NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

A GENERALIZED ANALYTIC FOR THE DETECTION OF SYNTHETIC MEDIA

by

Patrick L. Reilly

June 2021

Thesis Advisor:	Robert L. Bassett
Second Reader:	Ross J. Schuchard

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*

1. AGENCY USE ONLY *(Leave blank)*	**2. REPORT DATE** June 2021	**3. REPORT TYPE AND DATES COVERED** Master's thesis

4. TITLE AND SUBTITLE A GENERALIZED ANALYTIC FOR THE DETECTION OF SYNTHETIC MEDIA	**5. FUNDING NUMBERS**
6. AUTHOR(S) Patrick L. Reilly	
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)

Convolutional neural networks (CNNs) and generative adversarial networks (GANs) have quickly become leading tools for the creation of convincing synthetic images. Such images increase the difficulty of discerning fact from fiction in the information space, where such challenges can degrade the quality and timeliness of decision-making. To compete, we must develop tools that can automatically detect artificially generated images. A major challenge in this area centers around the high number of unique image generation methods. We therefore seek a classification analytic that can successfully generalize when tested on images from multiple image generation algorithms. The 2020 paper "CNN-Generated Images Are Surprisingly Easy to Spot... For Now" by Wang et al. proposes such an approach. The study conducted here independently tests and validates this analytic in a variety of use cases. We begin by focusing on the reproducibility of the analytic using both publicly released and retrained models, the performance of the analytic on a dataset of images where generator type is unknown, and the analytic's effectiveness in the detection of traditional deepfakes. We also examine the analytic's robustness in response to reductions in image quality via compression and adversarial perturbations. Finally, we attempt to improve the analytic's performance by using a state-of-the-art generator to produce a new image training set.

14. SUBJECT TERMS image classification, machine learning, synthetic media, image generation, neural networks, convolutional neural networks, CNNs, generative adversarial networks, GANs, deepfakes	**15. NUMBER OF PAGES** 69
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	**18. SECURITY CLASSIFICATION OF THIS PAGE**	**19. SECURITY CLASSIFICATION OF ABSTRACT**	**20. LIMITATION OF ABSTRACT**
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

A GENERALIZED ANALYTIC FOR THE DETECTION OF SYNTHETIC MEDIA

Patrick L. Reilly
Captain, United States Marine Corps
BA, University of Virginia, 2011
MA, University of Virginia, 2012

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN OPERATIONS RESEARCH

from the

NAVAL POSTGRADUATE SCHOOL
June 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

Convolutional neural networks (CNNs) and generative adversarial networks (GANs) have quickly become leading tools for the creation of convincing synthetic images. Such images increase the difficulty of discerning fact from fiction in the information space, where such challenges can degrade the quality and timeliness of decision-making. To compete, we must develop tools that can automatically detect artificially generated images. A major challenge in this area centers around the high number of unique image generation methods. We therefore seek a classification analytic that can successfully generalize when tested on images from multiple image generation algorithms. The 2020 paper "CNN-Generated Images Are Surprisingly Easy to Spot... For Now" by Wang et al. proposes such an approach. The study conducted here independently tests and validates this analytic in a variety of use cases. We begin by focusing on the reproducibility of the analytic using both publicly released and retrained models, the performance of the analytic on a dataset of images where generator type is unknown, and the analytic's effectiveness in the detection of traditional deepfakes. We also examine the analytic's robustness in response to reductions in image quality via compression and adversarial perturbations. Finally, we attempt to improve the analytic's performance by using a state-of-the-art generator to produce a new image training set.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

THIS PAGE INTENTIONALLY LEFT BLANK

List of Figures

THIS PAGE INTENTIONALLY LEFT BLANK

List of Tables

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

AP	average precision
CIELAB	CIE L*a*b*
CRN	cascaded refinement network
CNN	convolutional neural network
DFDC	Deepfake Detection Challenge
DFDD	Deepfake Detection Dataset
DOD	Department of Defense
GAN	generative adversarial network
IMLE	implicit maximum likelihood estimation
LSUN	Large Scale Scene Understanding
mAP	mean average precision
NPS	Naval Postgraduate School
ReLU	rectified linear unit
SAN	second-order attention network
SITD	See-in-the-Dark
SISR	single image super-resolution
WFIR	Which Face is Real

THIS PAGE INTENTIONALLY LEFT BLANK

Executive Summary

Convolutional neural networks (CNNs) and generative adversarial networks (GANs) have quickly become leading tools for the creation of synthetic images. A significant challenge in the detection of these artificial images centers around the high number of unique CNN-based image generation methods now available, each of which is algorithmically distinct, with its own set of strengths and weaknesses as to the types of images it produces. Organizations and agencies whose duties include media authentication must possess tools that enable the accurate classification of suspected synthetic images without prior knowledge of the images' provenance. We therefore seek a classification analytic that can successfully generalize when tested on images from multiple image generation algorithms.

The paper *CNN-generated images are surprisingly easy to spot... for now* by Wang et al. (2020) proposes such an approach. Using training images from a single CNN model with real images as negative examples, the authors claim to achieve a minimum of 88.2 percent average precision against 10 different CNN-based image generators, as well as traditional deepfake face replacements (Wang et al. 2020). If authentic and reproducible, these results would represent a significant breakthrough in the fight against manipulated imagery.

In this study, we seek to understand and validate the algorithm proposed in Wang et al. (2020) by examining the assumptions of the paper and attempting to recreate the paper's results. After testing both publicly released models and retraining the models described in the paper, we find that while we were able to reproduce many of the paper's results, the analytic is not a reliable detector of deepfake face swaps. We also create a new dataset to test the analytic's performance in a simulated real-world scenario where the analytic must classify images of unknown provenance.

To further understand the strengths and weaknesses of the proposed analytic, we also evaluate the analytic's performance against four new deepfake datasets and assess the analytic's robustness to image compression and adversarial perturbations. Finally, we attempt to improve the performance of the analytic on high-resolution images and deepfakes by retraining its underlying models on StyleGAN2, a new state-of-the-art image generator.

List of References

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.

CHAPTER 1:
Introduction

This thesis is a reproducibility study of the paper *CNN-generated images are surprisingly easy to spot...for now* by Wang et al. (2020). The authors of this paper propose an analytic based on a convolutional neural network (CNN) for the classification of images as real or synthetic. When trained on images from a single artificial image generator, the analytic is reported to successfully classify images from a variety of different image generation algorithms. In this thesis, we attempt to independently reproduce the authors' results, extend the analytic's performance to additional datasets, evaluate the analytic's robustness to compression and adversarial perturbations, and improve the analytic's performance by retraining the analytic on images from a new, state-of-the-art image generator.

1.1 Motivation

CNNs have quickly become leading tools for the creation of synthetic images. A significant challenge in the detection of these artificial images centers around the high number of unique CNN-based image generation methods now available, each algorithmically distinct with its own set of strengths and weaknesses as to the types of images it produces. Researchers have successfully trained classifiers capable of detecting images generated by individual CNN-based algorithms, but such classifiers have been shown to perform unsatisfactorily when tested against similar images created by different methods (Wang et al. 2020). We therefore seek a classification analytic that can successfully generalize when tested on images from multiple image generation algorithms.

The paper independently evaluated in this thesis proposes such an approach. Using training images from a single CNN model with real images as negative examples, the authors claim to achieve a minimum of 88.2 percent average precision (AP) against 10 different CNN-based image generators, as well as traditional deepfake face replacements. If authentic and reproducible, these results would represent a significant breakthrough in the fight against manipulated imagery.

1.2 Relevance

Amplified in their influence by global social media networks, manipulated and artificial images and videos may soon impact domestic and international politics on a regular basis. In May 2019, a digitally manipulated video of House Speaker Nancy Pelosi was posted to several political Facebook pages, which appeared to show the House Speaker speaking incoherently in an onstage speech. Using only rudimentary editing techniques, the original video had been slowed down and re-rendered, with the audio track modified to match the video without a resulting loss in pitch. The video received wide distribution and at least 2 million views. A variation was shared by the sitting President of the United States on social media (Harwell 2019).

Elsewhere, in the central-African Gabonese Republic, a 2019 New Year's video address by an ailing President Ali Bongo was widely interpreted to be manipulated. Owing to perceived changes in the facial appearance of President Bongo, now known to be the result of a recent stroke, opposition figures claimed that the video featured the leader's face artificially superimposed onto a body double. Stoking fears of President Bongo's medical incapacitation and a subsequent digital cover-up, opposition leaders sparked an unsuccessful military coup (Cahlan 2020). These incidents exemplify the potential impact that manipulated images and videos may have in the domestic and international political spheres.

Moreover, good decision making requires access to timely, accurate information, which in turn requires the ability to quickly distinguish fact from fiction. Manipulated media threatens to diminish this ability, but by developing tools to detect artificially generated images we can better equip decision-makers with the information they need. Furthermore, by automating this detection process, we hope to compete with the rate at which artificial media is generated.

1.3 Technical Context

In the past decade, CNNs have emerged as a leading architecture for image analysis tasks, with applications that include both synthetic image generation and detection. In generative applications, neural network models essentially seek to approximate probability distributions by training on a set of representative samples from such distributions (Ruthotto and

Haber 2021). In the context of synthetic image generation, these samples consist of sets of images with common features that a user might seek to replicate artificially. For example, a user seeking to create synthetic images of human faces might train models on a set of real portraits.

While some image generation techniques employ non-adversarial methods to create synthetic images, an increasingly common approach is to employ a generative adversarial network (GAN). This architecture features a generative network and a discriminative network working in parallel. The generator attempts to construct images that fool the discriminator, while the discriminator attempts to determine if the generated images came from a training dataset or from the generator (Goodfellow et al. 2014). Insights from the discriminator are shared with the generator, theoretically allowing the generation of increasingly convincing synthetic images over time.

Another related and emerging technique for artificial image generation concerns *deepfakes*[1] in which two neural network-based autoencoders replace the face of an individual with that of another person. Unlike other generative techniques which create entirely new synthetic images, deepfakes replace only the facial region of a source image or video, leaving the rest of the source image or video in its original form.

The growth in research into synthetic image generation approaches has also sparked parallel research into techniques for detecting such images, and a variety of detection techniques have been proposed in the last five years. To detect fake GAN-based images, Zhang et al. (2019) propose a classifier model that does not rely on a dataset of synthetic images for training. Zhang et al. (2019) construct a synthetic image simulator that generates spectral artifacts similar to those produced by the upsampling layer of many GANs, and then overlays these artifacts onto real images. The researchers then train a classifier on the frequency spectra of their real and simulated synthetic images. The classifier demonstrates high accuracy against real and fake images from the dataset used to train their GAN simulator but fails to generalize when shown images from other datasets (Zhang et al. 2019).

[1]**A note on terminology**: Some organizations and publications use the term *deepfake* to refer to any synthetic image or video created using a deep learning approach. Here, we use the definition of *deepfake* in its most specific form to refer only to those images and videos in which the face of an individual is replaced with the face of a second individual through an autoencoder-based face swap. This convention matches that used in the documentation for our analytic of study.

Cozzolino et al. (2018) create a classifier based on an autoencoder architecture that learns forensic information such as high-frequency noise from an initial training set and preserves this information in the latent space alongside the data required for the classifier's prediction. The researchers then show their classifier a small sample of images from a new target image generator from which they wish to classify images. Cozzolino et al. (2018) achieve promising accuracy results, but the requirement of their classifier to see images from new datasets in advance before generalizing, along with the limited pairings of training and testing generators included in their study, restrict the utility of their classifier in areas where broad generalization across image generators is required (Cozzolino et al. 2018).

Other research teams have proposed methods which specifically focus on detecting synthetic human subjects. Wang et al. (2019) propose an analytic which uses the neuron behavior in the layers of a third-party facial recognition system as the input to a shallow binary classifier. This technique contrasts with the image classifiers above, which rely on image pixel data or frequency spectra inputs. The classifier exhibits strong performance against several GAN-based image generators and two deepfake datasets, but its reliance on facial recognition software prohibits generalization beyond images with human subjects and visible facial regions (Wang et al. 2019).

More recently, Liu et al. (2020) have proposed a classifier that relies on the differences in the texture data of real and synthetic images to make predictions. The researchers build upon a ResNet model by incorporating an additional layer to capture global texture data and then test their classifier on a variety of GAN-based datasets. The classifier shows promising performance, but as above, training and testing dataset pairings are limited, and the researchers restrict their test datasets to those featuring human faces (Liu et al. 2020).

