Final Project Report: Identifying the Faces of Humanity by Improving a Simple Convolutional Neural Network Model

Ethan Nguyen-Tu
Georgia Institute of Technology
Atlanta, GA 30332
ethannguyentu@gmail.com

Abstract

The combination of improved image generation models with their growing adoption has resulted in the creation of numerous online imagery that mislead human observers who cannot tell that they are generated. The efficiency and time saving capabilities of these models makes it unlikely that this trend will stop, so it is necessary to develop measures that can differentiate between authentic, human-originated images from artificially generated images. I proceed to test possible improvements on an existing model, developed by Fatima Salman and Samy Abu-Naser, that is designed to differentiate between real and fake human faces with the goal of enhancing the possible capabilities of modern models that can solve this issue. In doing so, I find that increasing the learning rate or adding a convolutiton-relu-maxpool layer group does not improve the model, while applying image transformation pre-processing improves the model's ability to generalize. The code for this paper can be found https://github.com/EthanNguyen-Tu/CS4644_FinalProject [9]

1. Introduction

The advancement of image generation from the development of Generative Adversarial Networks (GANs) has resulted in images that are increasingly difficult for the human eye to identify as fake [4]. While useful for illustrating ideas for those who are not artistically inclined, these artificial images have also been used for illegal activities, such as identity fraud and financial fraud, that harm the everyday person [10]. This malicious use of artificial intelligence makes detecting deepfakes a critical research area to help protect those who encounter fake imagery.

To support the effort of detecting fake imagery in the area of human identity, I proceed to test improvements to an existing model, developed by Fatima Salman and Samy Abu-Naser, that is capable of identifying the differences between real and fake images of human faces [12]. Specifically, I test three possible improvements of increased learning rate, additional layers, and image transformation pre-processing with the goal of discovering which modifications enhance the model's ability to differentiate fake images of human faces from real. In doing so, this paper provides clarity on avenues for additional improvements.

With the improvement of a model that can accurately differentiate between real and artificially generated images of human faces, people can gain confidence on the authenticity of the imagery they interact with online. For example, by applying the model to social media, users would be able to gain insight on whether a person's physique is realistically possible. Similarly, by applying the model to dating applications, users would be able to better identify misleading accounts.

2. Related Works

Recent years have seen significant improvement to human face classification due to the introduction and development of deep learning models and techniques. Notable models that have been developed include RetinaFace, FaceNet, GhostFaceNet, and DeepFace.

2.1. RetinaFace

RetinaFace is a model that performs multilevel pixel-wise face localisation using a single-shot. It takes advantage of joint extra-supervised and self-supervised multi-task learning. This approach saw improvement in face detection and managed to run real-time on a single CPU core by employing light-weight backbone networks [5].

2.2. FaceNet

FaceNet maps images to a Euclidean space using a face embedding model. It leverages a triplet mining method to achieve greater representational efficiency. This approach saw state-of-the-art face recognition performance using 128-bytes per face [3].

2.3. GhostFaceNet

GhostFaceNet is a lightweight face recognition model that was found to offer better performance than that of state-of-the-art big convolutional neural network (CNN) models while requiring lower computational complexity [8].

2.4. DeepFace

DeepFace is one of the first deep learning models that managed to achieve near-human accuracy when verifying whether a face is real or fake. It leveraged the largest facial dataset of its time, containing four million facial images belonging to over 4,000 identities, to train a nine-layer deep neural network [13].

2.5. Three Image Classes

It was found through the exploration of trends and patterns observed in real, deepfake, and synthetic images of faces that they are separate classes where categorizing images into these three categories instead of grouping deepfake and synthetic images together as a generalized fake category can help develop better models [11].

3. Method / Approach

The possible optimizations tested in this paper are increasing the learning rate of, adding additional layers to, and image transformation pre-processing for the Keras model implemented by Fatima Salman and Samy Abu-Naser in both Keras and PyTorch variants. By testing the modifications in both Keras and PyTorch variants, I am able to test both the potential optimizations and any differences between the Keras implementation and the PyTorch implementation.

