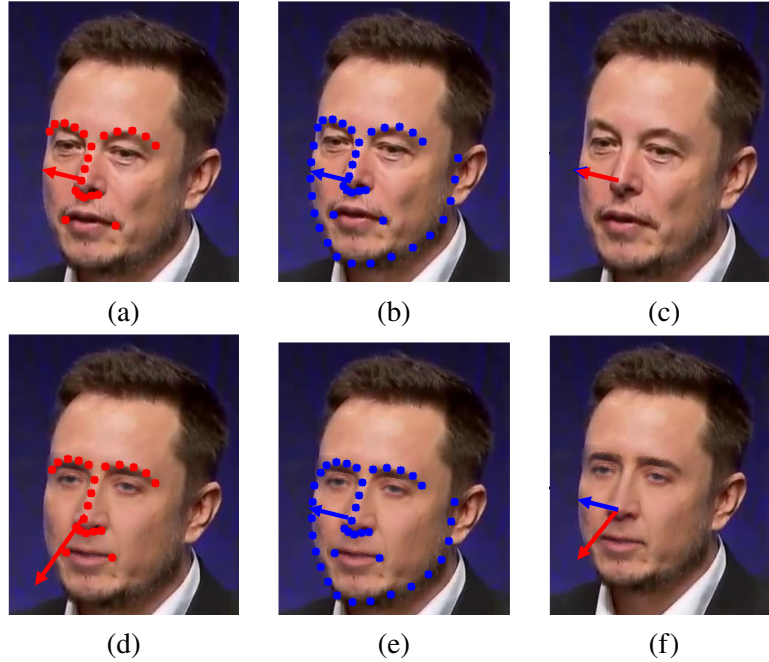


Figure 2.1. Visualization of the analytic applied to a real and fake image. Source: Yang et al. (2019).

2.2 Pinhole Camera Model

In this section, we derive the equation the analytic uses to approximate the action of the camera, which is based on the pinhole camera model. We also present the pinhole camera model in detail to understand its limitations, and use a simple example to illustrate the use of the model. Hata and Savarese (2015) provide an excellent introduction to the pinhole camera model along with a review of the simple lens model to provide a better understanding of the development of the model along with its limitations.

Coordinate Systems

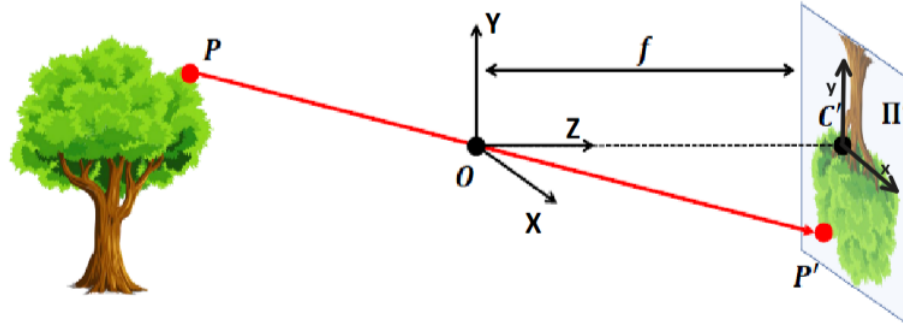


Figure 2.2. Basic pinhole camera model. Source: Hata and Savarese (2015).

Name	Origin	Units	Notation
World	Center of Face	mm	P_{wc}
Camera	Aperture of Camera O	mm	P_c
Image	Center of Image Plane C'	mm	P'_i
Pixel	Lower Left Corner of Image Plane	px	P'_{px}

Table 2.1. Summary of coordinate systems

Four separate coordinate systems are used to describe points in the formulation of this problem and are summarized in Table 2.1 and illustrated in Figure 2.2. The three-dimensional camera coordinate system is centered at the aperture of the camera O and is measured in millimeters (mm). The world coordinate system is also three-dimensional and measured in mm, but is centered on the subject's face. The two-dimensional image coordinate system is on the image plane and is centered at C' , where the optical axis intersects the image plane, and is measured in mm. The pixel coordinate system is a two-dimensional coordinate system on the image plane, but its origin is on the lower left hand corner of the image and is measured in pixels (px).

The camera and image coordinate systems are depicted in the Figure 2.2 above along with a point P in 3 dimensions which is projected to the point P' in the image plane. The distance f is the focal length of the camera, and is the distance between the camera aperture and the image sensor. In Figure 2.2 the focal length is shown along the optical axis between the aperture of the camera O and the image plane and is measured in mm. We now establish notation that we use throughout to demonstrate the pinhole camera model, and begin with the point P measured in world coordinates given by equation (2.5).

$$P_{wc} = \begin{bmatrix} U \\ V \\ W \end{bmatrix}. \quad (2.1)$$

Similarly, we denote the 3D point P in camera coordinates as

$$P_c = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}. \quad (2.2)$$

The projection of the point P onto the 2-dimensional image plane is represented by P' . The point P' can be measured using either the pixel coordinate system or the image coordinate system. The point P' represented in image coordinates is given by equation (2.3)

$$P'_i = \begin{bmatrix} x \\ y \end{bmatrix}. \quad (2.3)$$

Similarly, the point P' represented in pixel coordinates as

$$P'_{px} = \begin{bmatrix} p_x \\ p_y \end{bmatrix}. \quad (2.4)$$

Projecting a Point in World Coordinates onto the Image Plane

In this section we use the pinhole camera model to project a 3D point in world coordinates to a 2D point in pixel coordinates. This projection process is an important prerequisite for the pose estimation portion of the analytic because it allows us to evaluate a pose estimate. We evaluate a pose estimate by using the rotation matrix R and translation vector $\vec{t}$ to project the points from the average face model in world coordinates into the image plane in pixel coordinates for comparison to the position of the facial landmarks estimated from the image.

The first step is to transform an arbitrary point P from world coordinates to camera coordinates. This requires rotating the coordinate system with a rotation matrix R and translating the coordinate system with $\vec{t}$, so that the camera lens lies at the origin. We accomplish this transformation by using equation (2.5).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \vec{t} \quad (2.5)$$

We now have the position of point P measured in reference to the camera aperture, and are ready to project the point into the image plane. Figure 2.3 provides a two-dimensional representation that highlights the relationship between the X dimension of the point P in world coordinates and the x dimension of the point P' in image coordinates.

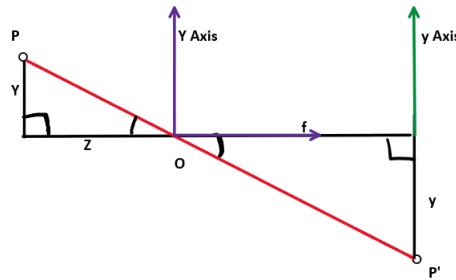


Figure 2.3. Similar triangles used to derive projection from camera coordinates to the image plane.

We develop equations 2.6 and 2.7 from the similar triangles shown in yellow in Figure 2.3 to determine the position of the point P' with respect to the center of the image plane in mm.

$$x = f \frac{X}{Z} \quad (2.6)$$

$$y = f \frac{Y}{Z} \quad (2.7)$$

Converting from Image Coordinates to Pixel Coordinates

The conversion from image coordinates to pixel coordinates requires a conversion from millimeters to pixels. Since pixels may not be square, separate conversion factors k and l in pixels per millimeter are used to convert the image coordinates. After the conversion of the x and y coordinates to pixels, they are shifted using c_x and c_y to reflect the change of the origin from the center of the image plane to the lower left corner of the image. The resulting conversions are shown in equations (2.8) and (2.9).

$$p_x = xk + c_x \quad (2.8)$$

$$p_y = yl + c_y \quad (2.9)$$

Mapping Points from Camera Coordinate System to Points in the Image Plane in Pixel Coordinates

In previous sections we show step by step how to project a point in the camera coordinate system to a point on the image plane measured in the pixel coordinate system. The projection is necessary in order to compare the positions of the projected points to the estimated positions of the facial landmarks gathered from the image. In this section we bring these steps together and represent them in a matrix form to simplify the implementation of the projections in later work.

Combining equations (2.4), (2.6), (2.7), (2.8), and (2.9) we get a representation of the mapping of the point P_c to the point P'_{px} .

$$P'_{px} = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} fk \frac{X}{Z} + c_x \\ fl \frac{Y}{Z} + c_y \end{bmatrix} \quad (2.10)$$

We can see from equation (2.10) that the operation is not linear because of the division by Z , so the operation cannot be represented as a matrix vector product. We can rewrite the equation utilizing homogeneous coordinates to develop a linear representation in matrix form.

$$P^{h'}_{px} = \begin{bmatrix} sp_x \\ sp_y \\ s \end{bmatrix} = \begin{bmatrix} fkX + c_xZ \\ flY + c_yZ \\ Z \end{bmatrix} \quad (2.11)$$

The same operation can now be represented in matrix form.

$$P^{h'}_{px} = s \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} fk & 0 & c_x \\ 0 & fl & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.12)$$

Although there is only a single focal length for the camera in a given image, substitutions are made in equation (2.12) $f_x = fk$ and $f_y = fl$ to reduce the number of parameters in the model, and are referred to as the focal length in the x and y directions, respectively. This leaves us with equation (2.13) as our final equation we use to approximate the action of the camera. We are now ready to discuss how the equation is used to develop 3D head pose estimates that are ultimately used to develop features for classification.

$$P'_h = s \begin{bmatrix} p_x \\ p_y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.13)$$

2.3 3D Head Pose Estimation

We now show how the positions of the facial landmarks are determined in the image, as well as how the landmark positions are approximated in world coordinates using the average face model. This is followed by a definition of the 3D head pose and an explanation of how equation 2.13, developed in the previous section, is used to formulate the optimization problem for estimating the head pose. The final subsection provides an introduction to the Levenberg-Marquardt algorithm used to solve the head pose estimation problem.

2.3.1 Estimating the Position of the Landmarks in the Image

The optimality of the head pose is based on the squared distance between the projected points from the head pose and the estimated landmark positions, making the landmark position estimation a critical part of the analytic. The landmark estimation process requires the isolation of the facial region followed by the estimation of the landmark positions. The analytic uses the `get_frontal_face_detector` method from the `dlib` package to isolate the facial region. Next the landmark positions are estimated by passing the facial region of the image into `dlib` package's default `dlib.shape_predictor` for detecting 68 facial landmarks, 38 of which are used by the analytic. The subset of these landmarks shown in white in Figure 2.4 are discarded because their positions aren't fixed with respect to the position and orientation of the head. The landmarks shown in red lie in the manipulated region of a face swap are categorized as central facial landmarks. The remaining landmarks shown in blue are along the outer region of the face, and it is assumed that their positions are unaffected by a face swap. The analytic utilizes these landmarks in two groups, the central landmarks group which are shown in red, and all landmarks group which include the landmarks shown in blue as well as the ones shown in red.

2.3.2 Estimating Landmark Position in World Coordinates

The position of the facial landmarks is unknown in both the camera and world coordinates, so approximations must be made to determine the 3D head pose. The analytic estimates the landmark positions in world coordinates by drawing them from a 3D model provided by Baltrusaitis et al. (2018), which is shown in Figure 2.5. This model represents the facial landmark positions of an average face. The analytic projects the landmark positions into the image plane using the rotation matrix R and translation vector $\vec{t}$ that define the 3D head

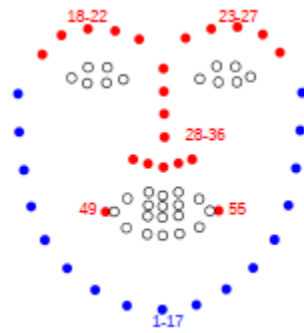


Figure 2.4. Facial landmarks detected using the `get_frontal_face_detector` package from `dlib` Source: Yang et al. (2019).

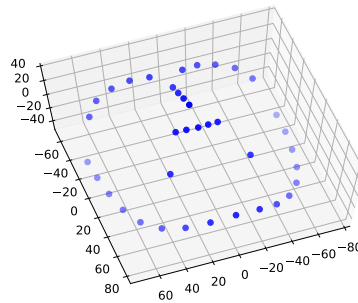


Figure 2.5. 3D average face model used as estimates in world coordinates. Source: Baltrusaitis et al. (2018).

pose, and then compares the positions of the projected points to the landmark positions estimated from the image to evaluate the fit of the selected head pose.

2.3.3 3D Head Pose Defined

For the purposes of this analytic, the 3D head pose corresponds to the rotation and translation of the world coordinates centered on the subject's face to the corresponding camera coordinates centered at the aperture of the camera. Specifically, let $[U, V, W]^T$ represent a facial landmark in world coordinates, and $[X, Y, Z]^T$ be its corresponding camera coordinates. The transformation between the two coordinate systems is given in equation (2.14) below.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \vec{t} \quad (2.14)$$

where R is the 3×3 rotation matrix with R^{-1} giving the orientation of the head relative to the camera, and $\vec{t}$ is the 3×1 translation vector giving the position of the head relative to the camera. The rotation matrix and translation vector are used to define the 3D head pose, and are used to create input features for the classifier.

2.3.4 3D Head Pose Estimation

Here we formulate the optimization problem used to estimate the 3D head pose. The estimation of the head pose is a fundamental part of the analytic because the head poses are used to develop features to classify the image. The formulation of the optimization problem is based on equation (2.15) developed in section 2.2 from the pinhole camera model, which relates the positions of the facial landmarks in camera coordinates to the landmark positions in the image plane.

$$s \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \quad (2.15)$$

where (x, y) is the position of the facial landmark in the image plane, f_x and f_y correspond to the focal lengths in the x and y directions, (c_x, c_y) is the optical center, and s is an unknown scaling factor. Since the camera used to capture the image is unknown, the intrinsic parameters of the camera must be approximated.