Guarnera et al. (2020) construct a classifier using an expectation maximization technique to record the local correlation between pixels in images from several common GAN-based generators, under the assumption that the final convolutional layers of such generators produce unique sets of correlations. The classifier can detect differences between real and fake images across several image GAN-based generators and can also correctly label the generator that produced a given fake image in a two-class dataset of synthetic images. However, the classifier must be trained on images from multiple generators to be effective, and the testing conducting by the researchers was limited to datasets featuring human faces.

Finally, while this report was in draft, Frank and Holz (2021) published a second independent reproducibility study of the analytic proposed by Wang et al. (2020). Our report and that completed by Frank and Holz (2021) have significant methodological differences and areas of focus. Frank and Holz (2021) reconstruct the analytic without the use of the authors' published code base, and focus on evaluating two specific claims regarding the impact of training set diversity and image augmentations at training time on the analytic's performance (a full summary of the analytic is included in Chapter 3). Despite the differences in our reports, we observe some common findings about the efficacy of the analytic, and taken together, these reports illustrate the high priority of research into this field.

In this chapter, we have briefly summarized the analytic of interest, provided motivation for our reproducibility study, and shown that this analysis has social, military, and political relevance. In the next chapter, we discuss the supporting technical concepts behind the analytic, and explore the image generators and datasets used to train and test the analytic's classifiers.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 2:
Supporting Concepts and Data

In Chapter 2, we define several machine learning concepts which are critical to the understanding of the analytic of study. We also document each of the image datasets used to train and test the analytic's proposed classifier.

2.1 Supporting Concepts

Chapter 1 briefly introduced the concepts CNNs and GANs in the context of image analysis. In this section, we define these concepts in greater detail to support the discussion of the analytic in subsequent chapters.

2.1.1 Neural Networks

Neural networks are a class of machine learning algorithms capable of modeling complex, nonlinear relationships. In the most general sense, a neural network seeks to model a true relationship f^* between an input $\mathbf{x}$ and an output $\mathbf{y}$. By learning a set of parameters $\boldsymbol{\theta}$, the neural network defines a new function f such that $\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta})$ models the true relationship f^* with the highest possible fidelity (Goodfellow et al. 2016).

In practice, neural networks are usually a composite of many different functions which together accept an input and produce an output. Each function within this composite will have a set of different parameters that must be learned and optimized by the model. We can visualize this composite as an acyclic graph with three or more vectors of nodes, with each vector representing a single function. We define each function or vector of nodes as a **layer** of the neural network; we further define each node within a layer as a **neuron**. A neural network must possess, at a minimum, an input layer, an output layer, and an internal (or **hidden**) layer (Goodfellow et al. 2016). The output of a network with no hidden layers amounts to a linear transformation of the input layer, and cannot model nonlinear relationships. Neural networks with a single hidden layer are referred to as **shallow**, while those with two or more hidden layers are defined as **deep**. A layer with neurons that receive

connections from every neuron in the previous layer is said to be **fully connected**. Figure 2.1 illustrates two basic networks with fully connected layers in graph form.

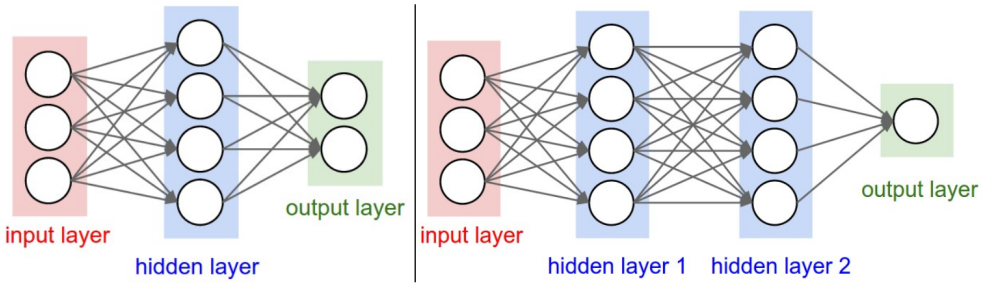


Figure 2.1. An illustration of two simple neural networks. Left: A neural network with a single hidden layer and a two-dimensional output. Right: A neural network with two hidden layers and a one-dimensional output. Source: Li et al. (2021b).

Each layer of a neural network consists of one or more nodes, or **neurons**. Each neuron is simply a cell in which the network stores a numeric value. The activation of a given neuron is determined by several operations on its inputs which are built into the design of the network or optimized at training time. Returning to Figure 2.1, we see that in both models, the values in each neuron are passed as input to neurons in the next layer via pathways called **weights**. Weights are scalars that can increase or decrease the influence of the inputs to a neuron on its eventual value; the initial value passed to a neuron is a linear combination of the input values from the previous layer and their weights. A **bias** offset is sometimes added to the end of this linear combination to allow for more granular control of the final value passed to a neuron (Li et al. 2021b).

The value passed to a neuron undergoes a final transformation by an activation function to allow the modeling of nonlinear relationships. Common activation functions include the sigmoid and rectified linear unit (ReLU) functions. The sigmoid activation function computes $S(x) = \frac{1}{1+e^{-x}}$, while the ReLU function computes $R(x) = \max(0, x)$. Different activation functions can be applied at different layers of a neural network as part of its design, while weights and biases are optimized during model training. Activation functions are not generally applied to the output layer of a neural network (Li et al. 2021b). In summary, we can represent the value of a given neuron y with respective vectors of inputs $\mathbf{x}$ and weights $\mathbf{w}$, bias b, and sigmoid activation function S as

$$y = S(\mathbf{x}^T\mathbf{w} + b). \tag{2.1}$$

Due to the inherent nonlinearity of the relationships that neural networks often model, loss functions for optimization of the network tend to be non-convex, making linear solvers and techniques that seek global convergence inappropriate for this application (Goodfellow et al. 2016). For this reason, training algorithms for neural networks seek to minimize a loss function at the output layer via gradient descent without a guarantee of achieving a global minimum. Stochasticity is introduced by calculating the loss function using only limited samples from the training set at each iteration (Li et al. 2021c). The size of the sample shown to the network during a given iteration is called the **batch size**, while the number of times the network sees the whole training set is defined as an **epoch**. Finally, a **learning rate** dictates the step size taken in the direction of steepest descent for the loss function at each iteration. Effective neural network designs require careful selection of the loss function to balance computational efficiency and modeling performance during training (Li et al. 2021c).

In Section 2.1.2, we extend the definitions introduced here to CNNs, which form the computational backbone of many common image analysis tasks.

2.1.2 Convolutional Neural Networks

CNNs are a subclass of deep neural networks inspired by the biology of animal vision that have exhibited strong performance when applied to the analysis of imagery and other types of data where information can be specified as an array. CNNs resemble the neural networks described in Section 2.1.1, but with restructured connections between layers and neurons (Li et al. 2021a). In applications like imagery analysis, using pixel data as input to a fully connected neural network quickly becomes computationally intractable, with the required numbers of weights expanding exponentially with each hidden layer. The architecture of CNNs treats the arrangement of neurons volumetrically in three dimensions. This approach makes sense for imagery analysis, since we can think of an image as a multidimensional array (or **tensor**) of pixel data with width and height dimensions equal to the image's resolution, with a depth equivalent to the number of color channels: red, blue, and green (Goodfellow et al. 2016). An example of such a tensor containing pixel data by color is

shown in Figure 2.2.

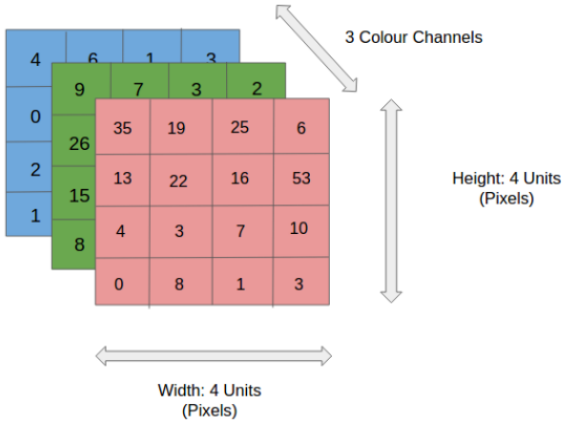


Figure 2.2. An illustration of an RGB image tensor. Color information for the image is spread across three individual arrays. Source: Saha (2018).

Again referencing the networks described in Section 2.1.1, CNNs require an input layer and output layer, and in most cases, several fully-connected layers. However, CNNs' use of convolutional layers and pooling layers set them apart from simple neural networks (Li et al. 2021a). The values of a convolutional layer are derived from a weighting filter (or **kernel**). The kernel is generally small in width and height relative to the input, but its depth is equal to that of the input. For example, if evaluated on the RGB tensor above, the kernel would also have a depth dimension of three. During the evaluation of an input, the kernel slides across the width and height of the input much like a rectangular magnifying glass over a block of text. This sliding operation is called a **convolution**. During a convolution, the kernel computes the inner product of the kernel weights and its input values at each step, producing a **feature map** that represents this computation at each position occupied by the kernel. Since the kernel also applies weighting over the depth of the traversed array, a feature map is created for each depth dimension of the input (Li et al. 2021a). Figure 2.3 shows a convolution over a 3x4 input.

Convolutional operations allow the network to "learn" specific features of the input, with successive layers allowing increasing refinement in the features learned. Convolutional layers may also be interspersed with **pooling** layers. As an input traverses a network, pooling layers at critical junctures allow controlled downsampling to reduce the dimensionality of the data, reduce computational complexity, and prevent overfitting (Li et al. 2021a).

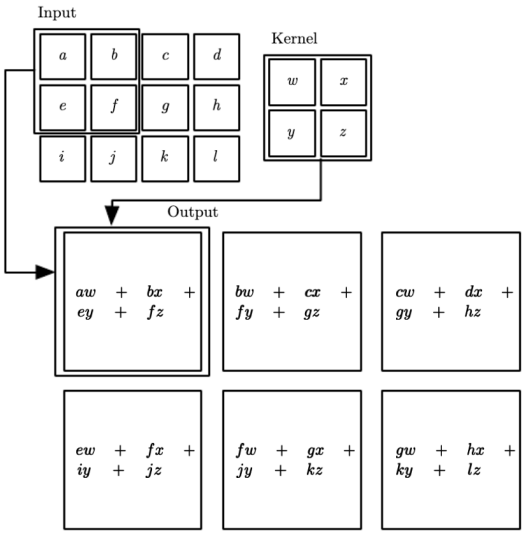


Figure 2.3. An illustration of a single CNN convolution. A 2x2 kernel operates on a 3x4 array with stride 1 (i.e., slides one unit at a time), producing a 2x3 output. Source: Goodfellow et al. (2016).

CNNs have applications in both generative and discriminative contexts. Generative models fundamentally seek to learn approximations of complex probability distributions by training on representative examples; these models then generate new samples that resemble those from the training sample (Ruthotto and Haber 2021). For example, a CNN-based image generator may be trained on images of planes and learn to reproduce samples from an approximation of the probability distribution that governs the pixel data in such images. Discriminative models are useful for tasks like classification. In these models, the network is again exposed to examples of a probability distribution during training, but given an input, the trained model reports a class score estimating the conditional probability of membership of the input in the training distribution. In image analysis, classification models can report a binary classification score, or report probabilities of membership in a specific image class. In the former case, we might assess whether an input image is real or synthetic; in the latter, we might test for an image's class membership to a specific subject category, such as planes or helicopters. In the next section, we examine the GAN architecture, which relies on both generative and discriminative networks in support of realistic image generation.

2.1.3 Generative Adversarial Networks

In Section 1.3, we briefly introduced the concept of GANs, an increasingly common generative architecture. We now explore this concept in greater detail. Like other generative models, GANs seek to create new samples from a learned approximation of a probability distribution. However, GANs use two neural networks working together to circumvent the limitations of traditional loss functions that would compare the training data to the generated samples (Goodfellow et al. 2016). We will use the example of an architecture designed to create synthetic images throughout this section.

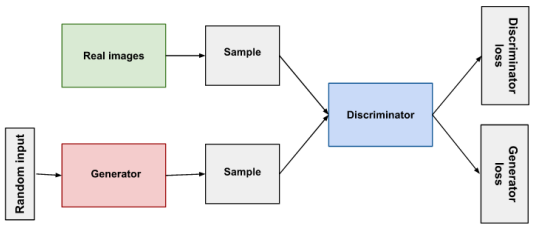


Figure 2.4. An illustration of a simple GAN architecture. A discriminator is initially trained on real images, while a generator initializes with random inputs. Generated data is passed to the discriminator, whose classification outputs influence both generator and discriminator loss. Source: Google Developers (2019).