3.1. Reference Model

The Keras Reference Model is a sequential model that uses the Adam optimizer and a learning rate of 0.0001. It has convolution, max pooling, flattening, and dense layers with ReLU and softmax as its activation methods. Specifically the sequential layers in order are:

1. Conv2D
2. MaxPooling2D
3. Conv2D
4. MaxPooling2D
5. Conv2D
6. MaxPooling2D
7. Conv2D
8. MaxPooling2D
9. Conv2D(256, (3, 3)
10. MaxPooling2D
11. Conv2D
12. MaxPooling2D
13. Flatten
14. Dense
15. Dense

Moving forward, I refer to the Fatima Salman and Samy Abu-Naser Keras model as the "Reference Model" [12].

3.2. Choice of Model Configuration

Since the goal of this paper's models is to correctly perform a classification task of determining whether an image of a human face is real or fake by outputing the probability of either classification, categorical cross entropy is chosen as the loss function for the Keras models while cross entropy loss is chosen as the loss function for the PyTorch models. As a result, the Keras models receive one-hot encoded target labels while the PyTorch models receive integer class labels. These loss functions were chosen as opposed to the binary variants of the loss functions to allow for the possibility of testing the separation of the fake classification into deepfake and synthetic images for accuracy improvements, which was not conducted due to time constraints [11].

Since the Keras Reference utilizes the Adam optimizer and a learning rate of 0.0001, the Adam optimizer is given to all models to serve as their optimizer, while the learning rate of 0.0001 is maintained for all models except for those that test increases to the learning rate. For the PyTorch conversion, convolution kernel size of 3 and stride of 1 and maxpool kernel size of 2 and stride of 2 are chosen to match the Keras Reference, while a padding of 1 is added to the PyTorch conversion model. Since the default padding of Keras convolution 2D is "valid", which means no padding, setting padding to 1 was a mistake in translation of the Keras Reference to PyTorch, so it is a source of error in this paper and correcting it to 0 is left to future research.

3.3. Establishing the Baseline

I first implement the Reference Model to serve as the baseline for the Keras model variants detailed in this paper. If the transfer of the Reference Model, referred to moving forward as the "Keras Reference," achieves equal to or better accuracy and loss as the accuracy and loss of the Reference Model, then I conclude that the Reference Model has been successfully transferred over [12]. Then, for the PyTorch variants, I replicate the Reference Model using PyTorch to have this PyTorch conversion model, henceforth referred to as the "PyTorch Conversion," serve as the baseline. If the PyTorch Conversion likewise achieves equal to or better accuracy and loss as the Reference Model and has similar final training loss, training accuracy, validation loss, and validation accuracy to the Keras Reference, where the

layers and initial parameters are the same and given 2% tolerance to account for framework differences, then I conclude that the PyTorch Conversion successfully replicates the Reference Model.

3.4. Evaluation Procedure

Improvements to the final validation accuracy and testing accuracy of the models are the chosen core measures of performance and modification success. Final validation accuracy was chosen to see how the baseline model's performance changes due to each modification in its training process. Testing accuracy is chosen to see how each model performs on unseen data. Furthermore, two test sets are used in this paper, one split from the main dataset along with the train and validation sets and the other a 50-50 real to fake image distribution subset of a different related dataset. As a result, I compare each modification model's testing accuracy on both test datasets and final validation accuracy to that of its respective baseline to see if there is an improvement. Furthermore, I inspect each model's training and validation curves to visualize the effect of each modification on each model's training process. Lastly, I check if the model is able to achieve greater than 50% accuracy on the 50-50 dataset. If a model's training process curves show little to no dramatic fluctuations and the model results in a final validation accuracy and testing accuracy that is better than that of its baseline, or equal to the baseline if the baseline accuracy is 100%, then I conclude that the modification is a possible avenue for additional improvement. Furthermore, if the model achieves greater than 50% accuracy on the 50-50 dataset, then I can conclude that the model is able to generalize on out-of-distribution data (OOD).