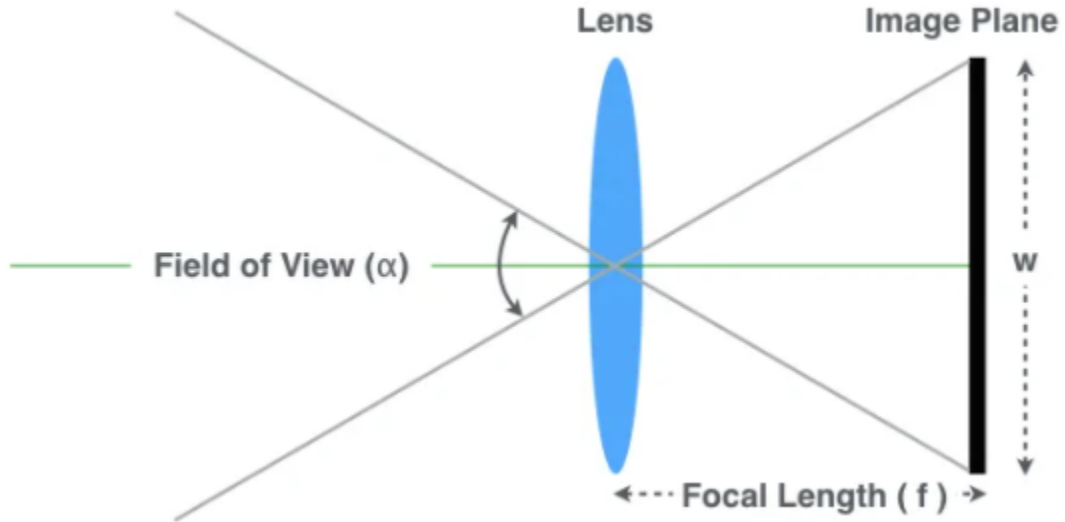


Figure 2.6. Figure illustrating approximation of focal width using field of view. Source: Satya Mallick (2016).

Figure 2.6 shows the relationship between the field of view (FOV) of the camera and its focal length. Satya Mallick (2016) provides an explanation for equation (2.16) that relates the FOV to the focal length f using trigonometry.

$$f = \frac{w}{2} \cot \frac{\alpha}{2} \quad (2.16)$$

The approximation of f by half the image width in pixels is valid if the FOV is close to 50 degrees. The FOV of a standard lens that is used to replicate the view natural to a human observer typically ranges between 40 and 60 degrees, while wide angle and telephoto lenses can provide fields of view higher than 180 degrees to lower than 1 degree.

We assume proper alignment of the image sensor by the use of the optical center as the image center and ignoring the possibility of skewness. An image is skewed when the angle between the image plane and the optical axis is not exactly 90 degrees. Skewness and misalignment may occur due to errors in the sensor manufacturing process or damage to the camera.

The pinhole camera model does not account for lens distortion. Kannala and Brandt (2006) present more detailed camera models that account for factors such as skewness and lens distortion, but these factors often cannot be reliably determined from a single image so the simple pinhole camera model is used.

With approximations made for the intrinsic parameters of the camera, the head pose is estimated by choosing rotation matrix R and translation vector $\vec{t}$ to transform the landmark positions in world coordinates shown in Figure 2.5 to minimize the error when the points are projected using equation (2.15). This is done by solving the following optimization problem using n facial landmarks (2.17).

$$\min_{R, \vec{t}, s} \sum_{i=1}^n \left\| s \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} - \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left(R \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix} + \vec{t} \right) \right\|^2 \quad (2.17)$$

This problem can be solved efficiently using the Levenberg-Marquardt algorithm (Bradski 2000). The resulting rotation matrix R is the rotation of the camera relative to the world coordinates, and the orientation of the head relative to the camera is obtained by R^{-1} .

For each image, two head poses are estimated. One is based on all of the facial landmarks and the second is based exclusively on the central facial landmarks. The resulting rotation matrix and translation vector for the head pose based on all the facial landmarks are denoted by R_a and $\vec{t}_a$, while the rotation matrix and translation vector resulting from the head pose based on the central facial landmarks are denoted R_c and $\vec{t}_c$. We see how these rotation matrices and translation vectors are used to develop features for classification in section 2.4, next we take a closer look at the Levenberg-Marquardt algorithm used to solve problem (2.17).

2.3.5 Levenberg-Marquardt Algorithm

The Levenberg-Marquardt (LM) algorithm is a specialized optimization algorithm designed to solve nonlinear least squares minimization problems (Marquardt 1963), and is implemented in the analytic to find the optimal head pose estimate. Ranganathan (2004) provides an introduction to the algorithm and a review of the Gauss-Newton and gradient descent

optimization methods for context. We provide a similar approach in this section which is broken into four subsections. First we provide a brief introduction to the Hessian and Jacobian matrices, then discuss the form and advantage of working with least squares problems. We then review the gradient descent and Gauss-Newton methods, and we finish with the development of the Levenberg-Marquardt algorithm.

Introduction of the Hessian and Jacobian Matrices

For a scalar valued function f , we have the gradient ∇f and Hessian matrix $\nabla^2 f$ providing indications of curvature and slope. The Hessian matrix is the square matrix of the second order partial derivatives of f and is given by equation (2.18).

$$\nabla^2 f = \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2} & \frac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_1 \partial x_n} \\ \frac{\partial^2 f}{\partial x_2 \partial x_1} & \frac{\partial^2 f}{\partial x_2^2} & \cdots & \frac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial^2 f}{\partial x_n \partial x_1} & \frac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \frac{\partial^2 f}{\partial x_n^2} \end{bmatrix} \quad (2.18)$$

Similarly, for a vector valued function $[f_1(x), f_2(x), \dots, f_m(x)]^T$ the Jacobian is the matrix of all first order partial derivatives and is given by equation (2.19).

$$\mathbf{J} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix} \quad (2.19)$$

Both of these matrices are fundamental tools in many optimization methods, but $\nabla^2 f$ can be difficult to compute if x has many components. In the following sections we show how an approximation to the Hessian can be made using the Jacobian in the case of least squares problems.

Least Squares Problems

We now introduce the form of the Least Squares problem and show how $\nabla^2 f$ can be approximated using $\mathbf{J}$, which removes the computational burden of calculating $\nabla^2 f$. Least squares problems take the form shown in equation (2.20).

$$f(x) = \frac{1}{2} \sum_{j=1}^m r_j^2(x) \quad (2.20)$$

In equation (2.20), $x = [x_1, x_2, \dots, x_n]^T$ is a vector, and each r_j is a function from $\mathbb{R}^n$ to $\mathbb{R}$. The r_j are referred to as residuals and it is assumed that $m \geq n$.

This form can be re-written to make it easier to work with by representing f as a residual vector $r: \mathbb{R}^n \rightarrow \mathbb{R}^m$ defined by equation (2.21).

$$r(x) = [r_1(x), r_2(x), \dots, r_m(x)]^T \quad (2.21)$$

This residual vector allows us to represent f as shown in equation (2.22).

$$f(x) = \frac{1}{2} \|r(x)\|^2 \quad (2.22)$$

Now the derivatives of f can be easily written in terms of the Jacobian $\mathbf{J}$ of r with respect to x , and are shown in equations (2.23) and (2.24)

$$\nabla f(x) = \sum_{j=1}^m r_j(x) \nabla r_j(x) = \mathbf{J}(x)^T r(x) \quad (2.23)$$

$$\nabla^2 f(x) = \mathbf{J}(x)^T \mathbf{J}(x) + \sum_{j=1}^m r_j(x) \nabla^2 r_j(x) \quad (2.24)$$

The appealing property of least squares problems is that the Hessian $\nabla^2 f(x)$ can be determined easily with $\mathbf{J}$ if the r_j s can be approximated as linear functions making the $\nabla^2 r_j(x)$

small, or if the r_j s themselves are small. These approximations reduce equation (2.24) to the equation shown in (2.25).

$$\nabla^2 f(x) = \mathbf{J}(x)^T \mathbf{J}(x) \quad (2.25)$$

Gradient Descent and Gauss-Newton Methods

Gradient descent and the Gauss-Newton methods are simple and intuitive methods for finding the minima of a function, but they both have disadvantages. In this section we briefly discuss the two methods, and in the following section we show how the methods combine to create the Levenberg-Marquardt algorithm.

Gradient descent is the simplest and most intuitive method to find the minima of a function, where steps are taken in the direction of steepest descent given by the opposite of the direction of the gradient. The update rule is shown in equation (2.26), where λ is a scalar representing step length and i is the iteration counter.

$$x_{i+1} = x_i - \lambda \nabla f \quad (2.26)$$

Gradient descent fails to account for curvature of the objective function, as quantified by the use of the Hessian. The Gauss-Newton method forms better local approximations to f by using curvature.

The Gauss-Newton method improves upon the simple gradient descent method by incorporating curvature as well as gradient information by utilizing the second derivatives. The update rule for the Gauss-Newton method is based on solving the equation $\nabla f(x) = 0$ by expanding ∇f using a Taylor series around x_0 , resulting in (2.27)

$$\nabla f(x) = \nabla f(x_0) + (x - x_0)^T \nabla^2 f(x_0) + \text{higher order terms of } (x - x_0) \quad (2.27)$$

The update rule is found by neglecting the higher order terms and solving for the minimum of f by setting $\nabla f(x) = 0$ giving us the following rule (2.28)

$$x_{i+1} = x_i - \left(\nabla^2 f(x_i) \right)^{-1} \nabla f(x_i) \quad (2.28)$$

where x_0 has been replaced by x_i and x by x_{i+1} .

Newton's method is based on a quadratic approximation of f since the higher order terms are ignored in the Taylor series approximation, so the Hessian matrix does not need to be evaluated exactly. An approximation of the Hessian is sufficient. The advantage of the Gauss-Newton method is its rapid convergence, but the rate of convergence is sensitive to the linearity around the starting location. The advantage of Gauss-Newton method over gradient descent is that, under certain conditions, it has a faster rate of convergence (Adby 2013).

Levenberg-Marquardt Algorithm

Levenberg observed that the simple gradient descent method and Gauss-Newton methods have complementary advantages, and developed an algorithm to blend the two which is accomplished by the following update rule shown in equation (2.29).

$$x_{i+1} = x_i - (H + \lambda I)^{-1} \nabla f(x_i) \quad (2.29)$$

where H is the Hessian matrix evaluated at x_i , and λ is the factor balancing the effects of gradient descent and Gauss-Newton methods. The algorithm is implemented using the following steps:

1. Perform an update using equation (2.29).
2. Evaluate the error at the new parameter vector.
3. If the error has increased as a result of the update, then reset weights to their previous values and increase λ by a factor of 10 and return to step (1).
4. If the error has decreased as a result of the update, then keep the weights at their new values and decrease λ by a factor of 10.

This algorithm does provide a balance between the two methods, but if λ gets too large the effect of gradient descent dominates and the Hessian matrix is not used at all. Marquardt

developed a modification to the update rule given by equation (2.30), which scales the component of the gradient based on the second derivatives.

$$x_{i+1} = x_i - (H + \lambda \text{diag}[H])^{-1} \nabla f(x_i) \quad (2.30)$$

This is the Levenberg-Marquardt algorithm, which scales the gradient based on the curvature. These modifications result better performance because the step size is appropriately scaled by the curvature of f .

The analytic implements the Levenberg-Marquardt algorithm to efficiently solve for the two head pose estimates required to develop the features for classification. To this point we have reviewed the methods required to develop these head pose estimates from an image. We proceed in the following section with a more focused discussion of how the original model was trained by taking a closer look at the data used as well as the classifier and features that were selected for implementation in the analytic.

2.4 Classification

We conclude this chapter by following the development, along with the justifications and results that Yang et al. (2019) present in the original paper. This discussion provides context for the review of fundamental assumptions and claims in chapter 3. We start by reviewing the data, and then follow with a presentation of the training process including discussions on feature selection and opt outs and special considerations. We have also included a review of the results presented by the authors and a summary of their interpretations. We conclude the chapter with a brief mathematical introduction to the classifier implemented in the analytic.

2.4.1 The Data

Yang et al. (2019) developed the analytic using the University of Albany DeepFake Video (UADFV) data set which they created in their previous work (Li et al. 2018b). The data set consists of 49 real videos and each one has a corresponding DeepFake video with the face of Nicolas Cage swapped. The average length of the videos is 11.14 seconds and the typical resolution is 294 x 500 pixels. Three sample frames are provided in Figure 2.7, and you can see that as individual frames the fakes can be convincing.



Figure 2.7. Sample frames from the UADFV data set. Source: Li et al. (2018b).

The data set are exclusively videos, and some DeepFakes are easily classified by eye due to fluctuations in the position of the synthesized region of the face from one frame to the next. This is especially apparent when the actor rotates their head. While the videos may not be convincing to the eye, the analytic processes each frame individually, so it does not pick up on these fluctuations the way a viewer would.

The performance of the analytic was validated using a subset of images from the DARPA MediFor GAN Image/Video Challenge (DARPA GAN) data set consisting of 241 real images and 252 DeepFake images. Overall the quality was much lower in the DARPA GAN data set, many of the fakes are blurry and several images appear to be subsequent frames in a video making the data set effectively much smaller.

2.4.2 Training the Classifier

The analytic estimates two head poses from each image, one based on all of the facial landmarks and a second based on only the central facial landmarks. These two head pose estimates result rotation matrices R_a and R_c along with translation vectors $\vec{t}_a$ and $\vec{t}_c$. The analytic creates the head orientation vector $\vec{v}$ and the Rodrigues' vector $\vec{r}$ from the rotation matrix as alternative representations of the orientation of the head. The head orientation vectors are unit vectors obtained by $\vec{v}_a = R_a^T \vec{w}$ and $\vec{v}_c = R_c^T \vec{w}$, where $\vec{w} = [0, 0, 1]^T$ is the unit vector in the direction of z -axis in world coordinates which points outward from the center

of the face. The head orientation vectors are shown in Figure 2.1 as the vectors pointing from the nose of each individual. The Rodrigues vector $\vec{r}$ is the same rotation represented by the rotation matrix R , but in the form of a three-dimensional vector. Piña (2011) provides an introduction to representing rotations using the Rodrigues vector.

Yang et al. (2019) tested the analytic on several combinations of these features to determine the best performance with a Support Vector Machine (SVM) classifier. Table 2.2 summarizes the results of the tests, and shows that as expected the performance improves as more descriptive features are provided to the SVM. The difference of the head orientation vectors only capture the rotation of the z -axis, so performance improves as the orientation is described fully using the rotation matrix or Rodrigues vector. The flattened difference between the rotation matrices $R_a - R_c$ along with the difference between the translation vectors $\vec{t}_a - \vec{t}_c$ are the best performing set of features and were selected for implementation in the analytic.