In a GAN architecture for image generation, a generator is paired with a discriminator in an adversarial game. We can think of the discriminator as a binary classifier. Prior to the start of the game, the discriminator is trained using examples of the target class of images to be generated, which are used as positive training examples. At the beginning of the game, the generator will produce only images of random noise, which are then passed to the discriminator for classification. The discriminator also augments its original training data with these synthetic images produced by the generator, which are coded as negative examples (Google Developers 2019). An example of this architecture is shown in Figure 2.4. After the discriminator has assessed a generated image, the output of the discriminator is returned to the generator, which is penalized for synthesizing an image that the discriminator classified as not belonging to the target distribution. Likewise, once the generator begins producing images that fool the discriminator, the discriminator is penalized for incorrectly classifying the images. Training generally concludes when the discriminator's performance amounts to random guessing, or 50 percent accuracy (Google Developers 2019). Synthetic

image generators frequently rely on GAN-based architectures; the majority of the synthetic testing images used by the analytic of interest were created using GAN-based generators. In the next section, we explore one final technological concept required to begin detailed discussion of the analytic.

2.1.4 Deepfakes

Deepfakes are an emerging class of synthetic media where the face of an individual is replaced with the face of another person in an image or video. Deepfakes have many potentially damaging applications in the social, political, and military information space. As such, academic institutions, government agencies, and media corporations have become increasingly interested in developing technologies to combat deepfakes.

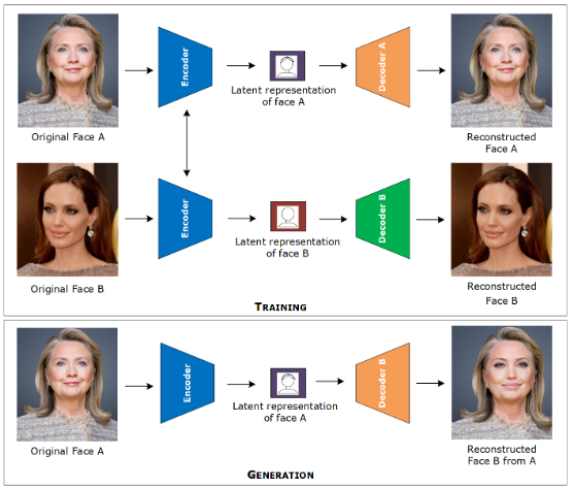


Figure 2.5. An illustration of the encoder/decoder training process and the creation of a deepfake. Note that the target face is passed to the source's decoder for reconstruction. Source: Masood et al. (2021).

While image and video face swaps have been possible for many years with traditional editing software, the proliferation of open-source deepfake generation tools has enabled the creation of such manipulations with greater speed and scale. While there now exist a variety of techniques to create convincing deepfakes, the datasets subsequently explored here rely on training autoencoder networks to perform the desired swap. An autoencoder is a type of neural network designed to recreate an input in a lower-dimensional space. In the

context of deepfakes, an autoencoder trained on images of a given individual's face learns to reconstruct an approximation of that face in a lower dimension (Zucconi 2018).

Deepfake creation techniques based on autoencoders generally require the training of at least two such networks. The first, called the **encoder**, learns to represent the faces of both the source individual and the target with respect to the shared features between both faces (Zucconi 2018). Separately, a **decoder** learns only the features of the source individual. To create the deepfake, an approximation of the target individual's face is reconstructed using the source individual's decoder and overlaid on the source. In a successful implementation of this technique, the pose and movements of the overlaid target face will match those of the source (Zucconi 2018). Different deepfake creation algorithms may also employ various types of processing on the source image or video and final output to produce deepfakes of higher visual quality. Figure 2.6 shows a general procedure for the creation and processing of deepfakes.

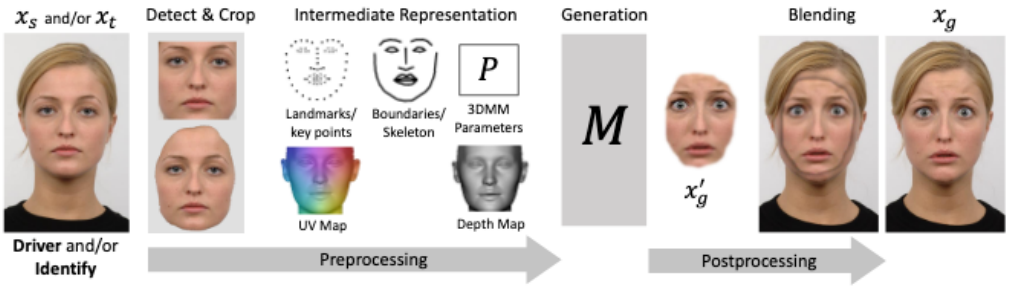


Figure 2.6. A general process for the creation of deepfakes. The source face is detected and cropped, and may undergo additional preprocessing to inform generation of the deepfake. The target face is then superimposed on the source, and may undergo additional post-processing to enhance the visual quality of the final output. Source: Mirsky and Lee (2021).

In this section, we described some of the necessary technical concepts required to introduce the analytic. In Section 2.2, we introduce the diverse array of synthetic image generators and datasets used to train and test the analytic.

2.2 The Data

The analytic of interest relies on a large corpus of image datasets for training and testing of its classifier models. Each dataset consists of both real and synthetic images, with the

naming convention of the dataset derived from the name of the generator used to create the synthetic images. These generators can be arranged into several broad categories, each of which is explored the following sections. With the exception of ProGAN, all datasets described here are used for classifier testing only.

2.2.1 Unconditional GANs

Image generators based on unconditional GANs attempt to recreate representative examples from the image dataset used for training. For example, an unconditional GAN successfully trained on real images of cars would learn to create synthetic images of cars. In this section, we explore four different unconditional GAN models used for training or testing in the analytic. Figure 2.7 shows representative real and synthetic examples from each dataset.

Figure 2.7. Representative images from the unconditional GAN datasets. From left to right, image classes are: *car*, *car*, *cat*, *dog*. Images from Wang et al. (2020) Git repository and derived from Karras et al. (2017, 2019, 2020b); Brock et al. (2018).

ProGAN

The ProGAN dataset is derived from an architecture proposed by Karras et al. (2017), and serves several important functions in the analytic of interest. The ProGAN architecture begins as a conventional GAN which is initially exposed to low-resolution training images.

As training progresses, the network is introduced to images of increasing resolution, and the generator and discriminator grow together through the addition of new layers. This process allows the network to learn high-level distributional information about the training dataset in the initial layers; as new layers grow the network learns refined features without significant disruptions to the already-optimized initial layers (Karras et al. 2017).

To create the dataset used in the analytic, Wang et al. (2020) generate synthetic images from 20 object categories defined by the Large Scale Scene Understanding (LSUN) dataset (Yu et al. 2015) and draw an equivalent number of real images from the same categories. Wang et al. (2020) then divide these images into three subsets, each with an equal proportion of real and synthetic images: a training set consisting of 720,000 images, a validation set with 4,000 images, and a test set of 8,000 images. All images are 256x256 pixels.

StyleGAN

StyleGAN is an image generator proposed by Karras et al. (2019) that relies on techniques inspired by neural style transfer concepts. The StyleGAN architecture omits an input layer in favor of a constant input learned in training. Inputs to the network are instead mapped to an intermediate latent space, which controls adjustments to certain features of the generated image at the convolutional layers of the network (Karras et al. 2019). This design enables macro-level features like camera orientation and subject identity to be combined with variable details like the superficial appearance of the subject (Karras et al. 2019). Wang et al. (2020) sample their synthetic StyleGAN images from publicly released models based on the LSUN bedroom, car, and cat object classes, with real images sourced from the training set of each model. Images based on the bedroom and cat classes are 256x256 pixels, while those from the car class are 512x384 pixels (Wang et al. 2020) The total dataset size is 12,000 images.

StyleGAN2

StyleGAN2 is an update to StyleGAN from Karras et al. (2020b) that makes several changes to the network architecture to address frequently occurring image artifacts in StyleGAN-generated images. Karras et al. (2020b) introduce additional regularization and normalization processes in their generator to reduce the number of undesired artifacts and allow the creation of more convincing synthetic images. Wang et al. (2020) construct their Style-

GAN2 dataset using publicly released models based on the cat, car, church, and horse LSUN classes, with real images again drawn from the training set of each model. All image resolutions are 256x256 pixels except for the car class, which is again 512x384 pixels (Wang et al. 2020). Due to the timing of the StyleGAN2 release, Wang et al. (2020) were only able to conduct limited testing with this dataset. In this report, we include a more comprehensive test of the analytic against this generator.

2.2.2 Which Face Is Real

To simulate a dataset and test their analytic on a dataset of real and fake images that might be found in on the internet or social media, Wang et al. (2020) extract 1,000 real and fake images from *whichfaceisreal.com* (West and Bergstrom 2019). *whichfaceisreal.com* features pairings of real and synthetic human portraits, with visitors to the site asked to select the real image for each pairing. Fake portraits are created using the StyleGAN generator from Karras et al. (2019), with real images sourced from the StyleGAN model's training set. All images on *whichfaceisreal.com* undergo JPEG compression with unspecified quality settings. Wang et al. (2020) only test a single classifier model on this dataset. In this study, we conduct a more comprehensive test.

BigGAN

BigGAN is an image generation architecture developed by Brock et al. (2018) that employs significantly more hyperparameters and a much larger batch size than its competitors, with a goal of increasing the visual quality and subject variations achievable by the generator. Wang et al. (2020) create and test a dataset of 4,000 images, with synthetic images generated by a publicly released model and real images drawn from the model's training data. All images are 256x256 pixels.

2.2.3 Conditional GANs

Whereas unconditional GAN-based generators reproduce representative examples from a learned distribution, conditional GANs train on data with class labels. This addition allows a user to specify a class label during image generation to produce images with desired attributes (Mirza and Osindero 2014). In this section, we explore three conditional GAN

architectures used to test the classifiers in the analytic. Figure 2.8 shows representative real and synthetic examples from each dataset.

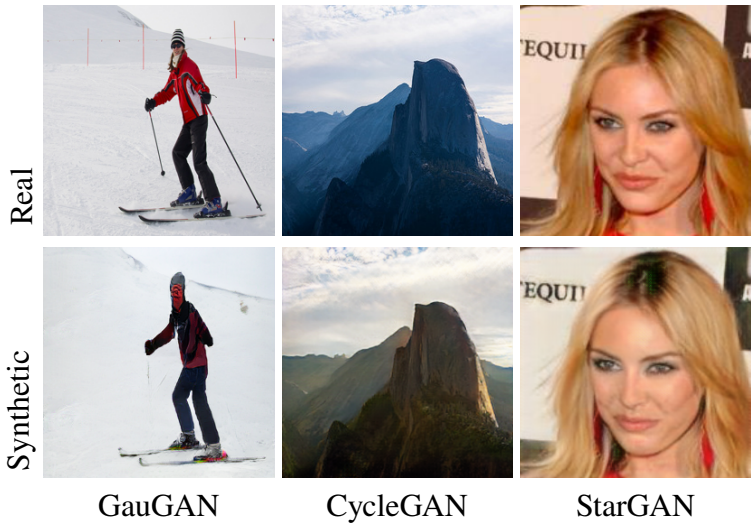


Figure 2.8. Representative images from the conditional GAN datasets. Images from Wang et al. (2020) Git repository and derived from Park et al. (2019); Zhu et al. (2017); Choi et al. (2018).

GauGAN

GauGAN is an architecture proposed by Park et al. (2019) that combines semantic image generation, in which an image is synthesized from a labeled pixel map, with style-based generation like that seen in Karras et al. (2019). Here, a user selects a pixel map and a reference image; the pixel map is used to generate the base image content, while the reference image controls finer details in the appearance of the scene or subject (Park et al. 2019). For example, a pixel map might describe an urban skyline, while a reference image provides the generator with additional context, like the lighting conditions associated with a certain time of day. Wang et al. (2020) use a public-release model to generate synthetic images and pair them with real images from the associated training data. The total dataset contains 10,000 images at 256x256 pixels each.