3.5. Modifications

Before given to any of the models, the dataset images are cleaned to remove any possible duplicate images using cw-somil's Duplicate Remover, where a duplicate image is identified by hashing the images and finding identical hash values, to improve generalization by preemptively preventing the model from overly learning features present in the duplicate images [2]. Then, the images are resized to a uniform 256×256 to fit the input requirements of the Reference Model and its derivatives. Lastly, the images are normalized to standardize the range of the image pixel values to improve model training efficiency and stability.

3.5.1 Learning Rate

For the learning rate, the Reference Model has a learning rate of 0.0001. Since improvements to validation and testing accuracy are the measure of success of a modification for this paper and the number of epochs is 10 due to resource constraints, I test 10x the learning rate to 0.001 and 100x the learning rate to 0.01. These modifications are done to see if the higher learning rate allows the models to improve accuracy by finding a better local minimum faster.

Dataset	Real Image		Fake Image		Total
	#	%	#	%	#
Duplicates	0	0.00	0	0.00	0
Training	470	45.59	561	54.41	1031
Validation	61	47.29	68	52.71	129
Test	58	44.96	71	55.04	129
Original	589	45.69	700	54.31	1289

Table 1. Breakdown of the Fake-Vs-Real Faces (Hard) dataset for the training and validation of this paper's models.

Dataset	Real Image		Fake Image		Total
	#	%	#	%	#
Duplicates	104	66.67	52	33.33	156
Test	129	50.00	129	50.00	258
Original	5413	49.64	5492	50.36	10905

Table 2. Breakdown of the deepfake and real images dataset for the testing of this paper's models.

3.5.2 Additional Layers

For the additional layers, I add a Conv2D, reLU, and MaxPooling2D layer group to the model to see if allowing the model to learn more complex, hierarchical features from the data allows the model to detect patterns that significantly helps it improve its validation and testing accuracy. I choose these layers because the Reference Model applies this ordered combination of layers 6 times before flattening into dense layers, so adding a 7th is of interest.

3.5.3 Image Transformation Pre-Processing

For the image transformation pre-processing, I apply random horizontal flip, random vertical flip, random rotation of 15 degrees, and color jitter transformations to the training set of the models. This is done to test if these manipulations increase the model's ability to generalize.

4. Data

The main dataset chosen for this project is the "Fake-Vs-Real-Faces (Hard)," by Hamza Boulahi, from Kaggle. This dataset is a part of the public domain and contains two image classes: real images and fake images (Fig. 1). There are 589 images of real human faces and 700 images of fake human faces of uniform 300×300 size in JPEG format for a 54.31% fake, 45.69% real total image distribution. The real images of human faces were gathered through the Unsplash website's API and cropped out using the OpenCV Library.