Features	Frame	Video
$\vec{v}_a - \vec{v}_c$	0.738	0.888
$\vec{r}_a - \vec{r}_c$	0.798	0.898
$R_a - R_c$	0.853	0.913
$(\vec{v}_a - \vec{v}_c) \& (\vec{t}_a - \vec{t}_c)$	0.840	0.949
$(\vec{r}_a - \vec{r}_c) \& (\vec{t}_a - \vec{t}_c)$	0.866	0.954
$(R_a - R_c) \& (\vec{t}_a - \vec{t}_c)$	0.840	0.949

Table 2.2. Area under the ROC curve based on videos and frames.
Source: Yang et al. (2019).

2.4.3 Opt Outs and Special Considerations

The analytic processes videos by evaluating each frame and returning a cumulative probability that the image is a DeepFake by averaging of probabilities of each individual frame. The analytic is not always capable of analyzing every single frame in a video, because of its dependence on the frontal face detector and shape predictor from `dlib`. If the actor in the frame is not facing the camera, the frontal face detector is not able detect a face and the

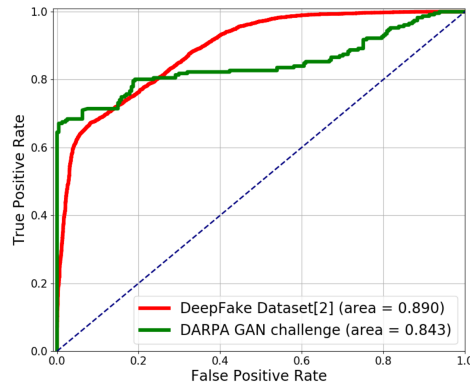


Figure 2.8. ROC curves showing the original classification results.
Source: Yang et al. (2019).

analytic opts out and the frame is ignored. If a face is detected but no landmark positions can be determined, the frame is classified as fake with probability 1. These factors show the strong dependence of the analytic on the performance on the frontal face detector and shape predictor. If these methods fail or malfunction, the performance of the analytic is not reliable.

2.4.4 Results

The analytic shows promising results on both the UADFV and DARPA GAN data sets, and we provide a summary of how Yang et al. (2019) performed the tests and interpreted their results. They first created a training and testing set from the UADFV data set. The frames were extracted from the videos and labeled real or fake based on the videos they were drawn from. Then a support vector machine (SVM) with a radial basis function (RBF) kernel was trained on the frames from the training set. The authors tested the analytic by classifying the individual frames in the testing set, and conducted a second test by classifying the images in the DARPA GAN data set. 2.8 summarizes the results as receiver operating characteristic (ROC) curves, which are drawn from the original paper. The authors claim that the analytic performs worse on the DARPA GAN images because the synthesized portions of the images are often blurry making it difficult for the landmark positions to be determined accurately.

2.4.5 Support Vector Machine

Previous sections demonstrate how the analytic was developed and trained, next we give a brief introduction to the theoretical concepts behind a support vector machine classifier. Whitaker (2020) provides an introduction to the foundational concepts. We focus our attention on the development of radial basis functions, and begin by working with a simple case where we build a linear decision boundary between classes that are linearly separable then extend that method to include the non-separable case. Followed by an introduction to the concept of basis functions and using them to build non-linear decision boundaries. We finish with a discussion of radial basis functions, since they are used in the support vector machine classifier for this analytic.

Linear Decision Boundaries

In its most basic form, the support vector machine is used to create a linear decision boundary between two classes. As an example, consider a set of n data points with two predictors x_1 and x_2 and $y_i \in \{-1, 1\}$ indicating class membership that are linearly separable. The linear decision boundary takes the form of equation (2.31).

$$b_0 + b_1x_1 + b_2x_2 \tag{2.31}$$

We define the optimal choice for the linear decision boundary as the one that maximizes the margin between the observations and the boundary. For convenience we define the vector $\vec{b} = [b_0, b_1, b_2]$. The coefficients defining a linear decision boundary are not unique so we scale the coefficients in problem (2.32) by $\|\vec{b}\|$.

$$\begin{aligned} \max_{b_0, b_1, b_2} \quad & M \\ \text{s.t.} \quad & \frac{y_i(b_0 + b_1x_{1i} + b_2x_{2i})}{\|\vec{b}\|} \geq M \end{aligned} \tag{2.32}$$

Since it doesn't matter what $\|\vec{b}\|$ is, we set $\|\vec{b}\| = \frac{1}{M}$ to transform it into a quadratic minimization problem shown in problem (2.33).

$$\begin{aligned}
& \min_{b_0, b_1, b_2} \quad \|\vec{b}\|^2 \\
& \text{s.t.} \quad y_i(b_0 + b_1x_{1i} + b_2x_{2i}) \geq 1
\end{aligned} \tag{2.33}$$

This formulation can be generalized to deal with the non-separable case by replacing the constraint with a penalty for points on the wrong side of the boundary. The penalty is equal to a cost C times the distance from the boundary as shown in problem (2.34).

$$\min_{b_0, b_1, b_2} \quad \|\vec{b}\|^2 + C \sum \max(0, 1 - y_i(b_0 + b_1x_{1i} + b_2x_{2i})) \tag{2.34}$$

Non-Linear Decision Boundaries

Non-linear boundaries can be created by extending the basis. The basis is the collection of real-valued functions of inputs that are used to generate features that are to be combined in a linear function. We demonstrate this by defining a quadratic polynomial basis for our example. Let $h_1(\vec{x}_i) = x_{1i}$, $h_2(\vec{x}_i) = x_{2i}$, $h_3(\vec{x}_i) = x_{1i}x_{2i}$, $h_4(\vec{x}_i) = x_{1i}^2$, and $h_5(\vec{x}_i) = x_{2i}^2$. The boundary is determined by solving problem (2.35)

$$\min_{b_0, b_1, \dots, b_5} \quad \|\vec{b}\|^2 + C \sum \max(0, 1 - y_i(b_0 + b_1h_1(\vec{x}_i) + b_2h_2(\vec{x}_i) + b_3h_3(\vec{x}_i) + b_4h_4(\vec{x}_i) + b_5h_5(\vec{x}_i))) \tag{2.35}$$

Another example is a radial basis function kernel which consists of Gaussian radial basis functions. RBFs are centered at each of the observations in the training set, so there is a total of n features in the basis. The basis functions are given by equation (2.36).

$$h_i(\vec{x}) = \exp\{-\gamma\|\vec{x} - \vec{x}_i\|^2\} \tag{2.36}$$

The function $h(\vec{x}_i)$ indicates whether the new data point $\vec{x}$ is in the neighborhood of $\vec{x}_i$. The magnitude of γ controls the size of these neighborhoods, large γ makes the peaks higher resulting in smaller neighborhoods. The resulting optimization formulation is shown in problem (2.37).

$$\min_{b_0, b_1, \dots, b_5} \|\vec{b}\|^2 + C \sum \max(0, 1 - y_i(b_0 + b_1 h_1(\vec{x}) + b_2 h_2(\vec{x}) + \dots + b_n h_n(\vec{x}))) \quad (2.37)$$

In practice we solve the dual of the formulation shown in problem (2.37) since only the inner products between the observations in the new space are required, rather than computing all of the new coordinates. This is known as the "kernel trick" and reduces the amount of computation required. A downside of the of the RBF kernel is that all of the training data points must be retained with the model since they are required to compute the inner products, but are a good choice if no prior knowledge of the structure of the data is known.

2.4.6 Conclusion

Chapter 2 has provided a review of the analytic including the methods used to process the image, the feature selection process, and the data used to train and test the model. We also presented the theoretical background required to understand the optimization method used to select the head pose estimates, and the SVM to classify the images. The next chapter presents problems that we uncovered through our research that leads us to doubt some fundamental assumptions and claims asserted in the development of the analytic. We show how these problems effect both the training and performance of the analytic, and cast doubt on its utility as a DeepFake detector.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:

Problematic Assumptions

The goal of this chapter is to show that three assumptions fundamental to the development of the analytic are false. The first assumption is that the estimated position of the outer landmarks are unaffected by the manipulations to the interior region of the face in a DeepFake face swap. The second assumption is that the optimization method the analytic uses is appropriate for estimating the head pose, and the final assumption is that the performance of the analytic generalizes across individuals as presented by the authors in their experiments. We use the insight gained from the discussion of these assumptions to interpret the results of our experiments that we present in the next chapter.

3.1 Landmark Position Estimation

The analytic takes advantage of the fact that in a DeepFake face swap only the inner region of the face is manipulated leaving the outer region unchanged. The analytic assumes that since the outer region of the face is unchanged, the estimated positions of the outer landmarks in this region are also unchanged. In this section we begin by presenting images as evidence that this assumption is not true, and then break down the operation of the method used to estimate facial landmarks as justification.

3.1.1 Example Demonstrating Dependence Between Facial Landmark Position Estimates

Figure 3.1 contains a pair of images Kazemi and Sullivan (2014) presented in their paper proposing the landmark position estimation algorithm that is implemented in the analytic. Each image contains a face with the estimated facial landmark positions shown as yellow dots. You can see that in both images several outer landmarks are blocked from the view of the camera by the hands in the image, but their positions are still estimated. This implies that the estimated positions of these landmarks, which are blocked from the view of the camera, are dependent on the estimated position of other landmarks. We take a closer look at the algorithm used in the analytic to estimate the position of the facial landmarks in the next section to demonstrate why this occurs.

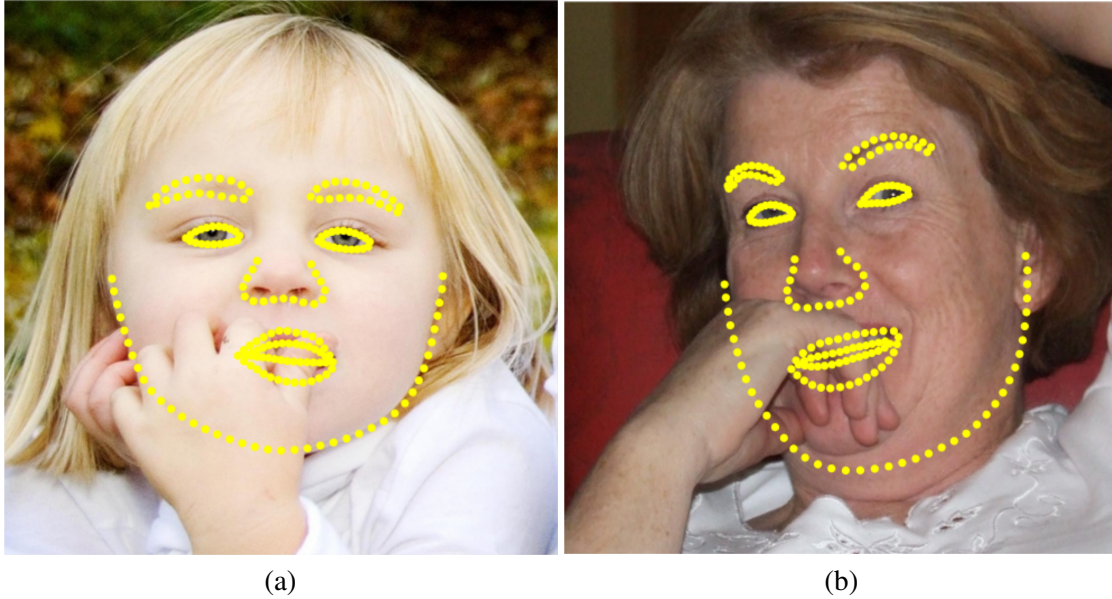


Figure 3.1. Photos illustrating dependence of facial landmark position estimates. The positions of the outer facial landmarks are estimated despite being blocked from the view of the camera by the hand in both images. Source: Kazemi and Sullivan (2014).

3.1.2 Summary of the Algorithm Used to Estimate Landmark Positions

This section presents the landmark position estimation algorithm developed by Kazemi and Sullivan (2014). We first introduce the overall structure of the model as a tree-based classifier along with other context necessary to understand the implementation of the algorithm. We then take a closer look at how split tests are designed to be shape invariant, and how splitting rules are determined in the training process. The overall goal of this section is to provide justification for the implications for the analytic presented in the next section.

This algorithm precisely estimates the position of the 68 facial landmarks in a computationally efficient manner by using a cascade of tree based regressors. We begin our discussion by introducing some terminology. Let $\mathbf{x}_i \in \mathbb{R}^2$ be the x and y coordinates of the i th facial landmark in an image I . We then define the shape vector $\mathbf{S} = [\mathbf{x}_1^T, \mathbf{x}_2^T, \dots, \mathbf{x}_{68}^T]$, which denotes the coordinates of all 68 facial landmarks in I . The algorithm is trained on data consisting of pairs of images and shape vectors $(I_1, \mathbf{S}_1), (I_2, \mathbf{S}_2), \dots, (I_n, \mathbf{S}_n)$. The mean shape $\bar{\mathbf{S}}$ is defined as the average of the shape vectors in the training set, and is used to initialize the algorithm.

The landmark detection technique refines, for a fixed number of iterations T , the mean shape $\bar{\mathbf{S}} \in \mathbb{R}^{68 \times 2}$ used as an initial estimate of the pixel coordinates for the 68 facial landmarks. At each iteration, k , the landmark positions are updated as

$$\hat{\mathbf{S}}_{k+1} = \hat{\mathbf{S}}_k + \gamma r_k(\hat{\mathbf{S}}_k, I) \quad (3.1)$$

where

- $\hat{\mathbf{S}}_{k+1}$ is the estimate of the landmark positions after iteration k
- $I \in \mathbb{R}^{m \times n}$ is a gray-scale representation of the image of interest, with dimensions $m \times n$.
- $\{r_k : \mathbb{R}^{68 \times 2} \times \mathbb{R}^{m \times n} \rightarrow \mathbb{R}^{68 \times 2}\}_{k=1}^T$ is a sequence of random forest regression functions.
- $\gamma \in (0, 1)$ is a *shrinkage parameter*, commonly used in boosted forests to mitigate overtraining.

The algorithm trains each tree based regressor r_k via gradient boosting with a squared error loss function. At each split node in the regression trees, a decision is made based on the difference between the intensities of two pixels. The position of the two pixels are defined by coordinates measured relative to the mean shape to improve the geometric invariance of the process. The algorithm determines the position of the pixels in the image by transforming the coordinates using the similarity transformation that is described in the next section. This transformation is the computationally expensive and is conducted once at each level in the cascade to maximize the speed of the algorithm. Next we take a closer look at how each tree based regressor is trained in the model, and how split tests are performed.