CycleGAN

CycleGAN is an image-to-image translator proposed by Zhu et al. (2017) that attempts to learn the relationships between images under multiple conditions, without a reliance on

paired images of the same subject. Zhu et al. (2017) also seek to make their generator "cycle-consistent," such that images transformed from one domain to another can be transformed back to the original domain with minimal loss (Zhu et al. 2017). For example, a landscape image depicting a sunset might be transformed to one that depicts the lighting conditions of midday, before being transformed back to an image of a sunset. Wang et al. (2020) produce synthetic images from six CycleGAN models, and pair the synthetic images with real images from the CycleGAN training data. The final test dataset features 2,600 images at 256x256 pixel resolution.

StarGAN

StarGAN is another image-to-image translator proposed by Choi et al. (2018). The StarGAN architecture attempts to conduct translation between multiple image domains with a single model, unlike competitors like Zhu et al. (2017) who rely on multiple models. By training on images that feature all possible or desired combinations of features, StarGAN can generate images with user-specified attributes provided there are sufficient representative examples in the training data (Choi et al. 2018). For example, given an input portrait of a human subject, a user could specify changes to personal appearance attributes like hair color, age, expression, or complexion. Wang et al. (2020) use real and synthetic images sourced from publicly released StarGAN models to produce a test dataset of 4,000 images at 256x256 pixel resolution.

2.2.4 Non-adversarial Models

In this section, we explore several different models and their associated datasets that do not rely on adversarial architectures for training. We specifically describe two generators based on semantic layouts, shown in Figure 2.9, and two low-light vision and super-resolution models, shown in Figure 2.10.

IMLE

As described in Section 2.2.3, semantic layout-based image generators seek to synthesize an image from a labeled pixel map specified as input. The generator proposed by Li et al. (2019) performs this task without adversarial training, instead relying on a conditional implicit maximum likelihood estimation (IMLE) optimization approach that allows the

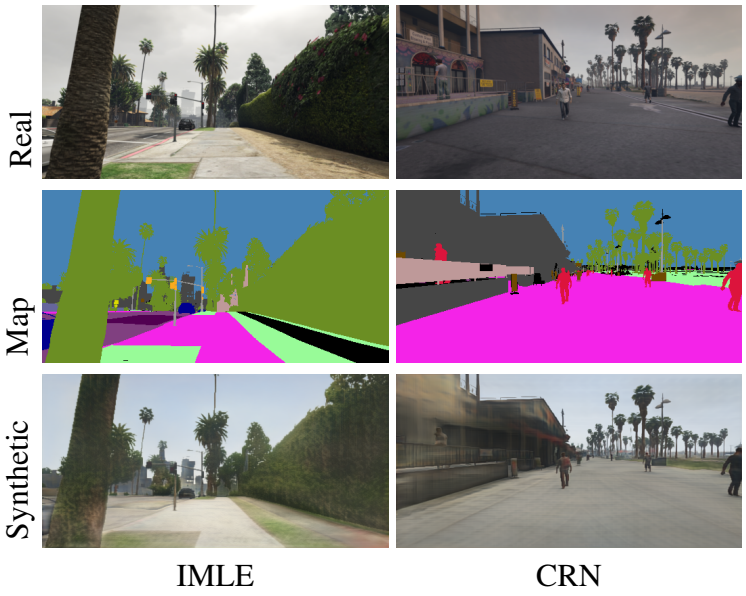


Figure 2.9. Representative images from the non-adversarial semantic layout datasets. A semantic layout map is included in the middle row for context. Wang et al. (2020) use only the real and synthetic images to test their classifiers. Real and synthetic images from Wang et al. (2020) Git repository and derived from Li et al. (2019); Chen and Koltun (2017). Semantic maps sourced directly from Li et al. (2019).

synthesis of multiple images from a single pixel map through variations in noise sampling. The authors use real images and semantic maps from the GTA-5 dataset, derived from the video game of the same name (Li et al. 2019). Wang et al. (2020) use publicly released IMLE models to generate synthetic images and draw real images from the released training data. The final testing dataset contains 12,800 images at a resolution of 512x256 pixels.

CRN

Similar to the IMLE generator developed by Li et al. (2019), Chen and Koltun (2017) employ a non-adversarial approach to image synthesis from semantic maps. In this case, the researchers construct a neural network with layers grouped into modules of increasing resolution. Chen and Koltun (2017) define this architecture as a cascaded refinement network (CRN). Wang et al. (2020) source the real images in the CRN dataset from IMLE repository, and generate synthetic images using the real images' associated pixel maps. The final CRN testing dataset matches the IMLE dataset in size and resolution.

Figure 2.10. Representative images from the non-adversarial low-light/high-resolution datasets. Images from Wang et al. (2020) Git repository and derived from Chen et al. (2018); Dai et al. (2019).

SITD

Chen et al. (2018) propose an algorithm for processing photographs taken in very dark lighting conditions without the resultant increase in image noise associated with the higher camera gains required in these conditions. In this application, Chen et al. (2018) construct a convolutional neural network trained to generate images from raw camera sensor inputs that are indistinguishable from long exposure photographs of the same subject. Wang et al. (2020) construct their See-in-the-Dark (SITD) dataset using publicly released synthetic images produced by the SITD algorithm and pair these images with their corresponding long exposure photographs. The resulting dataset contains 360 total images and a mean resolution of 5113x3420 pixels.

SAN

Similar to the approximation of low-light images in well conditions in the SITD model, single image super-resolution (SISR) generators seek to synthesize a high-resolution image from a low-resolution input (Dai et al. 2019). Designed to overcome the limitations of other CNN-based SISR generators, the second-order attention network (SAN) proposed by Dai et al. (2019) focuses on learning more complex relationships between image features using

higher-order feature statistics. Wang et al. (2020) create their SAN dataset from synthesized high-resolution training images and real input images from the public SAN code release. The final dataset consists of 440 images with a mean resolution of 691x563 pixels.

2.2.5 Traditional Deepfakes

In the analytic of interest, Wang et al. (2020) also test their classifier on a dataset of deepfake images derived from the FaceForensics++ dataset from Rossler et al. (2019). We describe this dataset below, alongside four additional datasets we have added as test sets in this study. Representative images from each dataset are shown in Figure 2.11.

Figure 2.11. Representative images from the deepfake datasets. FaceForensics++ (Author) images from Wang et al. (2020) Git repository and derived from Rossler et al. (2019). FaceForensics++ (Resampled), DFDD, DFDC, and CELEB-DF images sampled and cropped from Rossler et al. (2019); Dufour and Gully (2019); Dolhansky et al. (2020); Li et al. (2020)

.

FaceForensics++ (Author)

The FaceForensics++ dataset created by Rossler et al. (2019) is designed to serve as a benchmark for deepfake detection algorithms. The dataset features both real videos sourced from YouTube and deepfakes created using various source and target subject combinations in the real videos. All videos are formatted as .mp4. Rossler et al. (2019) use four different publicly available generators to create the deepfakes in their dataset: *Face2Face*, *FaceSwap*, *DeepFakes*, and *NeuralTextures*. The authors also apply varying levels of compression to

their deepfake outputs to simulate the wide variability in quality observed in deepfakes circulating on the internet. Wang et al. (2020) extract cropped subject faces from the real and deepfake videos in the FaceForensics++ dataset using the *Faced* face detection tool from Itzcovich (2018). In cases where the resulting face crop is smaller than 256 pixels on the short side, the image is resized with bilinear interpolation such that this condition is met (Wang et al. 2020). These steps result in a new dataset of 5,400 videos at a mean resolution of 261x291. In the analytic of interest, this dataset is referred to as *Deepfake*. To avoid confusion in this study, we will refer to this dataset as *FaceForensics++ (Author)*.

FaceForensics++ (Resampled)

In this study, we also create a new dataset based on FaceForensics++ for additional testing. We sample five frames from each real and fake video in the FaceForensics++ catalog, and crop and extract faces using the Dlib package from King (2009). The resulting dataset contains approximately 8,000 images with a mean resolution of 183x183 pixels[2]; we refer to this dataset as *FaceForensics++ (Resampled)*.

Deepfake Detection Dataset

In 2019, Rossler et al. (2019) augmented their FaceForensics++ benchmark with an additional dataset. Developed by Google AI in collaboration with Jigsaw, the Deepfake Detection Dataset (DFDD) contains approximately 13,000 real images and deepfakes featuring 28 paid actors, with the deepfakes generated using a variety of undisclosed methods and processing techniques (Dufour and Gully 2019). In this study, we use this dataset as an additional test set for the analytic's classifiers. We capture faces from each of the released DFDD images and produce an equivalently sized dataset with a mean resolution of 183x183 pixels.

Deepfake Detection Challenge

In 2020, Dolhansky et al. (2020) partnered with the data science website Kaggle to host the Deepfake Detection Challenge (DFDC), a public competition in support of deepfake detection technology development. The accompanying training and testing dataset features approximately 100,000 real and synthesized video clips with over 3,000 paid actors

[2]**A note on deepfake image size**: Instead of globally resizing the images in these additional deepfake datasets to 256 pixels on the short side, as seen in the FaceForensics++ (Author) dataset from Wang et al. (2020), we add the ability to resize deepfake images at classifier run time as part of our experimental design.

(Dolhansky et al. 2020). A variety of factors make this dataset especially challenging for classifiers designed to detect synthetic media and deepfakes. Dolhansky et al. (2020) use five different synthetic media generators to create the videos in the dataset, but do not disclose the number of videos created by each generator. The generators include a traditional autoencoder-based deepfake generator, a method that creates a face mask with fixed eye and mouth expressions, two additional GAN-based generators, and a method that replaces a source face with a synthesized StyleGAN face in every frame. Videos with face swaps also received additional blending and alignment operations in post-processing, with a sharpening effect applied to certain faces at random (Dolhansky et al. 2020). Seventy percent of videos in the dataset received additional augmentations including changes to video scale, rotation color, framerate, contrast, and compression. Thirty percent of the videos received text or graphics overlays (Dolhansky et al. 2020). To create the dataset used in this study, we randomly select 800 images from the real and deepfake videos in the public DFDC sample and capture cropped faces from each image using Dlib. Our final dataset has a mean resolution of 156x156 pixels.

CELEB-DF

The CELEB-DF dataset, created by Li et al. (2020), serves as another deepfake detection benchmark, with a special emphasis on the visual quality of the included deepfakes. Li et al. (2020) produce deepfakes using a standard autoencoder-based GAN architecture, but include additional post-processing steps to create deepfakes representative of the high-quality examples commonly circulated on the internet and social media. The CELEB-DF network produces synthetic faces at 256x256 pixel resolution, and the architecture includes provisions to reduce temporal noise along facial landmarks (Li et al. 2020). Synthesized face masks include a larger region of the face, allowing for more refined blending operations when the face is composited onto the source video. Output deepfakes also undergo color-matching between the source and target faces (Li et al. 2020). To create our dataset for testing, we extract five frames from each video in the original CELEB-DF dataset, and crop for faces using Dlib. We then balance our real and fake classes by sampling from the face captures of fake faces. We produce a final dataset of approximately 4,600 images with a mean resolution of 145x145 pixels.

In this chapter, we have explored several supporting technical concepts that provide context

for the analytic of interest. We have also explored each of the image generators and datasets used to train and test the analytic in Wang et al. (2020) and in our reproducibility study. In the next chapter, we provide a summary of the analytic itself.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Summary of the Analytic

In this chapter, we describe the aspects of classifier design, classifier training, and image pre-processing that are unique to the analytic of interest, and present the authors' results to be reproduced in this study.

3.1 Classifier Design

Wang et al. (2020) construct a binary classifier in PyTorch (Facebook, Inc 2021) using an off-the-shelf ResNet-50 model with pretrained ImageNet weights from He et al. (2016). The authors select an Adam optimizer with parameters $\beta_1 = 0.9$ and $\beta_2 = 0.999$, and train with a constant batch size of 64. The learning rate begins at 10^{-4} and undergoes a reduction by a factor of 10 if the classifier does not achieve an increase on the validation set of $\geq 0.1\%$ after five epochs. Training concludes when the learning rate reaches 10^{-6} (Wang et al. 2020).

3.2 Training the Classifier

In their paper, Wang et al. (2020) test the impact of training set diversity on classifier generalization by subsetting their ProGAN training set into smaller datasets of 2, 4, 8, and 16 LSUN object categories, but achieve the best overall analytic performance by training on their full 720,000-image training set. In this study, we seek to reproduce the generalization achieved by the best performing classifier, so the training process described here focuses on the classifiers trained with the full 20-class ProGAN dataset.