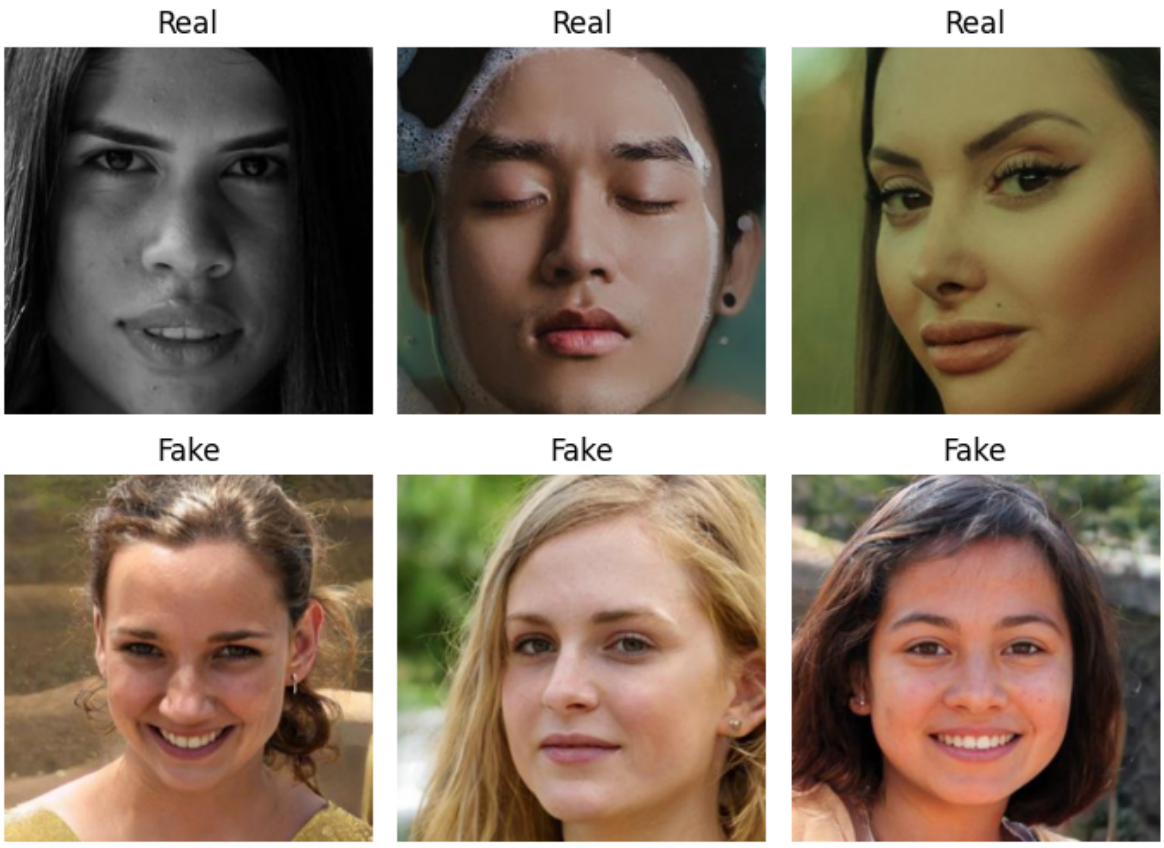

Figure 1. Example real and fake images from the Fake-Vs-Real Faces (Hard) dataset.

They were collected to have a fair representation of a variety of human features and characteristics like age, makeup, sex, and ethnicity. To increase the difficulty of correct classification, the fake images of human faces were sourced from https://thispersondoesnotexist.com/, whose images are generated using StyleGAN2 [1]. As a result of the fake images being sourced from https://thispersondoesnotexist.com/, the models of this paper trained on these images contain the biases and limitations of the website's image generation model.

After having no duplicates removed with cw-somil's Duplicate Remover and all images resized to 256×256 and normalized, the dataset is randomly shuffled before being separated into 80% train, 10% validation, and 10% test datasets for the training, validation, and testing of the models for this paper [2]. For each dataset, the image count for each class and its distribution percentage can be seen in (Table 1).

The secondary dataset chosen to test all of the models is the "deepfake and real images" dataset by Manjil Karki from Kaggle [6]. It is a processed dataset of the images from "OpenForensics: Multi-Face Forgery Detection And Segmentation In-The-Wild Dataset [V.1.0.0]" by Trung-Nghia Le, Huy H Nguyen, Junichi Yamagishi, and Isao Echizen, such that the images are 256×256 .jpg images. The original image dataset, OpenForensics, was designed for deepfake prevention and general human face detection purposes [7]. Of the Kaggle dataset, only a subset of the test folder images were selected for the test dataset for the models after all duplicate images were removed by cw-somil's Duplicate Remover (Fig. 2). Specifically, 129 files were selected from the real images folder and fake images folder each, for a total of 258 images and a 50-50 fake to real image distribution, after 104 and 52 duplicate images were removed from the real images and fake images folder respectively (Table 2). This secondary test dataset will be referred to moving forward as the "50-50 Dataset" to differentiate it from the test dataset subset from the main dataset.

5. Experiments and Results

All models were trained in Google Colab, using the free resources available with a Google account. Furthermore, improvements to the testing accuracy on both test datasets and final validation accuracy of the models are the core

Figure 2. Example real and fake images from the deepfake and real images test dataset.

measure of performance and modification success.