3.1.3 Tree based regressor

Each regression function r_k is composed of regression trees fit to residual targets as part of the gradient boosting algorithm. In the following subsections we look more closely at how the split tests are made shape invariant, as well as how the node splits are chosen. These details are used in the next section to justify why the estimated position of the outer facial landmarks are affected by manipulations to the inner region of the face.

Shape invariant split tests

As described by Kazemi and Sullivan (2014), the regression trees make a decision at each split node based on the difference between the intensities of two pixels. These pixels are defined by the coordinates $\mathbf{u}$ and $\mathbf{v}$ measured relative to the mean shape $\bar{\mathbf{S}}$. The position of these pixels in the image I are found by using the similarity transformation that transforms $\hat{\mathbf{S}}_k$ to $\bar{\mathbf{S}}$.

The steps to determine the transformation are as follows. Let $\bar{\mathbf{x}}_{k_u}$ be the facial landmark in the mean shape that is closest to $\mathbf{u}$. The offset from $\mathbf{u}$ is defined in equation (3.2).

$$\delta \mathbf{x}_u = \mathbf{u} - \bar{\mathbf{x}}_{k_u} \quad (3.2)$$

Then for a shape $\hat{\mathbf{S}}_k$ in image I , the position $\mathbf{u}'$ in image I that is qualitatively similar to the position of $\mathbf{u}$ relative to the mean shape is given by equation (3.3), and the equation for $\mathbf{v}'$ is similarly defined.

$$\mathbf{u}' = \mathbf{x}_{k_u} + \frac{1}{s} R^T \delta \mathbf{x}_u \quad (3.3)$$

The scale s and rotation matrix R are from the similarity transformation that transforms $\hat{\mathbf{S}}_k$ to $\bar{\mathbf{S}}$, which is determined by solving problem (3.4).

$$\min_{s, R, \mathbf{t}} \sum_{j=1}^p \|\bar{\mathbf{x}}_j - (sR\mathbf{x}_j + \mathbf{t})\|^2 \quad (3.4)$$

Solving problem (3.4) is the most computationally expensive part of the algorithm, so it is only done once at each level of the cascade. We can see in Figure 3.2 that the initial estimates of the shape have high error, this means that the position relative to the true shape for the pixels in the image used for comparison differ greatly from the relative positions that were intended when defining the split test. As the cascade progresses, the shape estimate improves and the relative positions of the pixels match more closely resulting in more effective tests.

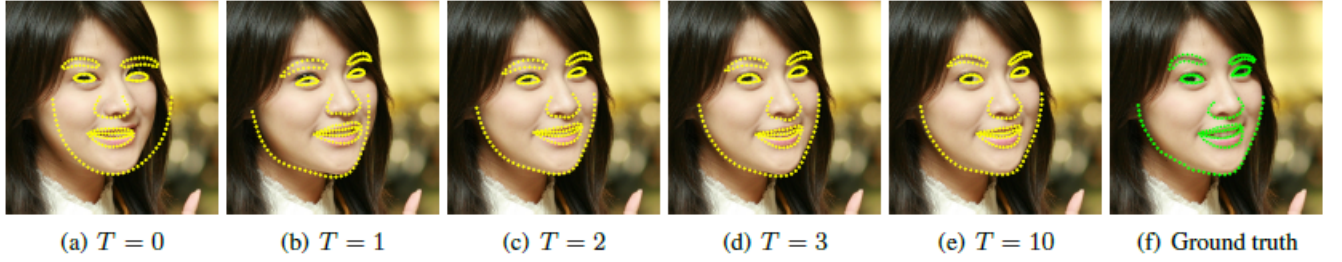


Figure 3.2. Figure depicting the progression of landmark position estimation at separate levels of the cascade. Shifting occurs to both inner and outer landmarks in initial levels of the cascade, while more focused corrections occur in later levels. Source: Kazemi and Sullivan (2014).

The split test is defined using three parameters, $\theta = (\tau, \mathbf{u}, \mathbf{v})$ where τ is the threshold for the difference in pixel intensity that determine the leaf that is assigned. The split test is given by equation (3.5).

$$h(I, \hat{\mathbf{S}}_k, \theta) = \begin{cases} 1 & I(\mathbf{u}') - I(\mathbf{v}') > \tau \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

Node splits

Each regression tree approximates residuals with a piecewise constant function by fitting a constant vector to each leaf node. Due to the large number of potential splits, the regression trees were trained by randomly generating a set of candidate splits, θ s, at each node, and then choosing the θ^* that minimizes the sum of squared error.

3.1.4 Implications for the Analytic

One of the foundational assumptions for the analytic is that the estimated position of the outer landmarks are unaffected by manipulations to the inner facial region of a DeepFake face swap. The assumption is false because the analytic determines the position of the facial landmarks by comparing the intensities of pairs of pixels that have been chosen at random in the training process. There is no condition that the pixel intensities used to estimate the position of the outer facial landmarks must be outside of the inner facial region. Also

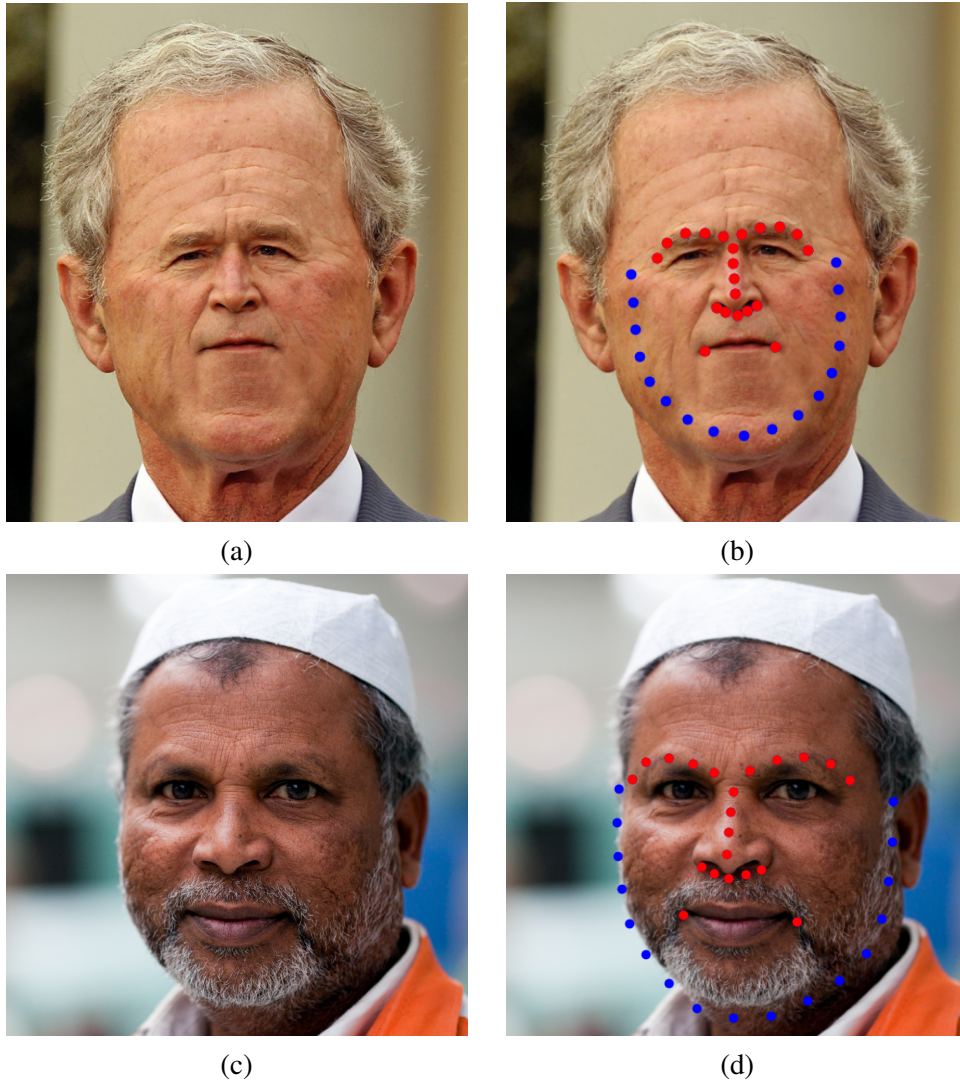


Figure 3.3. Photos illustrating that the estimated position of the outer landmarks are affected by manipulations of the inner facial region. Sources: @CelebsWith (Jan 20, 2014) and Flickr Faces Karras et al. (2019).

the regression trees used to update the estimated shape predict residual vectors that update the entire shape. This means that the analytic does not estimate the landmark positions individually, instead they are estimated together.

Figure 3.3 provides an example. In images (a) and (b), we have an image where the inner facial region has been manipulated. The images (c) and (d) on the same figure have a similar

pose and have not been manipulated. The estimated facial landmarks are shown in (b) and (d) with the central facial landmarks in red, while the outer facial landmarks are shown in blue. The outer facial landmarks in image (d) are at the outer edge of the face, while in image (b) the outer landmarks have been drawn in towards the center of the face. This shows that the estimated position of the outer landmarks are affected by the manipulations to the inner region of the face.

The assumption that the estimated position of the outer landmarks are not affected by the manipulation of the inner region of the face in a DeepFake face swap is a fundamental assumption for the analytic. Without the assumption, we cannot rely on the estimated position of the outer facial landmarks to retain the pose of the original image. This makes it unclear whether the comparison of the estimated head poses is a useful predictor to classify DeepFakes.

3.2 Head Pose Estimation Problem

Our goal in this section is to show that the optimization method the analytic uses is not appropriate for head pose estimation. We show how the optimization method can become trapped in a local minimum resulting in head poses that are selected facing away from the camera. For simplicity, we refer to this mistake as a flipped pose. As motivation for the following sections, we begin by demonstrating the consequences of the mistake by showing a flipped pose projected onto an image. We then provide a brief review of the formulation of the optimization problem as it was introduced in section 2.3.4, to provide context for the discussion of why the flipped poses occur. We finish by taking a look at the implications for the analytic when one of the two poses selected for an image is a flipped pose.

3.2.1 The Problem

Figure 3.4 includes two photos of Nicolas Cage. We added the estimated facial landmark positions to each photo in blue along with the points projected onto the image based on the estimated head pose in orange. The orange and blue points align closely in photo (a) while they are not as closely aligned in image (b), especially around the chin and outer edge of the face. This indicates that there is a problem with the head pose estimation process, because we have the same actor with similar poses and significantly different results. In the

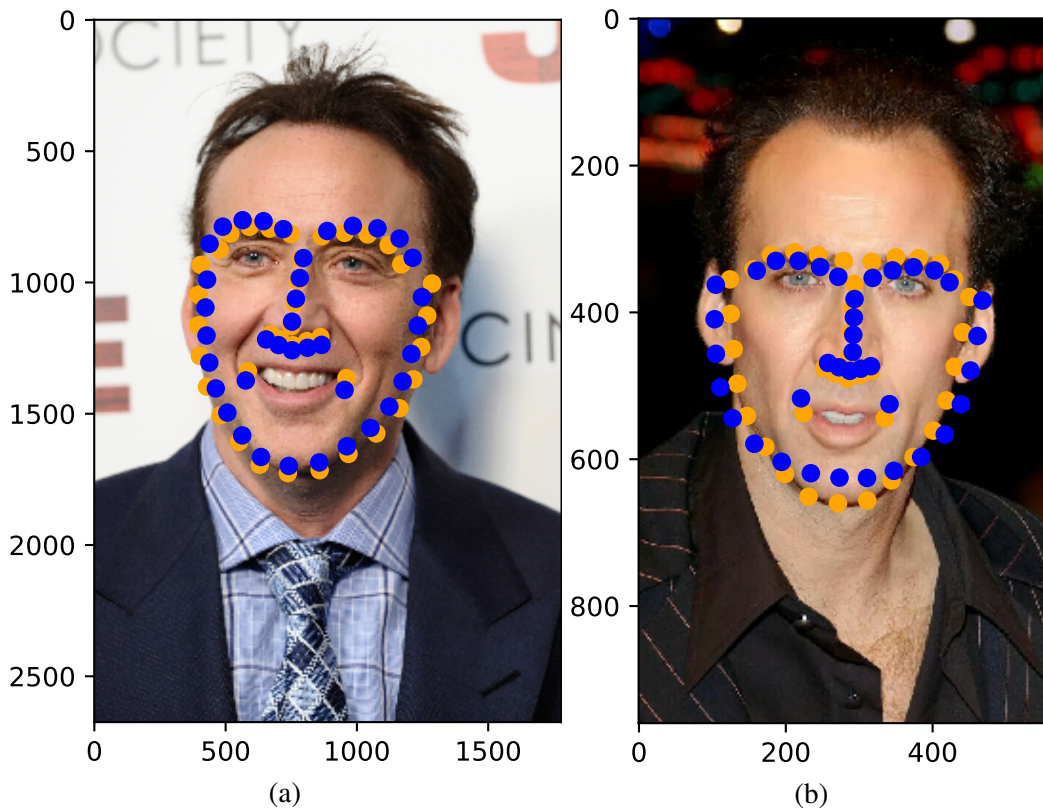


Figure 3.4. Photos of Nicolas Cage illustrating the effects of the optimization algorithm terminating in a local minimum. The estimated facial landmark positions are shown in orange, while the projected positions based on the head pose are shown in blue. You can see that the fit is better in (a) than in (b), especially around the chin. Source: Eliza Berman (2015)

next section we revisit the head pose estimation problem and show how these poses were selected.

3.2.2 Review of the Head Pose Estimation Problem

Recall from section 2.3.4 that the analytic estimates the head pose by choosing rotation matrix R and translation vector $\vec{t}$ to project the landmark positions in world coordinates, shown in Figure 2.5, to pixel coordinates in the image plane using equation (2.15). It then chooses the head pose to minimize the error between the projected points in the image plane and the estimated landmark positions by solving the optimization problem (3.6) using n

facial landmarks.

$$\min_{R, \vec{t}, s} \sum_{i=1}^n \left\| s \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} - \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left(R \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix} + \vec{t} \right) \right\|^2 \quad (3.6)$$

The analytic estimates two head poses for each image, one based on all of the facial landmarks, and the second based exclusively on the central facial landmarks. We denote the resulting rotation matrix and translation vector for the head pose based on all the facial landmarks by R_a and $\vec{t}_a$, while we use R_c and $\vec{t}_c$ to denote the rotation matrix and translation vector resulting from the head pose based on the central facial landmarks. The analytic utilizes the `solvePnP` method from `opencv` (Bradski 2000) to solve problem (3.6), which develops an initial solution by using the direct linear transform (DLT) method.