All training images are left-right flipped with probability 0.5 and receive a 224x224-pixel random crop (Wang et al. 2020). To test robustness to certain image post-processing operations that might characterize synthetic images in the wild, Wang et al. (2020) create five different classifiers with various image pre-processing steps performed at training time, as summarized in Table 3.1. In the first model, referred to in this study as **No Aug**, training images receive no augmentation beyond the left-right flipping and cropping described above. In the **Blur Only** model, Gaussian blur is applied to training images using the SciPy library from Virtanen et al. (2020) with $\sigma \sim \text{Unif}(0, 3)$ with probability 0.5, where σ denotes

Table 3.1. Summary of image pre-processing operations performed at training time by model. If an operation is not performed in a given model, the associated probability and operation parameters are specified as N/A.

Name	Probability		Parameters	
	Blur	JPEG	Blur	JPEG
No Aug	N/A	N/A	N/A	N/A
Blur Only	0.5	N/A	$\sigma \sim \text{Unif}(0, 3)$	N/A
JPEG Only	N/A	0.5	N/A	Qual. $\sim \text{Unif}\{30, 100\}$
Blur + JPEG (0.1)	0.1	0.1	$\sigma \sim \text{Unif}(0, 3)$	Qual. $\sim \text{Unif}\{30, 100\}$
Blur + JPEG (0.5)	0.5	0.5	$\sigma \sim \text{Unif}(0, 3)$	Qual. $\sim \text{Unif}\{30, 100\}$

the standard deviation of the Gaussian kernel, and where $\text{Unif}(a, b)$ denotes a continuous uniform distribution with support $[a, b]$ (Virtanen et al. 2020; Wang et al. 2020). In the **JPEG Only** model, training images are converted to JPEG by the Python Imaging Library from Clark (2021) with probability 0.5 and quality defined by $\sim \text{Unif}\{30, 100\}$, where $\text{Unif}\{a, b\}$ denotes a discrete uniform distribution with support $\{a, a + 1, \ldots, b - 1, b\}$ (Wang et al. 2020). Two additional models combine both Gaussian blur and compression. In the **Blur + JPEG (0.1)** model, blur and JPEG compression are applied independently to the training images, each with probability 0.1. In the **Blur + JPEG (0.5)** model, these probabilities increase to 0.5 (Wang et al. 2020).

3.3 Evaluating Test Datasets

At testing time, images passed to a trained classifier for evaluation are center-cropped to 224x224 pixels. No other processing operations are performed at testing time in the original analytic (Wang et al. 2020). In their final code release, Wang et al. (2020) also include an option to evaluate test images without an explicit crop. Wang et al. (2020) configure the average pooling layer of the ResNet model in an adaptive setting, allowing the evaluation of images of arbitrary size. We incorporate this option into our experimental design, and test all classifiers in both settings.

3.3.1 Average Precision

Wang et al. (2020) select AP as their primary means of evaluating their classifiers' performance on a given dataset, and we continue this practice in our study. AP is defined as the area under a precision-recall curve. Precision, denoted as p, describes the ratio of true positive predictions to the sum of all positive predictions made by a model, as shown in Equation 3.1. Recall, denoted as r, describes the ratio of true positive predictions to the sum of all actual positive examples in the evaluated data, as shown in Equation 3.2. Wang et al. (2020) also compute the mean average precision (mAP) of each trained model over all evaluated generators $g \in G$ to compare overall model performance, as shown in Equation 3.4, where AP_g denotes the AP achieved over each generator g and $n(G)$ denotes the total number of tested generators. We continue this use of mAP in our study.

$$p = \frac{\text{True Positives}}{\text{True Positives + False Positives}} \tag{3.1}$$

$$r = \frac{\text{True Positives}}{\text{True Positives + False Negatives}} \tag{3.2}$$

$$AP = \int_0^1 p(r)\, dr \tag{3.3}$$

$$mAP = \frac{1}{n(G)} \sum_{g \in G} AP_g \tag{3.4}$$

The calculation of precision and recall does not require knowledge of the true negative example rate in the evaluated data, making AP robust to datasets with numerically imbalanced classes. Additionally, unlike accuracy and other similar measures of performance, AP does not require that a specific confidence level be selected as the cutoff between positive and negative predictions. In this study, we represent AP on the interval $[0, 1]$, with a score of 1 indicating perfect positive predictive performance and a score of 0.5 representing performance commensurate with guessing the positive labels at random.

3.4 Results

In models trained on the full 20-class training set, the analytic demonstrates strong performance against each of the tested datasets. This performance suggests that training on ProGAN images may contribute to generalization across CNN-based image generators and against non-adversarial models. Figure 3.1 shows the performance of each classifier model by test dataset. Table 3.2 provides a numerical summary of performance, with mAP also reported by model over all tested datasets.

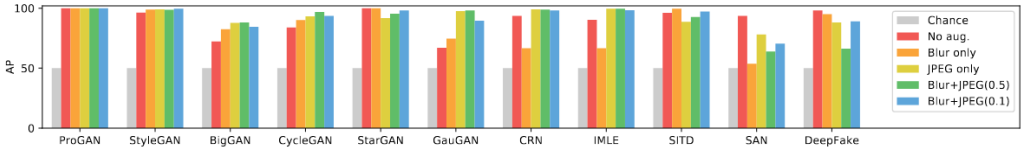


Figure 3.1. AP performance by dataset and model in the original analytic. Source: Wang et al. (2020).

Table 3.2. Summary of the original analytic results for the 20-class models. The **mAP** column on the right provides mean AP over all tested generators by model. The highest AP performance in each column is bolded. Note that the strongest performance on CycleGAN and IMLE were achieved by models trained on fewer LSUN classes, so no bolding appears in these columns. Source: Wang et al. (2020).

Name	Training settings					Individual test generators											Total
	Train	Input	No. Class	Augments		Pro-GAN	Style-GAN	Big-GAN	Cycle-GAN	Star-GAN	Gau-GAN	CRN	IMLE	SITD	SAN	Deep-Fake	**mAP**
				Blur	JPEG												
No aug	ProGAN	RGB	20			1.000	0.963	0.722	0.840	**1.000**	0.670	0.935	0.903	0.962	**0.936**	**0.982**	0.901
Blur only	ProGAN	RGB	20	✓		1.000	0.990	0.825	0.901	**1.000**	0.747	0.666	0.667	**0.996**	0.537	0.951	0.844
JPEG only	ProGAN	RGB	20		✓	1.000	0.990	0.878	0.932	0.918	0.975	**0.990**	0.995	0.887	0.781	0.881	**0.930**
Blur+JPEG (0.5)	ProGAN	RGB	20	✓	✓	1.000	0.985	**0.882**	0.968	0.954	**0.981**	0.989	0.995	0.927	0.639	0.663	0.908
Blur+JPEG (0.1)	ProGAN	RGB	20	†	†	1.000	**0.996**	0.845	0.935	0.982	0.895	0.982	0.984	0.972	0.705	0.890	0.926

Average precision (AP) across 11 generators. Symbols ✓ and † mean the augmentation is applied with 50% or 10% probability, respectively, at training.

Examining Table 3.2, Wang et al. (2020) observe the highest mAP performance from the **JPEG Only** model, although the **Blur + JPEG (0.1)** model also exhibits a high mAP. Several clear trends emerge. First, all models achieve an AP of 1.000 against the withheld ProGAN test set, indicating perfect fake image detection against new images from the generator used

for training. Second, we observe that image augmentations at training time generally result in stronger analytic performance across all generators, with three notable exceptions (Wang et al. 2020). In the case of StarGAN, we note that the **No Aug** and **Blur Only** models tie for best performance, but we also see near-equivalent performance from the **Blur + JPEG (0.5)** model, so we resist drawing conclusions about the impact of augmentations here. However, the **No Aug** model clearly exhibits the highest performance against the SAN and FaceForensics ++ (Author)[3] datasets. In the case of the SAN dataset, Wang et al. (2020) hypothesize that the high-frequency content produced by the generator provides the best indication of whether a given image is synthetic; models trained with augmentations that would destroy this information struggle to make correct predictions (Wang et al. 2020). In the case of the FaceForensics ++ (Author) dataset, Wang et al. (2020) refrain from drawing conclusions as to why the **No Aug** model exhibits the best performance.

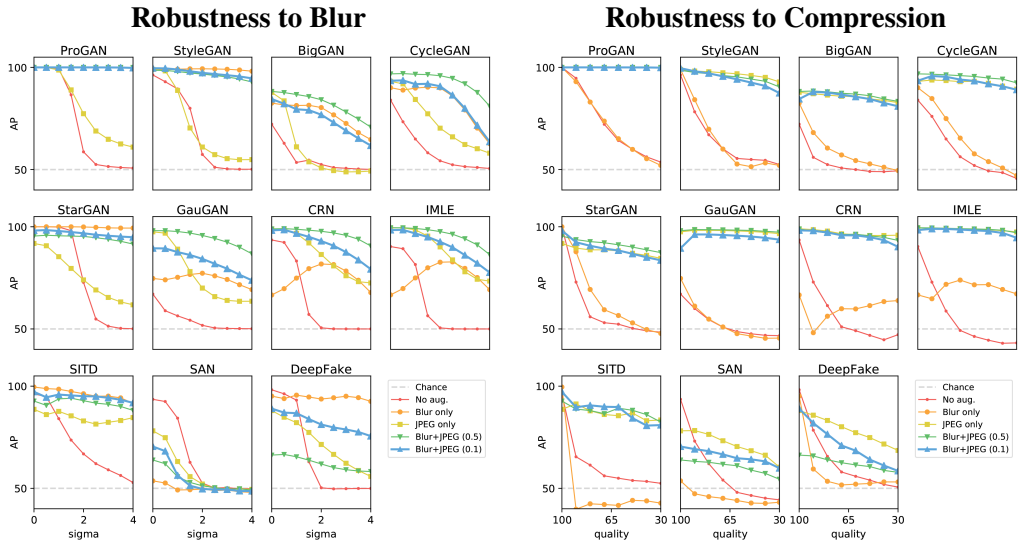


Figure 3.2. AP performance by dataset and model against test images augmented with blur and compression. Source: Wang et al. (2020).

Wang et al. (2020) also test their analytic's robustness to images with augmentations applied. Figure 3.2 shows AP performance by model against each dataset with Gaussian blurring and JPEG compression applied to the test images. The researchers note that the **No Aug** model performance decreases as more intense blurring and compression are applied. The

[3]As indicated in Section 2.2.5, Wang et al. (2020) refer to this dataset as *Deepfake* in the supporting tables and graphics here.

Blur Only and **JPEG Only** models appear fairly robust to their associated augmentation, while the combined models perform well in almost all cases (Wang et al. 2020). The results of this experiment indicate that models trained with augmentations may be suited to tasks requiring the detection of synthetic images suspected to have undergone transformations, like those seen on the internet or social media (Wang et al. 2020).

3.5 Analysis

The results achieved by Wang et al. (2020) are significant for several reasons. First, the ability of the analytic to train on images from a single CNN-based generator and record strong performance against other related generators is striking, and suggests that there may be common artifacts in CNN-based images that enable their detection (Wang et al. 2020). Second, while the creation of fully synthetic images and deepfake face swaps share underlying concepts, the outputs of deepfake creation are fundamentally different from those of other synthetic image generators. The strong performance of the **No Aug** model against the FaceForensics ++ (Author) dataset here is unexpected and worthy of further investigation. Finally, the analytic promises both high performance and simplicity, with only a single training set, an off-the-shelf model, and some variable pre-processing as its requirements. This simplicity in design contributes to the potential value and possible applications of the analytic, if reproducible.

In the next chapter, we describe the methods and results of our reproducibility study, and document additional experiments that further test the analytic of interest.

CHAPTER 4:
Reproducibility and Experiments

In this chapter, we first document the results of our reproducibility study. We then present the results of additional experiments designed to identify the strengths and weaknesses of the analytic of interest.

4.1 Reproducibility Results

In our reproducibility study, we conducted two experiments to verify the published results from Wang et al. (2020). We first attempted to reproduce the results from Wang et al. (2020) using two models included as part of the public code release for the analytic. We then retrained the five classifiers from the analytic described in Section 3.2, and tested them on the authors' original training set.

4.1.1 Reproducing Author Results with Publicly Released Models

As part of their public code release, Wang et al. (2020) include two pre-trained versions of their **Blur + JPEG (0.1)** and **Blur + JPEG (0.5)** models. In our first reproducibility experiment, we download these model weights and test them on each of the generator datasets included in the original study using both the center-crop and uncropped settings; we also test these models again the StyleGAN2 and Which Face is Real (WFIR) datasets. Table 4.1 provides a summary of our results.