5.1. Baseline

Like the Reference Model, the Keras Reference and PyTorch Conversion were given a learning rate of 0.0001 and the Adam Optimizer as their optimizer. As seen in Table 3, both the Keras Reference and PyTorch Conversion resulted in training and validation accuracy greater than that of the Reference Model when trained and validated with the new training dataset and validation dataset. While both models also saw higher training loss and lower validation loss, I conclude that the Keras Reference is a successful transition of the Reference Model because the Keras Reference was given a fraction of the amount of images and epochs that the Reference Model had access to, namely 9,000 images in the Reference Model dataset and 150 epochs, so it is expected to have higher loss. Furthermore, in combination with the PyTorch Conversion achieving results within the 2% tolerance of the Keras Reference, I conclude that the PyTorch Conversion also successfully replicates the Reference Model. Therefore, both models can serve as baselines for the following modifications.

5.2. 10x Learning Rate

Both the Keras and PyTorch variant models with 10 times the learning rate from 0.0001 to 0.001, labeled 10xLR in their respective tables (Table 4, Table 5), performed worse than their baselines. The Keras 10xLR had similar training accuracy of 100%, but saw decreases of 0.77% in validation accuracy, 1% in testing accuracy on the test dataset, and 6% in testing accuracy on the 50-50 Dataset with higher loss compared to the Keras Reference baseline. Similarly, the PyTorch 10xLR saw a decreases of 0.78% in validation accuracy, 2% in testing accuracy on the test dataset, and 5% in testing accuracy on the 50-50 Dataset with higher loss compared to the PyTorch Conversion baseline. Furthermore, the model also had a 0.29% decrease in training accuracy compared to the baseline. Since both models had lower testing accuracy on the 50-50 Dataset, the 10xLR variant models likely learned features that are only characteristic to the Fake-Vs-Real (Hard) dataset fake images, such that it is unable to accurately identify other kinds of fake images. As a result, increasing the learning rate by a factor of 10 failed as an optimization, indicating a learning rate of 0.0001 is more optimal for training the model.

Criterion	Reference Model	Keras Reference	PyTorch Conversion
Fake-Vs-Real (Hard)			
Training Accuracy (%)	100*	100	100
Validation Accuracy (%)	95.21*	99.22	100
Training Loss	0.000005*	0.000874	0.003613
Validation Loss	0.2906*	0.0205	0.0034
Testing Accuracy (%)	95*	99	100
deepfake and real images			
Testing Accuracy (%)	N/A	51	53

Table 3. Results of the Reference Model and baseline models, where Reference Model results are that of the reference paper (marked *).

Criterion	Keras Reference	10xLR	100xLR	Additional Layers	Image Transforms
Fake-Vs-Real (Hard)					
Training Accuracy (%)	100	100	54.41	100	99.13
Validation Accuracy (%)	99.22	98.45	53.71	99.22	99.22
Training Loss	0.000874	0.007879	0.689654	0.001556	0.021313
Validation Loss	0.0205	0.1320	0.6927	0.0126	0.0229
Testing Accuracy (%)	99	98	55	99	99
deepfake and real images					
Testing Accuracy (%)	51	45	50	49	53

Table 4. Results of the Keras models.

Criterion	PyTorch Conversion	10xLR	100xLR	Additional Layers	Image Transforms
Fake-Vs-Real (Hard)					
Training Accuracy (%)	100	99.71	98.06	99.61	100
Validation Accuracy (%)	100	99.22	99.22	99.22	95.35
Training Loss	0.003613	0.010666	0.052932	0.009069	0.002562
Validation Loss	0.0034	0.0066	0.0124	0.0324	0.1730
Testing Accuracy (%)	100	98	98	100	92
deepfake and real images					
Testing Accuracy (%)	53	48	50	48	53

Table 5. Results of the PyTorch models.