3.2.3 Head Pose Visualization

Here, we take a moment to describe a method that we commonly use in the following sections to visualize the head pose. We build this visualization by using equation (3.7) to transform the landmarks from world coordinates to camera coordinates and then plot the results. Figure 3.5 shows the effect, where (a) depicts the landmark positions in world coordinates, and (b) is the desired visualization with the landmark positions shown in camera coordinates and the camera aperture represented by the origin.

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = R \begin{bmatrix} U \\ V \\ W \end{bmatrix} + \vec{t} \quad (3.7)$$

3.2.4 Optimization trapped in a Local Minimum

The analytic initializes the Levenberg-Marquardt optimization algorithm by utilizing the DLT method to solve for the least squares solution of the system of homogeneous equations that includes equation (3.8) for each landmark position. The DLT method is a relaxed version of the pose estimation problem that returns a matrix R and translation vector $\vec{t}$ that

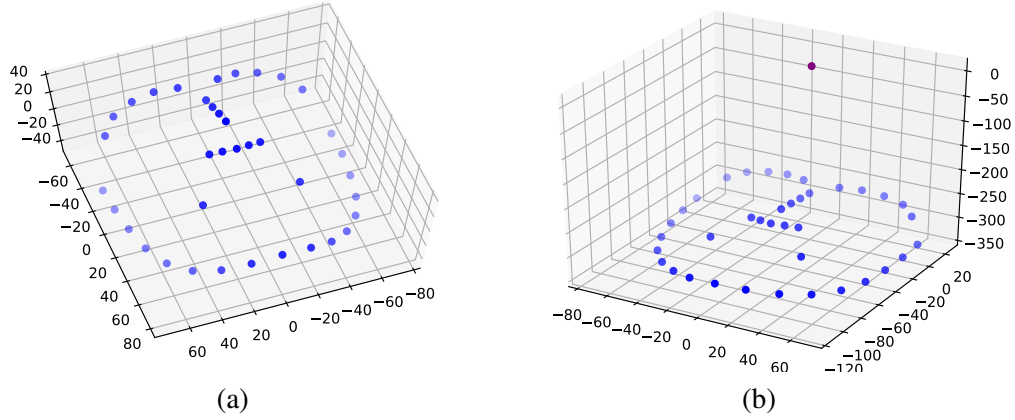


Figure 3.5. Figures illustrating transformation from world to camera coordinates. Figure (a) contains landmark positions in world coordinates based on the average face, while figure (b) shows the landmark positions in camera coordinates with the camera aperture represented by the origin in purple.

best satisfy the system of homogeneous equations, but does not require the matrix R to be a rotation matrix. The matrix R is then projected onto the space of rotation matrices to provide a rotation matrix for the initial solution. As a result, the analytic may initialize the Levenberg-Marquardt algorithm with a pose that is facing away from the camera.

$$s \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} - \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \left(R \begin{bmatrix} U_i \\ V_i \\ W_i \end{bmatrix} + \vec{t} \right) = 0 \quad (3.8)$$

Figure 3.6 shows visualizations of the head poses that were selected for the images in Figure 3.4 at the beginning of the section. We can see in (b) that the pose is facing away from the camera, while the pose in (a) is facing towards the camera. It is most important to notice among many differences there is a rotation of 180 degrees about the Z-axis along with the difference in translation along the Z-axis between the poses in Figures 3.6 (a) and (b). While these are drastically different approximations of the head pose, they result in similar projections into the image plane.

When the analytic uses equation (3.9) to project the landmarks in camera coordinates into

pixel coordinates on the image plane. The landmark positions shown in Figures 3.6 (a) and (b) result in similar projections in pixel coordinates. This is because the difference in translation along the Z-axis results in opposite signs for the Z-coordinates of each landmark, and the 180-degree difference in rotation between the poses results in opposite signs for each of the X and Y coordinates. These opposite signs are canceled in the fractions in equation (3.9), but the primary differences in the projections in the image plane are caused by the differences in magnitude of the Z-coordinates. In Figure 3.6 (a) the tip of the nose is closer to the camera and has a Z coordinate with smaller magnitude, while in (b) the outer edge of the face is closer to the camera aperture giving those landmarks Z coordinates with smaller magnitudes. The projection resulting from the flipped pose have wide cheeks when compared to the correct pose because the lower magnitude for the Z-coordinates produce larger magnitudes for p_x and p_y .

$$P' = \begin{bmatrix} p_x \\ p_y \end{bmatrix} = \begin{bmatrix} f_x \frac{X}{Z} + c_x \\ f_y \frac{Y}{Z} + c_y \end{bmatrix} \quad (3.9)$$

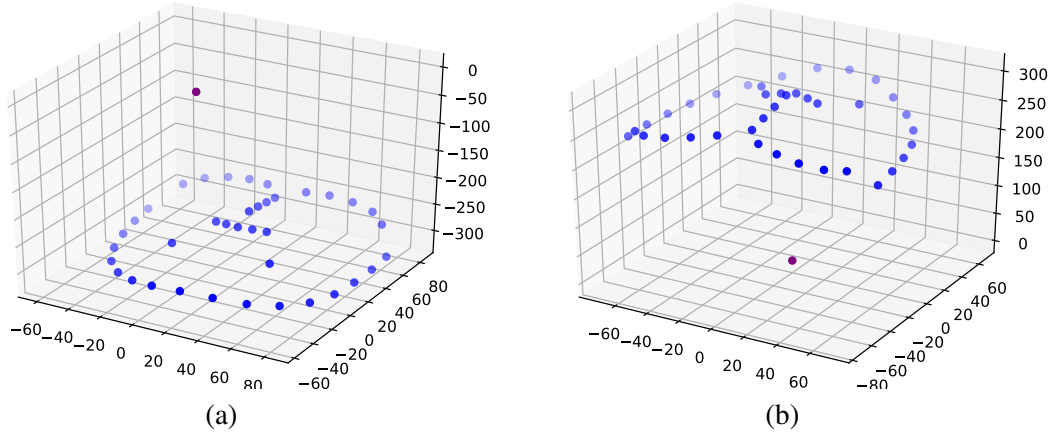


Figure 3.6. 3D plots illustrating the effects of the optimization algorithm terminating in a local minimum. Head poses selected based on all facial landmarks are shown in camera coordinates in blue. Notice the pose in (b) is facing away from the aperture of the camera represented by the origin.

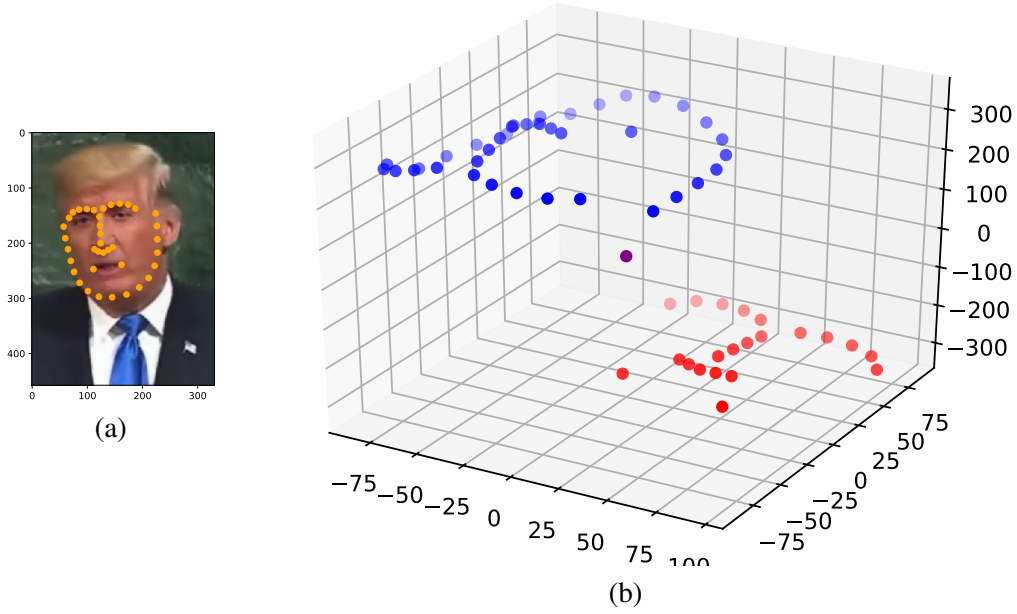


Figure 3.7. Figure illustrating the concept of a flipped head pose. The estimated position of the facial landmarks is shown in (a) while the head pose based on all facial landmarks is shown in blue in (b) and the head pose based on central facial landmarks are shown in red in (b).

3.3 Flipped Pose

We now present how this error in head pose selection has significant implications for the analytic when the error occurs on one of two head poses selected for a given image. Figure 3.7 (a) shows an image with the estimated landmark positions from `dlib` shown in orange. Figure 3.7 (b) are visualizations of the two poses that were chosen by the optimization algorithm with the blue pose chosen based on all of the facial landmarks and the red pose chosen based on the central facial landmarks. The optimization algorithm terminated in a local minimum when selecting the blue pose leaving it facing away from the camera. We can see that these poses differ significantly and the differences are reflected in the flattened difference in the rotation matrices $R_a - R_c$ and the flattened difference in translation vectors $t_a - t_c$ used to classify the image. These differences have consistently large magnitudes regardless of which of the two poses are flipped. In section 4.1.2 we explore the occurrence of the flipped poses in the UADFV data set, along with its implications for the training of the analytic and classification of the frames.

3.4 Generalizable Results

The goal of this section is to show by way of experimental results that the performance of the analytic does not generalize across new individuals. We begin by presenting the authors' results, and then reveal patterns in the data sets that improperly inflate the performance of the model. The remaining portions explain how these patterns influence the way the analytic classifies frames and present experimental results supporting these claims.

3.4.1 Author's Results

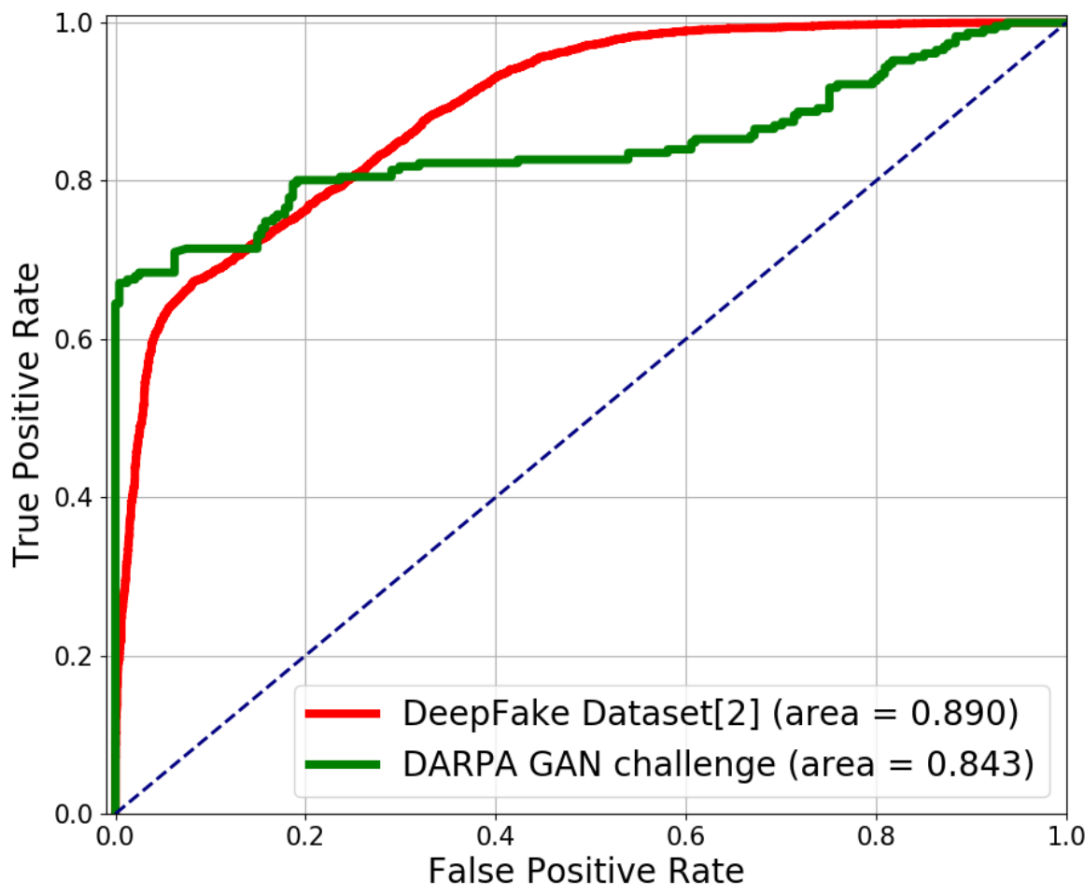


Figure 3.8. ROC curves showing the original classification results.
Source: Yang et al. (2019).

Figure 3.8 shows that the analytic performed well on both the UADFV data set as well as the subset from the DARPA GAN challenge data set. These results are undermined by the fact that the UADFV data set features Nicolas Cage's face exclusively in each of the fake

videos, and a large portion of the fake images in the DARPA GAN challenge data set are blurry. The remainder of this section demonstrates that these patterns make it easier for the analytic to classify DeepFakes in these data sets.

3.4.2 Blurry Images

The landmark estimation algorithm works by comparing the intensities of pairs of pixels in the image. Blurry portions of an image can interfere with these comparisons by blending the shadows preventing the landmark estimation algorithm from converging to the proper solution. Figure 3.9 illustrates the effect of progressively increasing the blurriness in the inner region of the face on the facial landmark estimates. The level of disruption to the facial landmark estimates depends on the pose of the face, along with shadows present in the background. If the head pose is far from the mean head pose used to initialize the algorithm, it is possible that pixels from outside of the face are used for comparison interfering with the estimation of the facial landmarks. If the background contains a shadow of the face or a similar edge, the algorithm can mistakenly outline this portion of the image as a feature of the face. These disruptions of the landmark estimation algorithm lead the analytic to create inaccurate head pose estimates that result in the analytic classifying the image as a DeepFake.