Table 4.1. Summary of reproducibility results using publicly released models from Wang et al. (2020). Results from the best performing model against each generator are bolded. In the center-crop setting, results on individual generators exactly match those achieved by Wang et al. (2020) in Table 3.2.

Name	Crop	ProGAN	StyleGAN	StyleGAN2	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	FF++ (Author)	WFIR	**mAP**
Blur+JPEG (0.1)	None	**1.000**	**0.998**	**0.995**	0.860	0.949	**0.990**	0.908	0.998	0.998	**0.998**	0.686	0.845	0.998	**0.940**
Blur+JPEG (0.1)	224px (center)	**1.000**	0.996	0.991	0.845	0.935	0.982	0.895	0.982	0.984	0.972	**0.705**	**0.890**	0.932	0.931
Blur+JPEG (0.5)	None	**1.000**	0.993	0.991	**0.904**	**0.979**	0.975	**0.988**	**1.000**	**1.000**	0.996	0.628	0.631	**1.000**	0.930
Blur+JPEG (0.5)	224px (center)	**1.000**	0.985	0.980	0.882	0.968	0.954	0.981	0.989	0.995	0.927	0.639	0.663	0.888	0.912

We observe that in the center-crop setting, the model results on individual generators exactly match those achieved by Wang et al. (2020), as shown in Table 3.2. Because we include

test results from the StyleGAN2 and WFIR datasets in our summary table, our mAP results for the **Blur + JPEG (0.1)** and **Blur + JPEG (0.5)** models in the center-crop setting are slightly higher than those reported by Wang et al. (2020) in Table 3.2. Moreover, while Wang et al. (2020) did not provide results for their models in the "no crop" setting in their original paper, we observe that in general, refraining from center-cropping images at testing time improves model performance, with the only exceptions occurring on the SAN and FaceForensics++ (Author) datasets.

4.1.2 Reproducing Author Results by Retraining Models

In our next experiment, we use the documentation in the public code release from Wang et al. (2020) to retrain each of the five 20-class models featured in the original paper on the authors' ProGAN training set, with pre-processing settings as prescribed in Table 3.1. We then test these models against each generator in the authors' test set, again including the StyleGAN2 and WFIR datasets. Table 4.2 provides a summary of our full results.

Table 4.2. Summary of reproducibility results using retrained models. Results from the best performing model against each generator are bolded. In the center-crop setting, results on individual generators are similar to those achieved by Wang et al. (2020) in Table 3.2.

Name	Crop	ProGAN	StyleGAN	StyleGAN2	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	FF++ (Author)	WFIR	**mAP**
No Aug	None	**1.000**	0.979	0.998	0.700	0.832	**1.000**	0.712	0.888	0.883	0.979	**0.956**	0.976	0.970	0.913
No Aug	224px (center)	**1.000**	0.976	0.996	0.686	0.798	**1.000**	0.679	0.873	0.858	0.970	0.919	**0.987**	0.891	0.895
Blur Only	None	**1.000**	0.989	**1.000**	0.843	0.926	**1.000**	0.772	0.766	0.766	**0.999**	0.525	0.868	0.981	0.870
Blur Only	224px (center)	**1.000**	0.984	**1.000**	0.822	0.904	**1.000**	0.751	0.712	0.714	0.996	0.571	0.926	0.842	0.863
JPEG Only	None	**1.000**	0.996	0.989	**0.909**	0.940	0.949	0.978	0.999	**1.000**	0.941	0.746	0.760	0.997	**0.939**
JPEG Only	224px (center)	**1.000**	0.991	0.977	0.896	0.916	0.915	0.971	0.987	0.995	0.854	0.747	0.813	0.913	0.921
Blur+JPEG (0.1)	None	**1.000**	**0.998**	0.996	0.859	0.950	0.990	0.922	0.998	0.999	0.996	0.688	0.821	0.995	**0.939**
Blur+JPEG (0.1)	224px (center)	**1.000**	0.996	0.992	0.841	0.934	0.980	0.909	0.982	0.989	0.969	0.702	0.875	0.906	0.929
Blur+JPEG (0.5)	None	**1.000**	0.993	0.992	0.901	**0.972**	0.978	**0.988**	**1.000**	**1.000**	0.996	0.631	0.664	**1.000**	0.932
Blur+JPEG (0.5)	224px (center)	**1.000**	0.983	0.980	0.880	0.959	0.959	0.981	0.992	0.995	0.907	0.639	0.706	0.920	0.915

We again observe that in the center-crop setting, individual generator results closely match those achieved by Wang et al. (2020) in Table 3.2, with the strong performance from at least one model on each generator. We observe the highest mAP performance from the **JPEG Only** and **Blur + JPEG (0.1)** models. As before, across all models, performance in the uncropped setting is generally higher than performance in the center-crop setting, with the exception in this case being results on the FaceForensics++ (Author) dataset. Nonetheless, the very high AP achieved by the **No Aug** model on the FaceForensics++ (Author) dataset in both crop settings is striking and warrants further investigation.

Our findings in both reproducibility experiments suggest that the analytic is suitable for detecting images from a wide variety of CNN-based generators, especially if information is available regarding which generator may have created a suspect image. However, since the authors' test set includes only one dataset of deepfake images, we refrain from drawing conclusions about the analytic's suitability for detecting deepfakes without further experiments.

4.2 Experiments

Guided by our reproducibility results, we conduct five additional experiments to further explore the analytic's strengths and weaknesses regarding generalization, performance on deepfakes, and robustness to exploitation by adversarial perturbations.

4.2.1 Performance on a Test Set of Mixed Generators

Results from our reproducibility experiments indicate that the analytic performs well against a wide variety of generators when tested individually. However, since each generator dataset may have its own ideal prediction confidence threshold that separates real and synthetic images, measuring overall model performance with mAP across all generators may result in an overly optimistic assessment of the analytic's ability to generalize. In this experiment, we construct a new test set featuring a sample of real and synthetic images from each generator dataset such that the ideal confidence threshold dividing our real and synthetic images becomes uncertain. This experiment also simulates real world applications of the analytic, where potential users may not know the provenance of suspected synthetic images.

To construct our new test set, we sample 360 real and fake images from each of the 13 generators in the authors' testing dataset. We then test each of our retrained models on the 9,360 images in this new test set in both available crop settings. Table 4.3 summarizes the results of this experiment.

We observe that for most models, performance against this new multi-generator dataset is similar to the mAP achieved across all generators in Section 4.1.2. Greater image preprocessing at training time appears beneficial in this application, with the **Blur + JPEG (0.5)** model in the uncropped setting exhibiting the best performance. The **JPEG Only** and **Blur + JPEG (0.1)** models also perform well. Meanwhile, the **Blur Only** and **No Aug**

Table 4.3. Summary of the analytic's results on a resampled test set consisting of images from multiple generators. Results from the best performing model are bolded. Model results are similar to the mAP results achieved across all generators in Section 4.1.2.

Name	Crop	AP
No Aug	None	0.838
No Aug	224px (center)	0.856
Blur Only	None	0.802
Blur Only	224px (center)	0.794
JPEG Only	None	0.945
JPEG Only	224px (center)	0.926
Blur+JPEG (0.1)	None	0.932
Blur+JPEG (0.1)	224px (center)	0.919
Blur+JPEG (0.5)	None	**0.950**
Blur+JPEG (0.5)	224px (center)	0.930

models demonstrate slightly worse performance in comparison to their mAP results across all generators in Section 4.1.2. As before, we see that models operating in the uncropped setting achieve slightly higher performance than when operating in the center-crop setting, with the exception here being the **No Aug** model.

These results in a simulated real-world setting suggest that the analytic may perform well against images from an unknown subset of CNN-based image generators. Our results also match the findings of Wang et al. (2020) and our own reproducibility experiments, where a mixture of blur and compression pre-processing at training time appears beneficial for classifier generalization.

4.2.2 Performance on Additional Deepfake Datasets

To better understand the analytic's performance against deepfake face swaps, we test the analytic on the additional deepfake datasets introduced in Section 2.2.5. Additionally, while Wang et al. (2020) resized the face crops in their FaceForencics++ (Author) dataset to 256x256 pixels with bilinear interpolation prior to testing the images, we add an option to resize test images at classifier runtime, and include this option in our experimental design. We therefore test each model using all four combinations of our crop and resize settings.

Images not resized at testing time are left at their original resolution prior to any cropping operations. In cases where the classifier must center-crop an un-resized image smaller than 224x224 pixels in either dimension, the center-cropping operation pads the test image with zero prior to cropping. Table 4.4 provides our full results.

Examining Table 4.4, we observe mixed results on the additional deepfake datasets. On our resampled version of FaceForensics++, our best performing model demonstrates similar performance to that achieved by Wang et al. (2020) on their FaceForensics++ (Author) dataset in Table 3.2. We also achieve relatively high performance on DFDD. We note that the **No Aug** model in the resize setting achieves the best results against both of these datasets. Center-cropping improves performance on DFDD but reduces performance against FaceForensics++ (Resampled).

Table 4.4. Full results of retrained model testing on additional deepfake datasets in cropped/uncropped and resized/not resized settings. Results from the best performing model against each dataset are bolded.

Model	Crop	Resize	CELEB-DF	DFDC (Facebook)	DFDD (Google)	FF++ (Resampled)	**mAP**
No Aug	None	No	0.527	0.519	0.918	0.786	0.687
No Aug	None	Yes	0.525	0.523	0.923	**0.860**	0.708
No Aug	224px (center)	No	0.528	0.520	0.920	0.783	0.688
No Aug	224px (center)	Yes	0.526	0.525	**0.926**	0.858	**0.709**
Blur Only	None	No	0.533	0.517	0.922	0.786	0.690
Blur Only	None	Yes	0.523	**0.533**	0.917	0.851	0.706
Blur Only	224px (center)	No	0.533	0.511	0.925	0.787	0.689
Blur Only	224px (center)	Yes	0.523	0.528	0.921	0.853	0.706
JPEG Only	None	No	0.534	0.518	0.921	0.790	0.691
JPEG Only	None	Yes	0.517	0.527	0.924	0.846	0.704
JPEG Only	224px (center)	No	0.533	0.517	0.921	0.795	0.692
JPEG Only	224px (center)	Yes	0.516	0.526	0.924	0.851	0.704
Blur+JPEG (0.1)	None	No	0.533	0.516	0.921	0.791	0.690
Blur+JPEG (0.1)	None	Yes	0.522	0.527	0.920	0.846	0.704
Blur+JPEG (0.1)	224px (center)	No	0.531	0.517	0.923	0.794	0.691
Blur+JPEG (0.1)	224px (center)	Yes	0.521	0.528	0.922	0.848	0.705
Blur+JPEG (0.5)	None	No	0.536	0.516	0.922	0.789	0.691
Blur+JPEG (0.5)	None	Yes	0.520	0.525	0.920	0.846	0.703
Blur+JPEG (0.5)	224px (center)	No	**0.536**	0.519	0.923	0.793	0.693
Blur+JPEG (0.5)	224px (center)	Yes	0.520	0.529	0.921	0.850	0.705

In the case of CELEB-DF and DFDC, we observe poor performance across all models, with results commensurate with guessing the positive labels at random. We note that the DFDC dataset is intended to be a challenging deepfake detection benchmark, and the analytic's performance here resembles that achieved by most of the models submitted as part of the

associated competition on the competition's private test set. However, the best submission achieved an AP of 0.902 across all real and synthetic videos in the challenge (Dolhansky et al. 2020). CELEB-DF is a similarly challenging benchmark; Li et al. (2020) tested 13 deepfake detection algorithms against their dataset and achieved a maximum AUC of 65.5. Despite the analytic's good performance on FaceForensics++ (Resampled) and DFDD, our results on DFDC and CELEB-DF suggest that the analytic is not suitable for real world deepfake detection tasks.

4.2.3 Robustness to Compression on Deepfake Datasets

To further examine the analytic's performance against deepfakes, we test the analytic's robustness to compression in deepfake test images. In this experiment, we compress the real and synthetic images in each deepfake dataset using the Python Imaging Library from Clark (2021) at quality settings 25, 50, 75, and 95[4]. We also include a fifth *Random* quality category, where we compress images with quality $\sim$ Unif$\{30, 100\}$, recalling the application of image compression at training time for some of the analytic's classifier models. Table 4.5 provides our results from the **JPEG Only** model.

Table 4.5. **JPEG Only** model performance on deepfake datasets. Image compression on deepfake datasets generally reduces analytic performance.