5.3. 100x Learning Rate

Like the 10xLR models, both the Keras and PyTorch variant models with 100 times the learning rate from 0.0001 to 0.01, labeled 100xLR in their respective tables (Table 4, Table 5), performed worse than their baselines, but to a larger degree. The Keras 100xLR saw a large decrease in its overall performance when compared to the Keras baseline with decreases of 45.59% in training accuracy, 45.51% in validation accuracy, 44% in testing accuracy, and 1% in testing accuracy on the 50-50 Dataset with higher loss. In contrast, the PyTorch 100xLR saw a less steep decline of 1.94% in training accuracy, 0.78% in validation accuracy, 2% in testing accuracy, and 3% in testing accuracy on the 50-50 Dataset compared to the PyTorch Conversion baseline with saw higher loss. Since both models had testing accuracy of 50% on the 50-50 Dataset, this indicates that their generalization ability is similar to that of a coin toss on unseen data that is not closely similar to that of which it was trained on. Therefore, increasing the learning rate by a factor of 100 failed as an optimization. These results support the conclusion that a learning rate of 0.0001 is more optimal for training the model and that increasing the learning rate further is unlikely an avenue for additional improvement.

5.4. Additional Layers

Labeled Additional Layers in their respective tables, (Table 4, Table 5), the Keras Additional Layers model saw slightly improved results while the PyTorch Additional Layers model saw slightly worse results after a 7th convolution 2D, reLU, and maxpooling 2D layer group was added to the baseline model architecture. Specifically, the Keras Additional Layers model had the same rounded training accuracy, validation accuracy, and testing accuracy on

the test dataset as the Keras Reference with higher training loss and lower validation loss. That being said, it performed 2% worse on the 50-50 Dataset, which indicates that the additional layers might have overly trained the model on features present in the Fake-Vs-Real (Hard) dataset images, such that it has poor generalization on OOD. Similarly, the PyTorch Additional Layers model performed 5% worse on the 50-50 Dataset, indicating a poor generalization on OOD. Furthermore, the model saw decreases of 0.39% in training accuracy and 0.78% in validation accuracy with higher loss, but similar testing accuracy as the PyTorch Conversion Baseline. Therefore, these results indicate that adding an additional convolution 2D, reLU, and maxpooling 2D layer group provides limited to no improvement to the model.

5.5. Image Transformation Preprocessing

Labeled Image Transforms in their respective tables, (Table 4, Table 5), the Keras Image Transforms model saw roughly similar results to that of its baseline with higher testing accuracy on the 50-50 Dataset while the PyTorch Image Transforms model saw worse results than its baseline with similar testing accuracy on the 50-50 Dataset. Specifically, the Keras Image Transforms model had same rounded validation accuracy and test accuracy on the test dataset as its baseline with a 0.87% decrease in training accuracy and higher loss, but it achieved a 2% increase in testing accuracy on the 50-50 Dataset. The PyTorch Image Transforms model had same rounded training accuracy as its baseline, but it saw a 4.65% decrease in validation accuracy, lower training loss, higher validation loss, and 8% decrease in testing accuracy on the test dataset to its baseline. That being said, it achieved the same rounded 53% testing accuracy on the 50-50 Dataset. Since the Keras Image Transforms model achieved better performance on the 50-50 Dataset than its baseline while achieving similar accuracy and loss as its baseline and the PyTorch Image Transforms model achieved same testing accuracy on the 50-50 Dataset, preprocessing the images improves the model's ability to generalize. Furthermore, since the PyTorch model saw decreases in validation accuracy after the image transforms were applied, it indicates that the fake images included in the Fake-Vs-Real Faces (Hard) dataset are not significantly distorted by horizontal flips, vertical flips, random rotation, or color jitter.