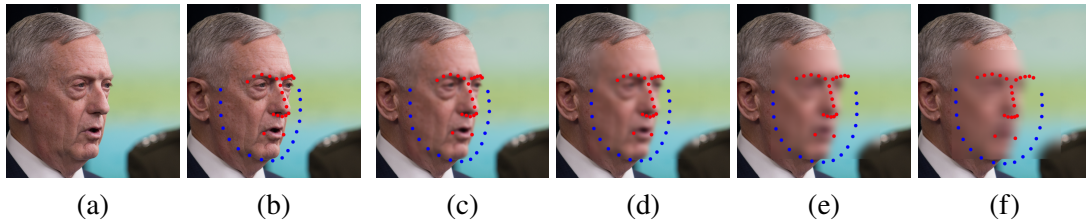


Figure 3.9. Photos illustrating the effect of blurring on estimated facial landmark position. Original photo in (a), original photo with landmarks in (b) increasing amounts of blurring to the facial region in (c) through (f) : Source- Karras et al. (2019)

3.4.3 Nicolas Cage Bias

We present an experiment to explore the effect of using Nicolas Cage exclusively in the fake videos in the UADFV data set. The experiment is conducted on a new data set consisting

of 100 photos scraped from a Google image search. Half of the photos contain Nicolas Cage, while the other half do not. While none of the images are DeepFakes, we use the model trained by the authors to attempt to classify the images containing Nicolas Cage. The experiment relies on the presence of bias in the training of the model to cause the analytic to classify images of Nicolas Cage as fake. Figure 3.10 summarizes the results of this experiment as a ROC curve.

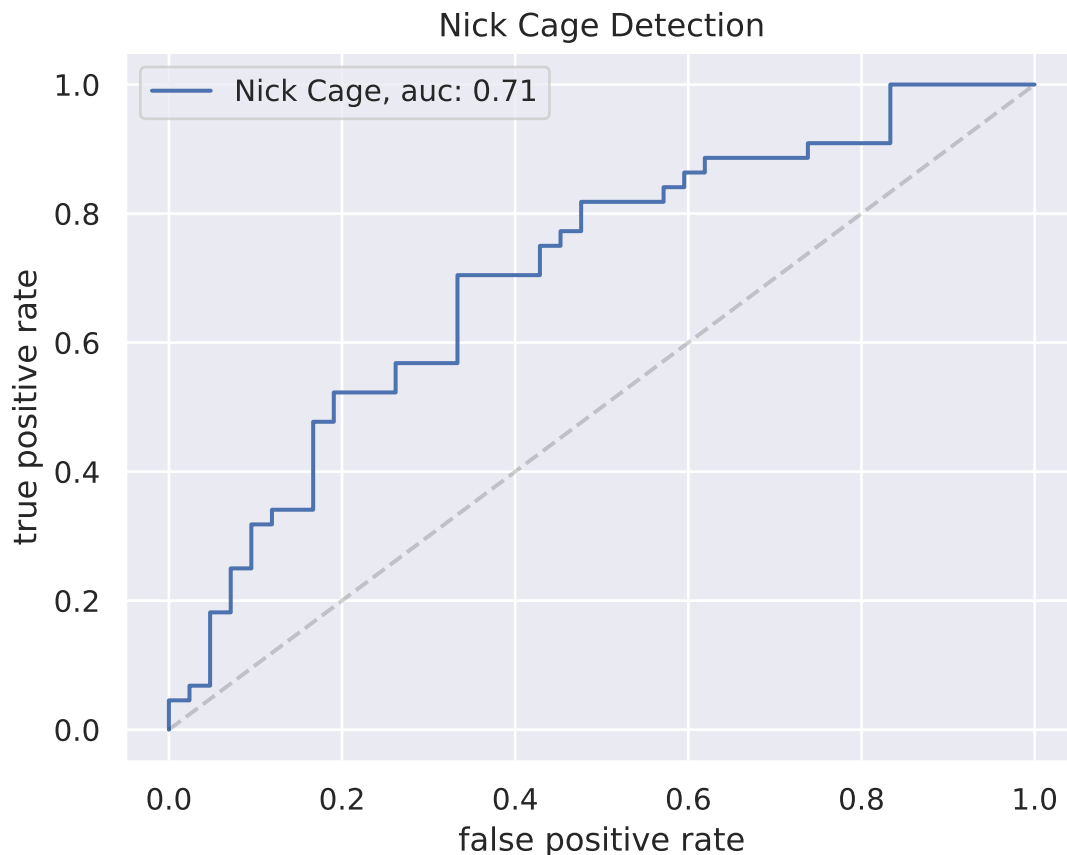


Figure 3.10. ROC curve showing the results of using the original model as a Nicolas Cage detector.

The resulting area under the curve (AUC) for this experiment is .71, which indicates that there is a strong bias in the training of the model. The presence of the bias is important because it shows that the analytic can generate bias towards individuals in the training set, as well as the fact that the bias inflated the performance of the analytic in the author's

experiment on the UADFV data set.

3.5 Conclusion

Throughout this chapter, we show that three assumptions fundamental to the development of the analytic are false. The first assumption is that the estimated position of the outer landmarks are unaffected by the manipulations to the inner region of the face in a DeepFake face swap. We present an example of an image with manipulations restricted to the inner region of the face that result in changes to the estimated position of the outer facial region. Section 3.1.2 provides a detailed description of the landmark estimation algorithm, which supports our conclusion that contradicts the authors' assumption. The second assumption is that the optimization method the analytic uses is appropriate for estimating the head pose. We show that the analytic may choose head poses that face away from the camera, which are clearly not appropriate approximations. The third and final assumption is that the performance of the analytic generalizes across individuals with the same performance the authors present from their experiments. We present patterns in the data sets used in the authors' tests and show how these patterns boosted the performance of the analytic in these tests.

CHAPTER 4:

Results

The purpose of this chapter is to show that exceptional performance of the analytic on the UADFV data set is due to the distribution of features caused by the flipped head poses, and demonstrate through experimental results that the performance of the analytic does not generalize to new data sets. We take a close look at the UADFV data set including the distribution of features both before and after the head poses are corrected. Next we describe the new data sets that we utilize in our experiments along with the methods we use to develop training and testing sets. We finish by presenting the results of our experiments showing that the performance of the analytic does not generalize to these new data sets with or without corrected head poses.

4.1 Classification Performance on the UADFV Data Set

This section demonstrates that the exceptional performance of the analytic on the UADFV data set is due to the distribution of the rotation matrix and translation vector features caused by the flipped head poses. The first subsection shows that we are able to replicate the performance presented by Yang et al. (2019) in their paper, and then demonstrate that the train test split leads to optimistic results. We then describe how we modify the analytic to correct flipped head poses, and then proceed to show how the value of both the rotation matrix and translation vector features are diminished when the head poses are corrected. The section concludes with a presentation of results that show how the classification performance drops when the head poses are corrected.

4.1.1 Replication of Author’s Results

In our first experiment we successfully train a model using the training set provided by the authors, and replicate the results Yang et al. (2019) present in their paper. Figure 4.1 shows our results. The success of this experiment shows that we are able to correctly train and implement the analytic.

We explore the performance of the model further by repeating this replication experiment

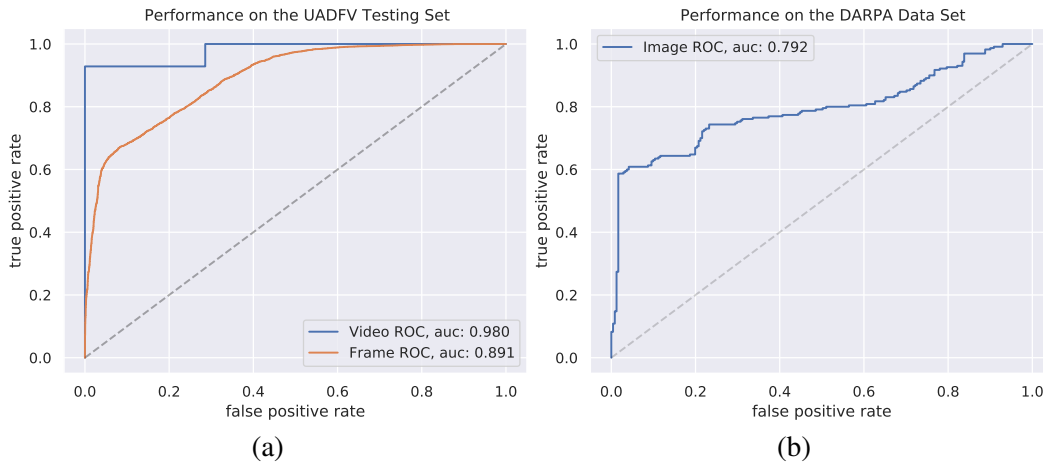


Figure 4.1. ROC curves showing the results of our experiment replicating the results presented by the author on both the UADFV (a) and the subset of the DARPA GAN challenge data set (b).

30 times utilizing a randomly selected training and testing split in each trial. Figure 4.2 summarizes the results of this new experiment. The red line represents the results presented by the authors, and show that the author's choice of the training and testing split create optimistic results. While the results of this experiment do not completely discredit the effectiveness of the analytic, it does lead us to second guess the generalizability of the results the authors present.

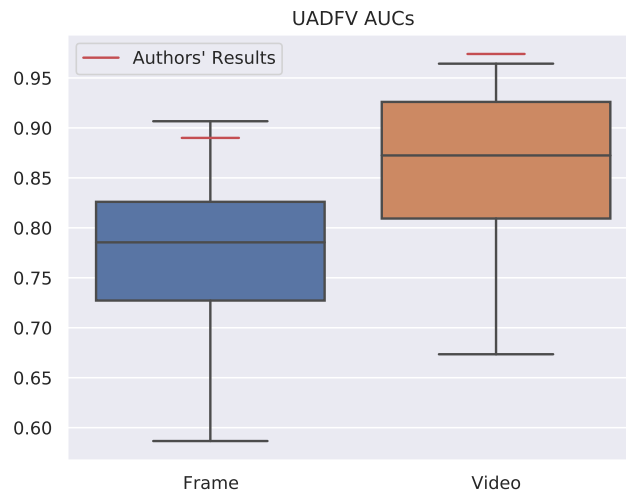


Figure 4.2. Boxplot summarizing 30 experiments with random train test splits from the UADFV data set.

4.1.2 Correcting Flipped Poses

In order to better understand the effect of the flipped poses on the classification results, we modify the analytic to correct for head poses facing away from the camera by first detecting the incorrect pose, and then re-initializing the optimization algorithm with a pose facing the camera. The modified analytic detects incorrect poses by checking the z component of the translation vector that was selected using the `solvePnP` method. If the z component is positive, the pose is facing away from the camera and requires correction, otherwise the pose does not require correction. The poses are corrected by re-initializing the `solvePnP` method with the same translation vector with opposite sign for the z component and an additional 180-degree rotation around the z -axis. This new pose used to initialize the algorithm is close enough to the correct head pose that the Levenberg-Marquardt algorithm successfully converges to a head pose facing towards the camera.

4.1.3 Effect of Flipped Poses on Classification

The modified analytic allows us to correct head poses that are facing away from the camera, as well as track the number of corrections that were required. Figure 4.3 shows the number of frames in the UADFV data set that require zero, one, or two pose corrections. The color of the bars indicate whether the frames are drawn from a real or DeepFake video. This plot indicates that a vast majority of the frames in the UADFV data set require one or more pose corrections, and a majority of the frames requiring a single pose correction are drawn from the DeepFake videos.

We have seen in section 3.2.4 that the frames requiring a single pose correction result in head pose estimates that differ significantly. Figure 4.4 provides an example of a frame from the UADFV data set that requires a single pose correction, and we can see the significant difference in the estimated head poses in Figure 4.4(b). These differences are reflected in both the $\vec{t}_a - \vec{t}_c$ and $R_a - R_c$ features used to classify the frame.

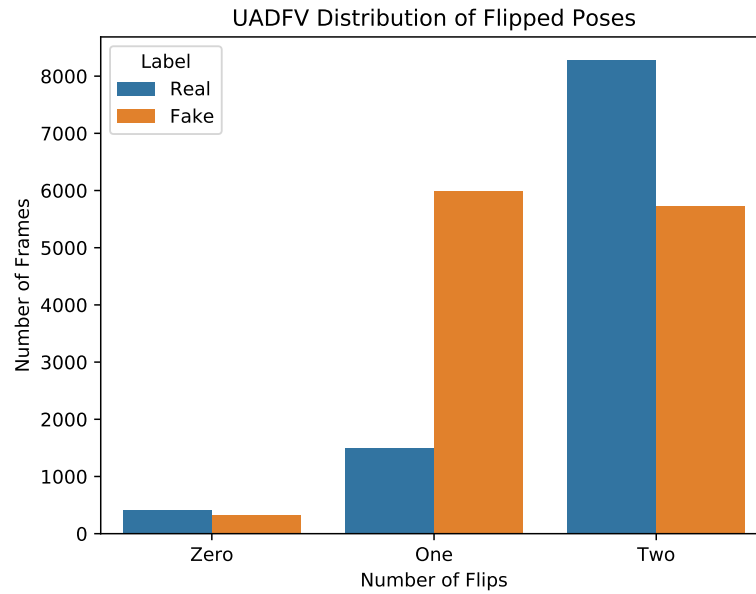


Figure 4.3. Barplot showing the number of frames requiring pose corrections in the UADFV data set.