Model	Crop	Resize	Quality	CELEB-DF	DFDC	DFDD	FF++ (Resampled)	FF++ (Author)	**mAP**
JPEG Only	None	No	25	0.527	0.538	0.907	0.583	0.673	0.646
JPEG Only	None	No	50	0.529	0.540	0.906	0.589	0.689	0.651
JPEG Only	None	No	75	0.529	0.539	0.908	0.597	0.702	0.655
JPEG Only	None	No	95	0.530	0.541	**0.910**	0.613	0.720	0.663
JPEG Only	None	No	Rand	0.529	0.539	0.907	0.596	0.700	0.654
JPEG Only	None	Yes	25	0.549	0.558	0.886	0.659	0.673	0.665
JPEG Only	None	Yes	50	**0.551**	0.558	0.885	0.664	0.689	0.669
JPEG Only	None	Yes	75	0.550	0.557	0.886	0.672	0.702	0.673
JPEG Only	None	Yes	95	0.550	0.559	0.888	**0.691**	0.720	**0.682**
JPEG Only	None	Yes	Rand	0.550	0.557	0.886	0.671	0.700	0.673
JPEG Only	224 px (center)	No	25	0.525	0.545	0.899	0.608	0.705	0.656
JPEG Only	224 px (center)	No	50	0.527	0.551	0.899	0.619	0.716	0.662
JPEG Only	224 px (center)	No	75	0.528	0.555	0.901	0.63	0.726	0.668
JPEG Only	224 px (center)	No	95	0.530	**0.561**	0.904	0.652	**0.737**	0.677
JPEG Only	224 px (center)	No	Rand	0.527	0.552	0.901	0.628	0.724	0.666
JPEG Only	224 px (center)	Yes	25	0.550	0.550	0.888	0.644	0.670	0.660
JPEG Only	224 px (center)	Yes	50	**0.551**	0.555	0.888	0.653	0.684	0.666
JPEG Only	224 px (center)	Yes	75	0.550	0.558	0.888	0.663	0.695	0.671
JPEG Only	224 px (center)	Yes	95	**0.551**	**0.561**	0.891	0.682	0.709	0.679
JPEG Only	224 px (center)	Yes	Rand	0.550	0.556	0.889	0.661	0.692	0.670

[4]Clark (2021) recommends using a maximum quality setting of 95; quality settings above 95 result in larger image file sizes without an accompanying increase in image quality.

We observe that on datasets where the analytic had previously exhibited strong performance, image compression generally reduces the analytic's AP results. The **JPEG Only** model achieves the highest performance across all datasets, with the smallest reduction in AP generally observed in images with quality 95. Performance against DFDD remains high, with results similar to those seen from the **JPEG Only** model in 4.2.2. Results on both FaceForensics++ datasets demonstrate greater performance losses, while results against CELEB-DF and DFDC exhibit no meaningful changes from their uncompressed baselines.

Given that deepfake images and videos encountered in real world scenarios may suffer from compression and poor image quality, the loss of performance observed here provides further evidence that the analytic is not suitable for deepfake detection applications.

4.2.4 Training on StyleGAN2 Images

In Section 4.2.2, we noted that the DFDC and CELEB-DF deepfake datasets represent relatively new, challenging benchmarks for deepfake detection. However, the ProGAN generator from Karras et al. (2017) used to create the synthetic images in the training and validation sets of the analytic is now several years old. To better understand the analytic's performance on newer deepfake detection benchmarks, we construct new training and validation datasets using synthetic images generated with StyleGAN2 from Karras et al. (2020b). As introduced in Section 2.2.1, StyleGAN2 is a recently updated conditional GAN architecture capable of producing very convincing synthetic images. We retrain each of our five classifier models using these new training and validation sets, with all other training parameters as described in Chapter 3. We then test our retrained models against the authors' test set and the additional deepfake datasets.

Creating a StyleGAN2 Training Dataset

To create the training and validation images used in our new dataset, we source real images from the Animal Faces HQ dataset from Choi et al. (2020), the Flickr-Faces-HQ dataset from Karras et al. (2019), and the Metfaces dataset from Karras et al. (2020a). We then obtain five StyleGAN2 models from Karras et al. (2020b), each pre-trained on one of our real image sources, and generate a number of synthetic images equal to the number of real images available by dataset. We then divide the images into a training set of approximately 86,000 real and synthetic images and a validation set of 1,300 real and synthetic images. The

Figure 4.1. Example images from each class of our new StyleGAN2 training dataset. Real images sourced from the Animal Faces HQ dataset from Choi et al. (2020), the Flickr-Faces-HQ dataset from Karras et al. (2019), and the Metfaces dataset from Karras et al. (2020a). Synthetic images generated using pre-trained StyleGAN2 models from Karras et al. (2020b).

Table 4.6. StyleGAN2 training and validation sets by model/image category and the number of real and synthetic images in each set.

Category	Training		Validation	
	Real	Synthetic	Real	Synthetic
AFHQ-Cat	5,573	5,573	80	80
AFHQ-Dog	5,159	5,159	80	80
AFHQ-Wild	5,158	5,158	80	80
Flickr-Faces-HQ	68,960	68,960	1,040	1,040
Metfaces	1,316	1,316	20	20
Total	86,166	86,166	1,300	1,300

resulting proportion of training images to validation images approximates the proportion used for the ProGAN training and validation sets seen in Wang et al. (2020). Table 4.6 shows the number of real and synthetic images in each dataset by StyleGAN2 model category. Figure 4.1 shows example real and synthetic images from our new training and validation datasets.

StyleGAN2 Model Results

Testing our new StyleGAN2-trained models on the authors' dataset, we observe similar results to those achieved by Wang et al. (2020) when using classifiers trained on ProGAN

images. We note strong performance against each of the GAN-based image generators, with at least one classifier achieving an AP above 0.900 against each generator. However, we also observe greater variability in model performance on certain generators, with larger differences between the AP achieved by the best and worst performing models than those seen in Wang et al. (2020). We also see reduced model mAP across all generators compared to Wang et al. (2020) and our own reproducibility results in Section 4.1.2. As before, we generally observe higher model performance in the uncropped setting, excepting performance against the FaceForensics++ (Author) dataset. We again observe that the **No Aug** model offers the best performance against this deepfake dataset, but still underperforms slightly in comparison to the **No Aug** model trained on ProGAN images from Wang et al. (2020).

Testing the StyleGAN2 models against our additional deepfake datasets, we observe strong performance against DFDD across all models. However, we see no significant improvement in performance against DFDC and CELEB-DF, and we also observe a catastrophic loss in performance against FaceForensics++ (Resampled), where the ProGAN-based models had previously exhibited relatively strong performance.

This experiment reinforces the hypothesis in Wang et al. (2020) that training a classifier on images from a single CNN-based image generator does result in generalization sufficient to detect images from other CNN-based generators, although it appears that the ProGAN-based training set from Wang et al. (2020) offers superior generalization in comparison to our StyleGAN2-based training set created for this experiment. We note that Karras et al. (2020b) offer several additional pre-trained StyleGAN2 models not used here; potential future work might encompass the creation of a larger or more diverse StyleGAN2 training set. This experiment also demonstrates that training the analytic on images from StyleGAN2 does not improve performance on challenging deepfake benchmarks, and actually results in a loss of performance on a dataset where ProGAN-based models had performed strongly. This finding provides further evidence to suggest that the analytic is not suitable for deepfake detection tasks.

Table 4.7. StyleGAN2 model results on the authors' dataset. Results from the best performing model against each generator are bolded.

Model	Crop	StyleGAN2	StyleGAN	ProGAN	BigGAN	CycleGAN	StarGAN	GauGAN	CRN	IMLE	SITD	SAN	FF++ (Author)	WFIR	mAP
No Aug	None	0.971	0.903	0.997	**0.959**	**0.933**	**1.000**	0.933	0.801	0.897	0.932	**0.802**	0.808	0.942	**0.914**
No Aug	224px (center)	0.966	0.910	0.995	0.946	0.919	**1.000**	0.907	0.675	0.733	0.926	0.703	**0.880**	0.921	0.883
Blur Only	None	**0.987**	0.955	**0.998**	0.878	0.928	0.999	0.898	**0.992**	**1.000**	0.938	0.533	0.817	0.964	**0.914**
Blur Only	224px (center)	0.978	0.946	0.996	0.860	0.915	0.996	0.867	0.960	0.995	0.921	0.556	0.829	0.928	0.904
JPEG Only	None	0.845	0.956	0.963	0.783	0.851	0.844	0.932	0.923	0.987	0.907	0.614	0.470	0.985	0.851
JPEG Only	224px (center)	0.794	0.919	0.950	0.755	0.843	0.818	0.907	0.828	0.938	0.915	0.628	0.442	0.871	0.816
Blur + JPEG (0.1)	None	0.916	**0.962**	0.986	0.882	0.880	0.922	0.931	0.942	0.998	**0.999**	0.596	0.574	0.991	0.891
Blur + JPEG (0.1)	224px (center)	0.887	0.943	0.978	0.860	0.870	0.873	0.904	0.814	0.974	0.971	0.587	0.570	0.963	0.861
Blur + JPEG (0.5)	None	0.808	0.938	0.963	0.777	0.845	0.823	**0.937**	0.919	0.995	0.973	0.527	0.573	**0.997**	0.852
Blur + JPEG (0.5)	224px (center)	0.750	0.881	0.950	0.758	0.827	0.766	0.912	0.759	0.956	0.878	0.538	0.564	0.900	0.803

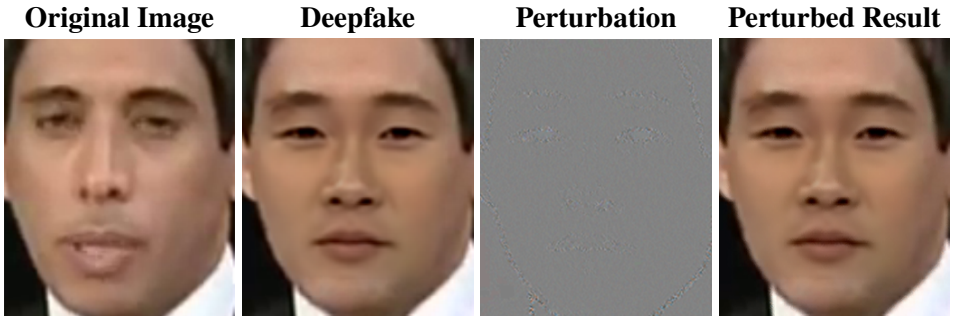
Table 4.8. StyleGAN2 model results on the additional deepfake datasets. Results from the best performing model against each generator are bolded.

Model	Crop	Resize	CELEB-DF	DFDC	DFDD	FF++ (Resampled)	**mAP**
No Aug	None	No	0.537	0.518	0.899	0.575	0.632
No Aug	None	Yes	0.512	0.512	0.924	0.572	0.630
No Aug	224px (center)	No	**0.538**	0.516	0.897	0.577	0.632
No Aug	224px (center)	Yes	0.513	0.511	0.923	0.574	0.630
Blur Only	None	No	0.536	0.518	0.901	0.583	**0.634**
Blur Only	None	Yes	0.513	0.513	0.919	0.568	0.628
Blur Only	224px (center)	No	0.536	0.515	0.901	0.583	**0.634**
Blur Only	224px (center)	Yes	0.513	0.511	0.920	0.568	0.628
JPEG Only	None	No	0.530	0.516	0.901	0.564	0.628
JPEG Only	None	Yes	0.516	0.508	0.920	**0.587**	0.633
JPEG Only	224px (center)	No	0.527	0.518	0.900	0.563	0.627
JPEG Only	224px (center)	Yes	0.513	0.509	0.920	0.586	0.632
Blur + JPEG (0.1)	None	No	0.530	0.513	0.896	0.578	0.629
Blur + JPEG (0.1)	None	Yes	0.517	0.514	0.925	0.572	0.632
Blur + JPEG (0.1)	224px (center)	No	0.529	0.512	0.895	0.579	0.629
Blur + JPEG (0.1)	224px (center)	Yes	0.516	0.513	**0.926**	0.572	0.632
Blur + JPEG (0.5)	None	No	0.533	0.516	0.897	0.581	0.632
Blur + JPEG (0.5)	None	Yes	0.515	0.509	0.925	0.570	0.630
Blur + JPEG (0.5)	224px (center)	No	0.532	**0.519**	0.896	0.581	0.632
Blur + JPEG (0.5)	224px (center)	Yes	0.513	0.511	0.925	0.570	0.630

4.2.5 Robustness to Adversarial Perturbations

In our final experiment, we test the analytic's robustness to adversarial image perturbations. We draw upon previous research conducted at Naval Postgraduate School (NPS) by Bassett et al. (2020) to apply color and edge-aware adversarial perturbations to the real and synthetic images in the FaceForensics++ (Author) dataset, and test our retrained ProGAN-based **No Aug** model against this perturbed dataset.