6. Conclusion

In this paper, I test three possible optimizations of increasing the learning rate of, adding additional layers to, and image transformation pre-processing for the Keras Reference model on both Keras and PyTorch variants. From testing increasing the learning rate to 0.001 and 0.01, I found that both do not improve the model's accuracy or ability to generalize. Similarly, I found that adding a 7th iteration of a convolution 2D, reLU, and maxpooling 2D layer group to the model architecture also does not improve the model's ability to generalize, although its model results on the main dataset sees little change. In contrast, I found that adding image transformation pre-processing to the dataset of horizontal and vertical flips, random rotation, and color jitter does improve the model's ability to generalize, but lowers the model's validation and testing accuracy on the main dataset. Therefore, the results of this paper indicate that increasing the learning rate and adding a 7th iteration of the layer group fail as avenues for improving the Keras Reference model, while applying image transformation preprocessing to the dataset has merit in improving the Keras Reference Model.

6.1. Next Steps

In this paper, the PyTorch Conversion model had padding mistakenly set to 1. Since Keras's convolution 2D's default padding is set to 0, changing the PyTorch Conversion model's padding to 0 and testing the difference is an area for future research. Furthermore, testing splitting the general fake image category classification into deepfake and synthetic image categories is an area of possible future model improvement, due to the results of Naeem et. al., that was not able to be explored in this paper due to time constraints [11]. Lastly, providing a larger dataset size for the training and testing of the models over more epochs, to achieve greater similarity to that of the reference paper may lead to new findings [12].

References

[1] H. Boulahia. Fake-vs-real-faces (hard). https://www.kaggle.com/datasets/hamzaboulahia/hardfakevsrealfaces. Accessed: Feb. 7, 2025. 4

[2] Cw-Somil. Cw-somil/duplicate-remover: Repository to find duplicate images and similar images with the help of python. https://github.com/cw-somil/Duplicate-Remover, May 2020. Accessed: Feb. 7, 2025. 3, 4

[3] J. Philbin F. Schroff, D. Kalenichenko. Facenet: A unified embedding for face recognition and clustering, 2015. in Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition 2015. 1

[4] M. Mirza B. Xu D. Warde-Farley S. Ozair-A. Courville Y. Bengio Goodfellow, J. Pouget-Abadie. Generative adversarial networks, 2014. in arXiv preprint arXiv:1406.2661, 2014. https://arxiv.org/abs/1406.2661. 1

[5] N. Xue S. Zafeiriou J. Deng, J. Guo. Retinaface: Single-shot multi-level face localisation in the wild, 2019. arXiv preprint arXiv:1905.00641. 1

[6] Manjil Karki and Trung-Nghia Le. deepfake and real images. https://www.kaggle.com/datasets/manjilkarki/deepfake-and-real-images, 2022. Accessed: Apr. 27, 2025. 4

[7] Trung-Nghia Le, Huy H. Nguyen, Junichi Yamagishi, and Isao Echizen. Openforensics: Large-scale challenging dataset for multi-face forgery detection and segmentation in-the-wild. In *International Conference on Computer Vision*, 2021. 4

[8] S. Javed A. Shoufan Y. Zweiri M. Alansari, O. A. Hay and N. Werghi. Ghostfacenets: Lightweight face recognition model from cheap operations. *IEEE Access*, 11(3):35429–35446, 2023. 2

[9] Ethan Nguyen-Tu. Cs4644_finalproject. https://github.com/EthanNguyen-Tu/CS4644_FinalProject, April 2025. 1

[10] Federal Bureau of Investigation. Criminals use generative artificial intelligence to facilitate financial fraud. https://www.ic3.gov/PSA/2024/PSA241203, December 2024. Accessed: Feb. 7, 2025. 1

[11] M. Khan U. Tariq A. Dhall H. Al-Nashash S. Naeem, R. Al-Sharawi. Real, fake and synthetic faces – does the coin have three sides?, April 2024. under consideration at FG2024. 2, 7

[12] F. Salman and S. Abu-Naser. Classification of real and fake human faces using deep learning. *International Journal of Academic Engineering Research (IJAER)*, 6(3):1–14, 2022. (accessed Feb. 9, 2025). 1, 2, 7

[13] M. Ranzato L. Wolf Y. Taigman, M. Yang. Deepface: Closing the gap to human-level performance in face verification, 2014. in IEEE Conference on Computer Vision and Pattern Recognition (CVPR). 2