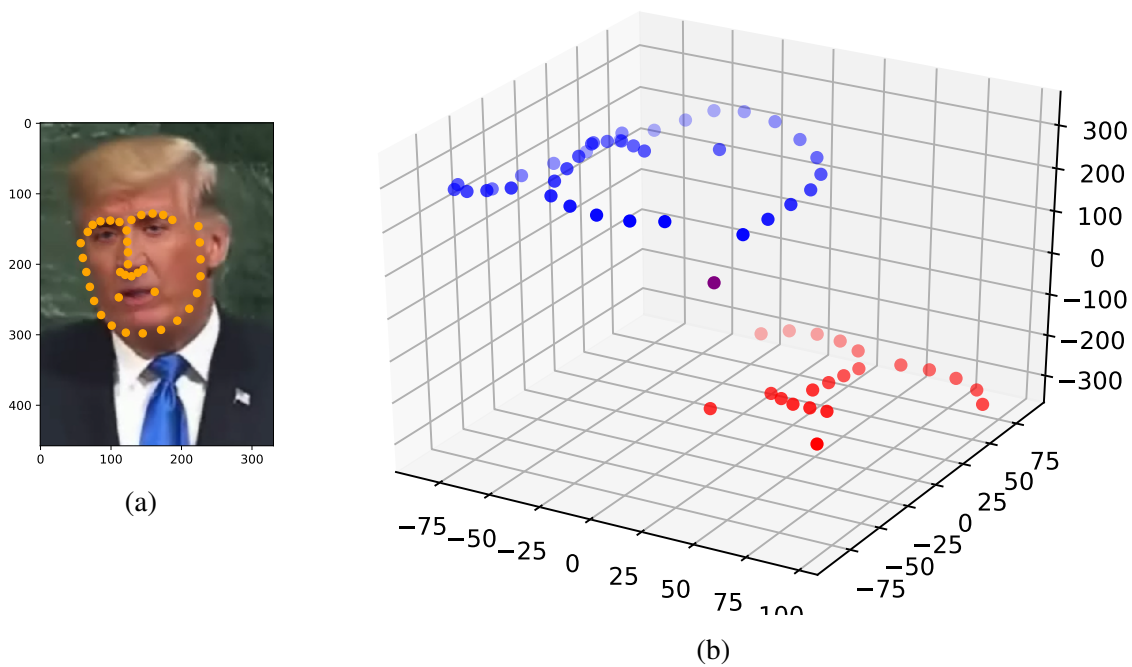


Figure 4.4. Figure illustrating the concept of a flipped head pose. The estimated position of the facial landmarks is shown in (a) while the head pose based on all facial landmarks is shown in blue in (b) and the head pose based on central facial landmarks are shown in red in (b).

Our next step is to use multiple plots to illustrate how the flipped poses affect the distribution of the features across the testing set. The scatter plot included in Figure 4.5(a) shows the frames from the UADFV data set plotted based on the x , y , and z components of the translation vector feature that are used by the analytic for classification. The points are colored based on the type of video the frames are drawn from, with purple corresponding to fake frames, and yellow corresponding to real frames. Figure 4.5(b) is a similar scatter plot, but the colors are based on the number of pose corrections that were required in each frame. The purple points require a single correction, green points require two corrections, and red frames require zero corrections. When considered together, these figures show that the two classes of images are well separated based on the translation vector feature alone, but this separation is primarily caused by the flipped poses that occur among the fake frames.

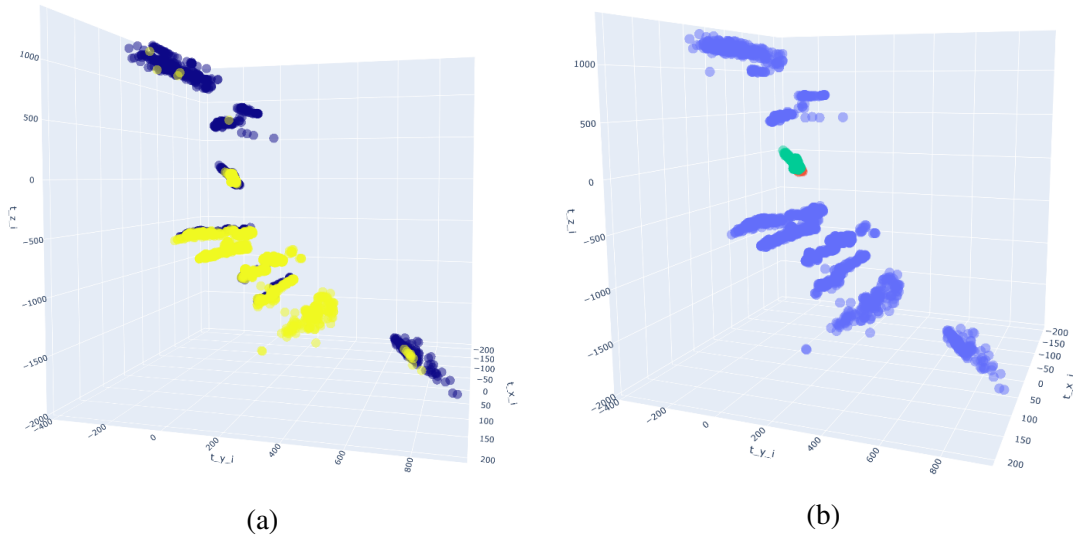


Figure 4.5. 3D Plots illustrating the distribution of the 3 dimensional translation vector feature in the UADFV testing set. Purple points in plot (a) correspond to fake frames, while yellow points correspond to real frames. Purple points in plot (b) correspond to frames requiring a single pose correction, while green points correspond to frames requiring two pose corrections, and red points representing frames requiring zero pose corrections.

Figure 4.6 (a) shows the distribution of the z -component of the translation vector feature when the feature is determined without correcting poses that face away from the camera. Similar to the barplot in Figure 4.3 the boxplots are categorized by the number of poses that require correction, and are colored by the type of video from which they originated. You can see that the fake frames that require a single pose correction have z components that are large and negative. Figure 4.6(b) is organized in the same manner, but shows the value of the z -component of the translation vector feature when it is calculated after correcting poses that face away from the camera. Considering the two plots together we can see that correcting the head pose reduces the magnitude of the z component in most of the frames and diminishes the distinction between the fake frames requiring a single pose correction and the rest of the data set.

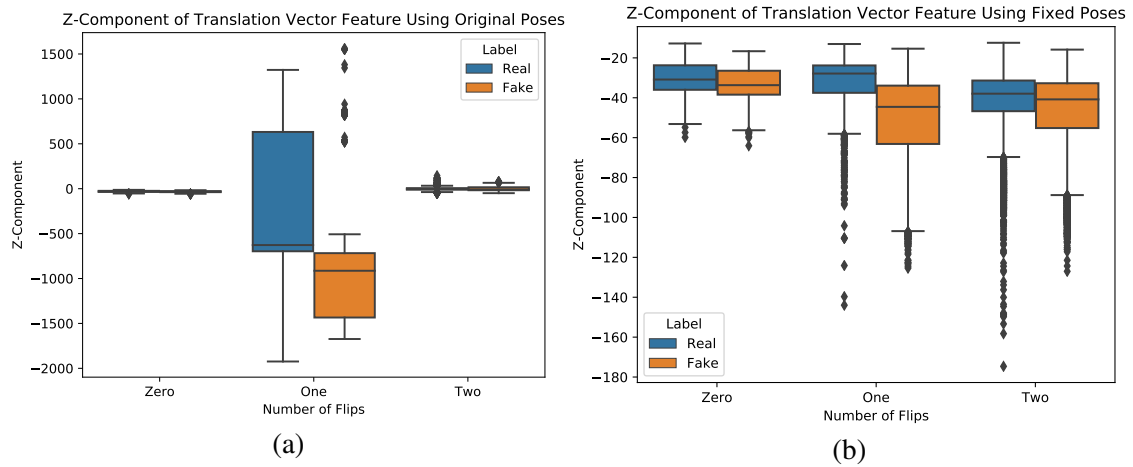


Figure 4.6. Boxplots illustrating the change in the distribution of the z -component of the translation vector feature after the poses are corrected in the UADFV data set.

This section has shown that the value of the translation vector feature is diminished once the head poses are corrected. Similarly for the rotation matrix when a single pose correction is required, an approximately 180-degree difference in rotation about the z -axis exists between the two poses. The correction of the pose facing away from the camera removes this difference in rotation, diminishing the value of the rotation matrix feature. In the next section we discuss the cosine distance feature that Yang et al. (2019) presents as evidence to support the validity of using inconsistent head poses to classify DeepFakes.

4.1.4 Cosine Distance

Yang et al. (2019) presents the cosine distance between head orientation vectors as evidence of inconsistencies between the estimated head poses that develop as a result of a DeepFake face swap. In this section we review the definition of head orientation vectors, and show how the cosine distance is calculated. We then present the distribution of the cosine distances in the UADFV data set both before and after the correction of the head poses, and ultimately show that the cosine distance cannot be used to reliably distinguish between real and DeepFake videos.

We begin with the definition of the head orientation vectors given by equations (4.1) and (4.2).

$$\vec{v}_a = R_a^T \vec{w} \quad (4.1)$$

$$\vec{v}_c = R_c^T \vec{w} \quad (4.2)$$

The vector $\vec{w} = [0, 0, 1]^T$ points from the center of the face outward through the nose along the z axis, and the head orientation is the vector that results from applying the rotation matrix to $\vec{w}$. The cosine distance between head orientation vectors give an indication of the difference in the directions that these two vectors are pointing, and is given by equation (4.7). Vectors pointing in the same direction have cosine distance close to 0, while vectors pointing in opposite directions have a cosine distance of 2.

$$d = 1 - \frac{\vec{v}_a \cdot \vec{v}_c}{\|\vec{v}_a\| \|\vec{v}_c\|} \quad (4.3)$$

The authors presented Figure 4.7 (a) in their paper, and it shows kernel density estimates (KDE) that approximate the distribution of cosine distances between head orientation vectors from a subset of the frames in the UADFV data set. The blue KDE represents the distribution for real frames, and the red KDE represents the distribution for DeepFake frames. Figure 4.7 (b) and (c) are boxplots that we created, and are organized in the same manner as those in Figure 4.6 in the previous section. These boxplots show that there is no meaningful

difference in the distribution of the cosine distance between head orientation vectors when considering the entire UADFV data set either before or after the head poses are corrected.

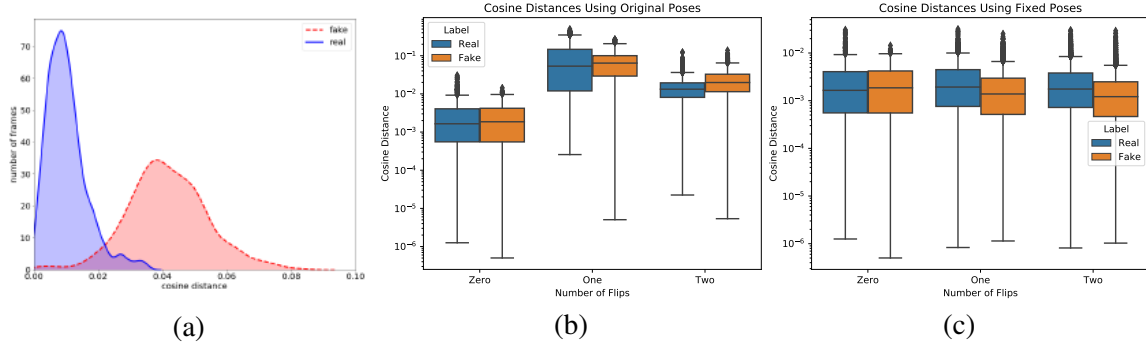


Figure 4.7. Plot (a) was presented by Yang et al. (2019) and shows the distribution of cosine distances for frames drawn from a subset of the UADFV data set. Plots (b) and (c) show that we do not see this cosine distance trend when considering the entire UADFV data set either before or after the poses are corrected.

The flipped poses do not have a considerable impact on the cosine distance metric. This is because a flipped pose is facing in the same direction as a correct pose, but is on the wrong side of the camera. This shows that the metric is not as effective as the authors intend.

In summary, we were unable to reproduce the plot in Figure 4.7 (a). It is worth noting that the authors use "a subset" of DeepFakes to generate this plot. Despite our inability to identify what subset of data the authors used for this plot, we did not detect a similar trend in any of the data sets considered throughout our work using either the uncorrected or corrected version of the analytic. Also, this is a poor metric because it is unable to detect the flipped head poses, which look the same direction but are on opposite sides of the camera.

4.1.5 Classification Results with Corrected Poses

Throughout this section we have shown how the correction of the flipped poses affect the distribution of features in the UADFV data set. Figure 4.8 presents the results of an experiment utilizing the same training and testing split developed by the author to show how the correction of flipped poses affect classification performance. The new model was trained

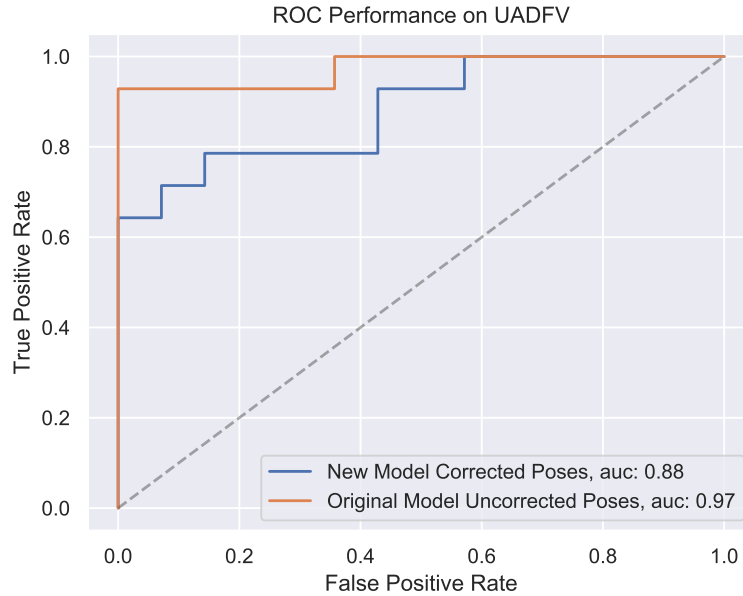


Figure 4.8. ROC curve showing the effect of correcting the head poses on classification performance on the UADFV data set.

and tested on the author’s train test split with corrected head poses, while the results of the original model match those presented by Yang et al. (2019) in the paper. We can see that the performance of the analytic drops with the implementation of the corrected pose, but an AUC of .88 indicates that the analytic performs as a worthwhile classifier on the UADFV data set. We attribute this performance to the overtraining of the analytic on Nicolas Cage as section 3.4.3 demonstrates. We present the results of a series of experiments on new data sets to further evaluate the performance of the analytic in the following section.

4.2 Descriptions of the Data Sets

Here, we provide a brief description of each data set that we used in our experiments. We include basic information such as the size of the data set, source of the videos and actors, number of manipulation techniques implemented, along with a subjective assessment of the quality of the DeepFakes. The goal is to provide enough context to understand the development of the train test splits that we summarize in the next section.