Adversarial perturbations exploit a weakness of neural networks, in which very small changes to an input passed to a network can result in unexpected outputs. In an image classification context, small changes to the pixel values of an input image can cause a neural network-based classifier to report the image as belonging to an entirely different class (Geng and Veerapaneni 2018).



Far left: A real image from the FaceForensics++ (Author) dataset. The **No Aug** model correctly predicts that the image is real. Center left: A deepfake face swap performed on the original image. The classifier correctly predicts that the image is synthetic. Center right: A color and edge-aware perturbation generated using the deepfake image and targeting the real class label. Far right: The deepfake image with the perturbation applied. The **No Aug** model now incorrectly predicts that the image is real. Real and deepfake images sourced from Git repository of Wang et al. (2020) and derived from Rossler et al. (2019). Perturbation and perturbed result generated using code from Bassett et al. (2020).

Figure 4.2. An example of an adversarial perturbation applied to a deepfake image from the FaceForensics++ (Author) dataset.

Color and Edge-Aware Adversarial Image Perturbations

Bassett et al. (2020) propose a new procedure for the application of targeted adversarial perturbations, where noise is injected into an image to cause a neural network to report a specific desired misclassification. Bassett et al. (2020) specifically contribute two methods for the construction of adversarial perturbations which are imperceptible or near-imperceptible to the human eye.

The edge-aware method penalizes changes to an image in smooth regions where the human eye is more likely to detect the added noise (Bassett et al. 2020). The color-aware method converts images to the CIE L*a*b* (CIELAB) color space prior to the creation of perturbations, and constructs perturbations which are small with respect to the ℓ_2 distance in CIELAB space between the image's original pixel values and the perturbed pixel values. Since differences in color which are small with respect to ℓ_2 in the CIELAB color space are less perceptible to the human eye than corresponding changes in other color spaces, the color-aware method creates less noticeable adversarial perturbations (Bassett et al. 2020). In

the context of deepfakes, perturbed images created using color and edge-aware techniques preserve the appearance of the swapped face, with perturbations applied to high-contrast regions of the face (Bassett et al. 2020).

Perturbation Application and Testing

To construct our perturbed dataset, we use our retrained ProGAN-based **No Aug** model to apply color and edge-aware perturbations to the real and synthetic images in the FaceForensics++ (Author) dataset while targeting the opposite class label. We then test all five retrained ProGAN-based models on the perturbed dataset.

Table 4.9. Model performance on the FaceForensics++ (Author) dataset with the real and synthetic images perturbed with color and edge-aware perturbations. We observe catastrophic degradation in performance across all models, indicating that the analytic is not robust to small changes in the input image.

Model	Crop	Resize	AP
No Aug	None	No	**0.392**
No Aug	None	Yes	0.368
No Aug	224px (center)	No	**0.392**
No Aug	224px (center)	Yes	0.368
Blur Only	None	No	0.381
Blur Only	None	Yes	0.368
Blur Only	224px (center)	No	**0.392**
Blur Only	224px (center)	Yes	0.379
JPEG Only	None	No	0.387
JPEG Only	None	Yes	0.368
JPEG Only	224px (center)	No	**0.392**
JPEG Only	224px (center)	Yes	0.373
Blur + JPEG (0.1)	None	No	0.385
Blur + JPEG (0.1)	None	Yes	0.368
Blur + JPEG (0.1)	224px (center)	No	**0.392**
Blur + JPEG (0.1)	224px (center)	Yes	0.375
Blur + JPEG (0.5)	None	No	0.391
Blur + JPEG (0.5)	None	Yes	0.368
Blur + JPEG (0.5)	224px (center)	No	**0.392**
Blur + JPEG (0.5)	224px (center)	Yes	0.369

We observe a catastrophic degradation in the analytic's performance across all models, as shown in Table 4.9. Our models universally produce predictions that are worse than selecting the predicted labels uniformly at random, indicating the active misclassification of negative examples as positive. This experiment indicates that the analytic is not robust to small changes in the input image.

4.3 Conclusion

In these experiments, we have shown that the initial results achieved by Wang et al. (2020) are both valid and reproducible. Our subsequent experiments related to generalization provide additional evidence in support of the authors' hypothesis that training a classifier on synthetic images from a single CNN-based generator allows the classifier to generalize and detect synthetic images from a wide variety of image generators.

However, we found that the results achieved by Wang et al. (2020) against the FaceForensics++ (Author) dataset do not extend to datasets featuring high quality deepfakes. We were also unable to improve performance against these datasets by training on similarly high quality StyleGAN2 images. These findings indicate that the analytic may not be suitable for classification tasks involving deepfake detection.

Finally, we observed that the analytic is not robust to small changes in the input image. Prospective users of the analytic should be aware of this lack of robustness and employ the analytic with caution.

Together, these findings indicate that the analytic proposed by Wang et al. (2020) is highly capable in classification tasks against many types of synthetic images, but may not be suitable for image analysis and classification tasks in the Department of Defense (DOD).

THIS PAGE INTENTIONALLY LEFT BLANK

List of References

Bassett R, Graves M, Reilly P (2020) Color and edge-aware adversarial image perturbations. *arXiv preprint arXiv:2008.12454*.

Brock A, Donahue J, Simonyan K (2018) Large scale GAN training for high fidelity natural image synthesis. *arXiv preprint arXiv:1809.11096*.

Cahlan S (2020) How misinformation helped spark an attempted coup in Gabon. *The Washington Post*.

Chen C, Chen Q, Xu J, Koltun V (2018) Learning to see in the dark. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 3291–3300.

Chen Q, Koltun V (2017) Photographic image synthesis with cascaded refinement networks. *Proceedings of the IEEE International Conference on Computer Vision*, 1511–1520.

Choi Y, Choi M, Kim M, Ha JW, Kim S, Choo J (2018) Stargan: Unified generative adversarial networks for multi-domain image-to-image translation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 8789–8797.

Choi Y, Uh Y, Yoo J, Ha JW (2020) Stargan v2: Diverse image synthesis for multiple domains. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8188–8197.

Clark A (2021) Pillow. Pillow. Accessed April 21, 2021, https://pillow.readthedocs.io/en/stable/.

Cozzolino D, Thies J, Rössler A, Riess C, Nießner M, Verdoliva L (2018) Forensictransfer: Weakly-supervised domain adaptation for forgery detection. *arXiv preprint arXiv:1812.02510*.

Dai T, Cai J, Zhang Y, Xia ST, Zhang L (2019) Second-order attention network for single image super-resolution. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11065–11074.

Dolhansky B, Howes R, Pflaum B, Baram N, Ferrer CC (2020) The deepfake detection challenge (dfdc) dataset. *arXiv preprint arXiv:2006.07397* 1(2).

Dufour N, Gully A (2019) Contributing data to deepfake detection research. Google AI. Accessed April 16, 2021, https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html.

Facebook, Inc (2021) PyTorch. Facebook, Inc. Accessed April 15, 2021, https://pytorch.org/.

Frank J, Holz T (2021) [re] CNN-generated images are surprisingly easy to spot... for now. *arXiv preprint arXiv:2104.02984*.

Geng D, Veerapaneni R (2018) Tricking neural networks: Create your own adversarial examples. University of California, Berkeley. Accessed May 4, 2021, https://ml.berkeley.edu/blog/posts/adversarial-examples/.

Goodfellow I, Bengio Y, Courville A, Bengio Y (2016) *Deep learning*, volume 1 (MIT Press Cambridge).

Goodfellow IJ, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, Courville A, Bengio Y (2014) Generative adversarial networks. *arXiv preprint arXiv:1406.2661*.

Google Developers (2019) Generative adversarial networks. Google. Accessed April 9, 2021, https://developers.google.com/machine-learning/gan.

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 (CVPR) Workshops*.

Harwell D (2019) Faked Pelosi videos, slowed to make her appear drunk, spread across social media. *The Washington Post* 24.

He K, Zhang X, Ren S, Sun J (2016) Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Itzcovich I (2018) Faced. Github. Accessed April 21, 2021, https://github.com/iitzco/faced.

Karras T, Aila T, Laine S, Lehtinen J (2017) Progressive growing of GANS for improved quality, stability, and variation. *arXiv preprint arXiv:1710.10196*.

Karras T, Aittala M, Hellsten J, Laine S, Lehtinen J, Aila T (2020a) Training generative adversarial networks with limited data. *arXiv preprint arXiv:2006.06676*.

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.

Karras T, Laine S, Aittala M, Hellsten J, Lehtinen J, Aila T (2020b) Analyzing and improving the image quality of stylegan. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 8110–8119.

King D (2009) Dlib-ml: A machine learning toolkit. *Journal of Machine Learning Research* 10:1755–1758.

Li FF, Krishna R, Xu D (2021a) Convolutional neural networks: Architectures, convolution / pooling layers. Class notes, CS231n: Convolutional Neural Networks for Visual Recognition, Spring Quarter, Department of Computer Science, Stanford University, April, 5, Stanford, CA.

Li FF, Krishna R, Xu D (2021b) Neural networks part 1: Setting up the architecture. Class notes, CS231n: Convolutional Neural Networks for Visual Recognition, Spring Quarter, Department of Computer Science, Stanford University, April, 5, Stanford, CA.

Li FF, Krishna R, Xu D (2021c) Optimization: Stochastic gradient descent. Class notes, CS231n: Convolutional Neural Networks for Visual Recognition, Spring Quarter, Department of Computer Science, Stanford University, April, 5, Stanford, CA.

Li K, Zhang T, Malik J (2019) Diverse image synthesis from semantic layouts via conditional imle. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 4220–4229.

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.

Liu Z, Qi X, Torr PH (2020) Global texture enhancement for fake face detection in the wild. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Masood M, Nawaz M, Malik KM, Javed A, Irtaza A (2021) Deepfakes generation and detection: State-of-the-art, open challenges, countermeasures, and way forward. *arXiv preprint arXiv:2103.00484*.

Mirsky Y, Lee W (2021) The creation and detection of deepfakes: A survey. *ACM Computing Surveys (CSUR)* 54(1):1–41.

Mirza M, Osindero S (2014) Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*.

Park T, Liu MY, Wang TC, Zhu JY (2019) Semantic image synthesis with spatially-adaptive normalization. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2337–2346.

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.

Ruthotto L, Haber E (2021) An introduction to deep generative modeling. *arXiv preprint arXiv:2103.05180*.

Saha S (2018) A comprehensive guide to convolutional neural networks — the eli5 way. Towards Data Science. Accessed April 5, 2021, https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

Virtanen P, Gommers R, Oliphant TE, Haberland M, Reddy T, Cournapeau D, Burovski E, Peterson P, Weckesser W, Bright J, van der Walt SJ, Brett M, Wilson J, Millman KJ, Mayorov N, Nelson ARJ, Jones E, Kern R, Larson E, Carey CJ, Polat İ, Feng Y, Moore EW, VanderPlas J, Laxalde D, Perktold J, Cimrman R, Henriksen I, Quintero EA, Harris CR, Archibald AM, Ribeiro AH, Pedregosa F, van Mulbregt P, SciPy 10 Contributors (2020) SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17, https://rdcu.be/b08Wh.

Wang R, Ma L, Juefei-Xu F, Xie X, Wang J, Liu Y (2019) Fakespotter: A simple baseline for spotting ai-synthesized fake faces. *arXiv preprint arXiv:1909.06122* 2.

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.

West J, Bergstrom C (2019) Which face is real. URL https://www.whichfaceisreal.com, which Face Is Real. Accessed April 21, 2021, https://www.whichfaceisreal.com.

Yu F, Seff A, Zhang Y, Song S, Funkhouser T, Xiao J (2015) Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv preprint arXiv:1506.03365*.

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).

Zhu JY, Park T, Isola P, Efros AA (2017) Unpaired image-to-image translation using cycle-consistent adversarial networks. *Proceedings of the IEEE International Conference on Computer Vision*, 2223–2232.

Zucconi A (2018) Understanding the technology behind deepfakes. Alan Zucconi. Accessed April 13, 2021, https://www.alanzucconi.com/2018/03/14/understanding-the-technology-behind-deepfake.

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California