DeepFake Detection Challenge

Dolhansky et al. (2020) provides the DeepFake Detection Challenge (DFDC) data set, which is the largest of the data sets we worked with, consisting of over 100,000 clips featuring 3,426 paid actors. The DeepFakes were generated using 8 different methods along with varying levels of post processing. Not all of the DeepFakes are auto-encoder based face swaps, but many are. The data set was developed with the goal of being large enough and diverse enough so that a deepfake detection model can be trained only on the DFDC data set and can generalize to DeepFakes one might see "in the wild".

Face Forensics++

The Face Forensics++ data set created by Rossler et al. (2019) consists of 1,000 videos and feature four deep fake generation techniques. The videos in the data set were sourced from YouTube, and the videos were compressed at various levels using methods common to social network sites to create high, medium, and low quality videos. Each of the videos from YouTube occur once as a real video and once as a fake, and they are swapped in pairs so the subjects are not in multiple fake videos.

DeepFake Detection Data Set

The DeepFake Detection (DFDD) data set was developed by Dufour and Gully (2019) and contains hundreds of DeepFake videos that were produced using paid actors. The videos are organized around scenes, where an actor is filmed in one scene and several deepfakes were made by substituting actors with a similar appearance. The creators used a total of 5 different methods to generate deepfakes with varying levels of quality.

Celeb-DF Data Set

The Celeb-DF data set created by Li et al. (2020) consists of 5,639 high quality DeepFake videos along with 590 real videos. The real videos are interviews of 59 various celebrities sourced from YouTube with diversity in characteristics such as face size, orientation, and lighting. A single method is used to generate consistently high quality deepfake videos.

	Training Frames	Testing Frames
FF++	6392	1608
DFDD	2632	372
DFDC	600	600
Celeb-DF	3766	944

Table 4.1. Table summarizing the number of frames in each of the train test splits for our experiments.

4.3 Train Test Splits

In this section we describe the train test splits we developed for our experiments on each data set. These splits vary based on the size of the data sets, and the way the DeepFakes were generated. We create these train test splits to prevent overtraining on particular individuals or scenes, since previous experiments demonstrate the analytic’s capacity to do so. We are unable to identify which manipulation technique was featured in each video, so we were unable to balance the train test split based on the technique used to generate the image. Table 4.1 summarizes the number of frames that are included in the training and testing sets.

DeepFake Detection Challenge

Due to the organization of the DFDC data set, we are able to determine the actors featured in each video. This allows us to ensure that each actor is either included in the training set or testing set, and are either featured in a real video or fake video but not both.

Face Forensics++

The DeepFakes in the Face Forensics++ data set were created by swapping the identities in pairs of videos. We create a train/test split so that a single video from each pair is included in either the training or testing set, and as either a real or fake video. For example, a real video in the training set is not included as a deepfake video with a new identity in either the training or testing sets.

DeepFake Detection Data Set

The DeepFake Detection (DFDD) data set was organized around scenes, with an actor in the real video swapped with several other actors to create several deepfakes. We created the train/test split so that each scene is included in either the training or testing set, and as either a real or fake video.

Celeb-DF Data Set

Each DeepFake in the Celeb-DF data set is a video that was originally a video interview. The train test split was organized so that the actors were not repeated in either the training or testing sets.

4.3.1 Results

We conduct a series of three experiments on each of the new data sets to test the analytic using corrected and uncorrected head poses as well as varying the source of the training data. The results are summarized in Table 4.2. The three columns correspond to the three different experiments. The "Provided" column summarizes the results of using the author's pretrained model. The "Corrected" and "Uncorrected" columns summarize the results of training a model on the training set formed from the new data set and either using the corrected or uncorrected poses throughout the training and classification process. Overall, the analytic performs poorly on these new data sets. The AUCs are nowhere close to the promising results that Yang et al. (2019) present in their paper, and do not improve even if the model is retrained from the new data set or if the poses are corrected.

	Provided	Uncorrected	Corrected
FF++	.52	.60	.63
DFDD	.46	.47	.28
DFDC	.54	.45	.45
Celeb-DF	.50	.58	.59

Table 4.2. AUC scores for various models. The “Provided” model is pretrained on UADFV and provided by the authors. The “Uncorrected” model is trained on the training frames from Table 4.1, without the head pose correction. The “Corrected” model is also trained on the frames in Table 4.1, but uses the corrected head poses.

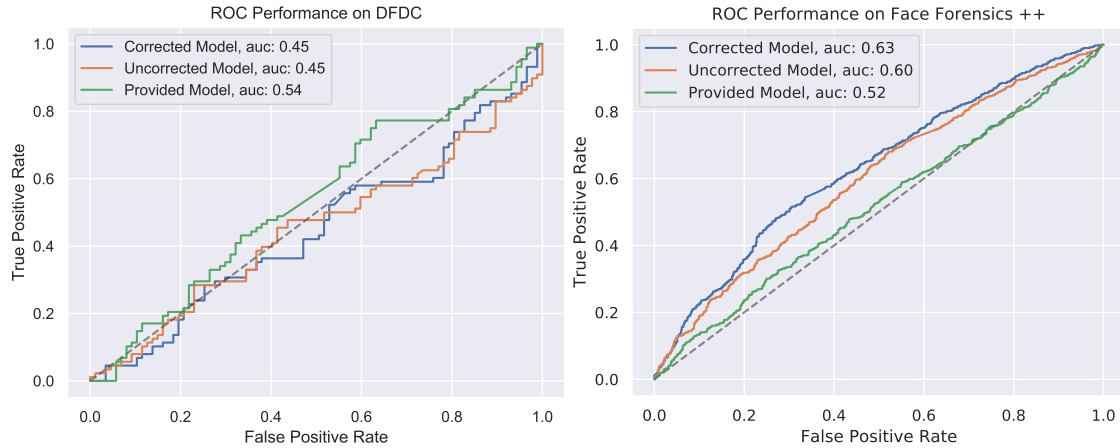


Figure 4.9. ROC curves presenting the results of our experiments on the DFDC and Face Forensics ++ data sets.

4.4 Conclusion

Throughout this chapter we show that the exceptional performance of the analytic on the UADFV data set is due to the distribution of features caused by the flipped head poses, along with the overtraining of the analytic on the facial features of Nicolas Cage. We present experimental results that show that the performance of the analytic does not generalize to new data sets. We justify our claims by presenting the distribution of features in the UADFV data

set both before and after the head poses were corrected, and finish by presenting the results of our experiments on train test splits designed to minimize the potential for overtraining and provided the best possible indication of the generalizability of the results. Overall, the analytic performed poorly in these tests indicating that several challenges remain to successfully classify deepfakes using inconsistent head poses.

List of References

- Adby P (2013) *Introduction to Optimization Methods* (Springer Science & Business Media).
- Afchar D, Nozick V, Yamagishi J, Echizen I (2018) Mesonet: a Compact Facial Video Forgery Detection Network. *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–7 (IEEE).
- Agarwal S, El-Gaaly T, Farid H, Lim SN (2020) Detecting Deep-Fake Videos from Appearance and Behavior. arXiv preprint arXiv:2004.14491.
- Agarwal S, Farid H, Gu Y, He M, Nagano K, Li H (2019) Protecting World Leaders Against Deep Fakes. *CVPR Workshops*, 38–45.
- Anonymous (2020a) DFaker GitHub. <https://github.com/dfaker/df>.
- Anonymous (2020b) FakeApp. <https://www.malavida.com/en/soft/fakeapp/>.
- Baltrusaitis T, Zadeh A, Lim YC, Morency LP (2018) Openface 2.0: Facial Behavior Analysis Toolkit. *2018 13th IEEE International Conference on Automatic Face & Gesture Recognition (FG 2018)*, 59–66 (IEEE).
- Bradski G (2000) The OpenCV Library. Dr. Dobb’s Journal of Software Tools.
- @CelebsWith (Jan 20, 2014) George Bush with a Tiny Face. <https://twitter.com/CelebsWith/status/425398141136154624>, accessed: Feb 26, 2021.
- Ciftci UA, Demir I, Yin L (2020) Fakecatcher: Detection of Synthetic Portrait Videos Using Biological Signals. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.
- Dolhansky B, Bitton J, Pflaum B, Lu J, Howes R, Wang M, Ferrer CC (2020) The Deepfake Detection Challenge Dataset. arXiv preprint arXiv:2006.07397.
- Dufour N, Gully A (2019) Contributing Data to Deepfake Detection Research. Google AI Blog.
- Eliza Berman (2015) Nicolas Cage: ‘Personal Issues Don’t Have a Place in Film Commentary’. Accessed March 5, 2021, <https://time.com/3973963/nicolas-cage-interview/>.
- Gareth Dunstone VA (2021) faceswap GitHub. <https://github.com/deepfakes/faceswap>.

- Guarnera L, Giudice O, Battiato S (2020) Deepfake Detection by Analyzing Convolutional Traces. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 666–667.
- Güera D, Delp EJ (2018) Deepfake Video Detection Using Recurrent Neural Networks. *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–6 (IEEE).
- Hata K, Savarese S (2015) Course Notes. CS231A, Stanford University.
- Huh M, Liu A, Owens A, Efros AA (2018) Fighting Fake News: Image Splice Detection via Learned Self-consistency. *Proceedings of the European Conference on Computer Vision (ECCV)*, 101–117.
- Kannala J, Brandt SS (2006) A Generic Camera Model and Calibration Method for Conventional, Wide-angle, and Fish-eye Lenses. *IEEE transactions on pattern analysis and machine intelligence* 28(8):1335–1340.
- Karras T, Laine S, Aila T (2019) A Style-based Generator Architecture for Generative Adversarial Networks. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4401–4410.
- Kazemi V, Sullivan J (2014) One Millisecond Face Alignment with an Ensemble of Regression Trees. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 1867–1874.
- King DE (2009) Dlib-ml: A Machine Learning Toolkit. *The Journal of Machine Learning Research* 10:1755–1758.
- Li Y, Chang MC, Lyu S (2018a) In Ictu Oculi: Exposing AI Created Fake Videos by Detecting Eye Blinking. *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–7 (IEEE).
- Li Y, Chang MC, Lyu S (2018b) In Ictu Oculi: Exposing AI Generated Fake Face Videos by Detecting Eye Blinking. arXiv preprint arXiv:1806.02877.
- Li Y, Yang X, Sun P, Qi H, Lyu S (2020) Celeb-df: A Large-scale Challenging Dataset for Deepfake Forensics. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 3207–3216.
- Marquardt DW (1963) An Algorithm for Least-Squares Estimation of Nonlinear Parameters. *Journal of the society for Industrial and Applied Mathematics* 11(2):431–441.
- Matern F, Riess C, Stamminger M (2019) Exploiting Visual Artifacts to Expose Deepfakes and Face Manipulations. *2019 IEEE Winter Applications of Computer Vision Workshops (WACVW)*, 83–92 (IEEE).

- Nguyen HH, Fang F, Yamagishi J, Echizen I (2019a) Multi-task Learning for Detecting and Segmenting Manipulated Facial Images and Videos. *2019 IEEE 10th International Conference on Biometrics Theory, Applications and Systems (BTAS)*, 1–8 (IEEE).
- Nguyen HH, Yamagishi J, Echizen I (2019b) Use of a Capsule Network to Detect Fake Images and Videos. arXiv preprint arXiv:1910.12467.
- Noon van der Silk VA, Leung (2019) faceswap-GAN GitHub. <https://github.com/shaoanlu/faceswap-GAN>.
- Petrov I, Gao D, Chervoniyy N, Liu K, Marangonda S, Umé C, Jiang J, RP L, Zhang S, Wu P, et al. (2020) Deepfacelab: A Simple, Flexible and Extensible Face Swapping Framework. arXiv preprint arXiv:2005.05535.
- Piña E (2011) Rotations with Rodrigues' Vector. *European journal of physics* 32(5):1171.
- Ranganathan A (2004) The Levenberg-Marquardt Algorithm. Accessed March 11, 2021, http://www.ananth.in/Notes_files/lmtut.pdf.
- Rossler A, Cozzolino D, Verdoliva L, Riess C, Thies J, Nießner M (2019) Faceforensics++: Learning to Detect Manipulated Facial Images. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1–11.
- Satya Mallick (2016) Approximate Focal Length for Webcams and Cell Phone Cameras. Accessed February 28, 2021, <https://learnopencv.com/approximate-focal-length-for-webcams-and-cell-phone-cameras/>.
- Shane S, Mazzetti M (2018) Inside a 3-year Russian Campaign to Influence U.S. Voters. *The New York Times* (February 17), <https://www.nytimes.com/2018/02/16/us/politics/russia-mueller-election.html>.
- Taub A, Fisher M (2018) Where Countries are Tinderboxes and Facebook is a Match. *New York Times* (April 22), <https://www.nytimes.com/2018/04/21/world/asia/facebook-sri-lanka-riots.html?auth=login-google>.
- Trump DJ (2017) National Security Strategy of the United States of America. Technical report, Executive Office of The President Washington DC Washington United States.
- Wang SY, Wang O, Zhang R, Owens A, Efros AA (2020) CNN-Generated Images are Surprisingly Easy to Spot... For Now. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8695–8704.
- Whitaker L (2020) Course Notes. OA4118, Data Mining, Naval Postgraduate School.

- Wu Y, AbdAlmageed W, Natarajan P (2019) Mantra-net: Manipulation Tracing Network for Detection and Localization of Image Forgeries with Anomalous Features. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9543–9552.
- Yang X, Li Y, Lyu S (2019) Exposing Deep Fakes Using Inconsistent Head Poses. *ICASSP 2019-2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 8261–8265 (IEEE).
- Yu N, Davis LS, Fritz M (2019) Attributing Fake Images to GANs: Learning and Analyzing GAN Fingerprints. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 7556–7566.
- Zhang L, Wen T, Shi J (2020) Deep Image Blending. *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 231–240.
- Zhang X, Karaman S, Chang SF (2019) Detecting and Simulating Artifacts in GAN Fake Images. *2019 IEEE International Workshop on Information Forensics and Security (WIFS)*, 1–6 (IEEE).
- Zhao Z, Wang P, Lu W (2020) Detecting Deepfake Video by Learning Two-Level Features with Two-Stream Convolutional Neural Network. *Proceedings of the 2020 6th International Conference on Computing and Artificial Intelligence*, 291–297.

Initial Distribution List

1. Defense Technical Information Center
Ft. Belvoir, Virginia
2. Dudley Knox Library
Naval Postgraduate School
Monterey, California