Detecting DeepFakes with Deep Learning

Eric C. Tjon
San Jose State University

Detecting DeepFakes with Deep Learning

A Project

Presented to

The Faculty of the Department of Computer Science

San José State University

In Partial Fulfillment

of the Requirements for the Degree

Master of Science

by

Eric C. Tjon

December 2020

The Designated Project Committee Approves the Project Titled

Detecting DeepFakes with Deep Learning

by

Eric C. Tjon

APPROVED FOR THE DEPARTMENT OF COMPUTER SCIENCE

SAN JOSÉ STATE UNIVERSITY

December 2020

Teng-Sheng Moh, Ph.D.	Department of Computer Science
Melody Moh, Ph.D.	Department of Computer Science
Robert Chun, Ph.D.	Department of Computer Science

ABSTRACT

DETECTING DEEPFAKES WITH DEEP LEARNING

by Eric C. Tjon

Advances in generative models and manipulation techniques have given rise to digitally altered videos known as deepfakes. These videos are difficult to identify for both humans and machines. Typical detection methods exploit various imperfections in deepfake videos, such as inconsistent posing and visual artifacts. In this paper, we propose a pipeline with two distinct pathways for examining individual frames and video clips. The image pathway contains a novel architecture called Eff-YNet capable of both segmenting and detecting frames from deepfake videos. It consists of a U-Net with a classification branch and an EfficientNet B4 encoder. The video pathway implements a ResNet3D model that examines short clips of deepfake videos. To test our model, we run experiments against the Deepfake Detection Challenge dataset and show improvements over baseline classification models for both Eff-YNet and the combined pathway.

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Teng-Sheng Moh, for his weekly guidance and advice throughout the years.

I would also like to thank my committee members, Dr. Melody Moh, for encouraging me in publishing my writing and Dr. Robert Chun, for supporting my project.

Lastly, I would like to thank my family. Without their continuing support, I would have never been able to complete this project or pursue my dreams.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

1 INTRODUCTION

Deep learning techniques have made great advancements in computer vision [1]. Generative models such as generative adversarial networks (GANs) are able to produce realistic and high quality digital images like never before. These models are also able to manipulate existing images with relative ease. Using deep learning to change a person's face or identity visually is also known as deepfake. These deepfake videos are difficult to quickly distinguish from genuine, photographic images even for human observers. Therefore, it is necessary for automated detection models to identify deepfake videos as shown in Fig. 1.

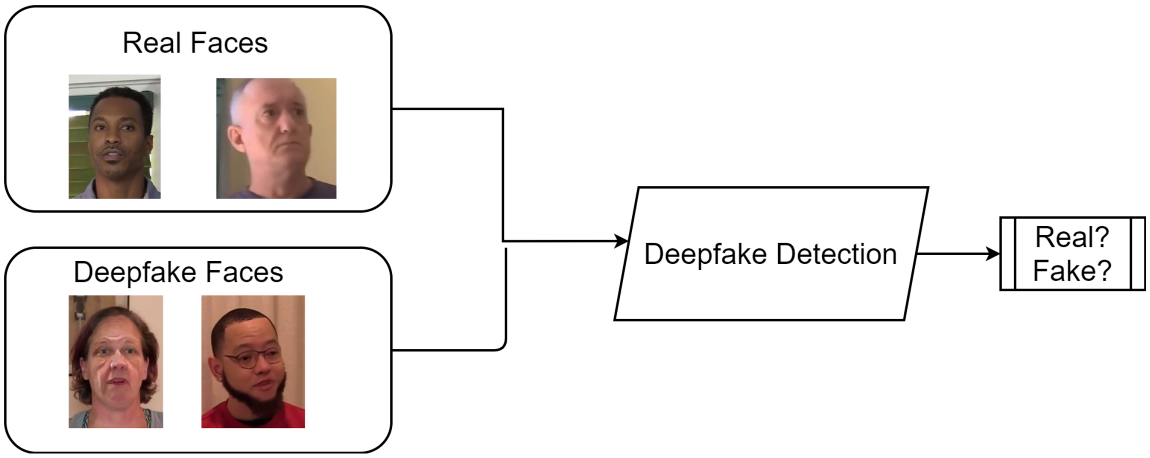

Fig. 1. Deepfake detection problem.

Prior to these developments, video forgery required expert knowledge and intensive effort to pass as real [2]. Alteration methods often included manual editing and the high cost of entry made the number of fake videos relatively small. However, deep learning based techniques allow novice users to edit videos like never before. Publicly downloadable tools such as DeepfakeLab are able to swap faces accurately on consumer hardware [3].

The ease and availability these tools increases the concern that fake images and videos can spread through social media and deceive the general public. Used maliciously, deepfake videos can misrepresent political figures and spread fake news. This misinformation can affect secure elections, so it is necessary to detect the videos before it can cause harm.

Two popular deep learning based generative models such as generative adversarial networks (GANs) and variational autoencoders. These two techniques are able to model complex swaps of faces. GANs generally create more realistic images but contain more variation between frames in the same video. Variational autoendcoders are the most popular method due to their smooth video production [1].

One way of stimulating research into this field is public competition. Specifically, one contest named the Deepfake Detection Challenge (DFDC) [1] encourages participants to build detection models to combat fake videos. DFDC is sponsored by Facebook, Microsoft, AWS, and the Partnership on AI. It is hosted on kaggle with a final submission deadline at the end of March 2020. The competition also hosts an extensive dataset consisting of over 100,000 real and deepfake videos. This dataset has noticeable improvements over prior data sets that make it suitable for research. The subjects within the videos are hired actors and actresses that give their consent to alter their appearance. Furthermore, the dataset utilizes a variety of modern techniques to alter faces that have better quality than previously published datasets.

Effective detection methods are important to combat malicious spread and use of damaging fake media. These detection methods utilize artifacts and inherent differences in generated images to successfully distinguish between real and fake. However even as detection methods improve, image generation techniques improve as well and are able to bypass detection. Therefore, it is necessary to continue research into robust detection to preemptively defend against stronger attacks.

1.1 Project Overview

This research project aims to implement a detection algorithm for classifying the DFDC dataset. This architecture would distinguish between real and fake videos by analyzing the faces present in the video. The pipeline for detection includes extracting frames from the source video, detecting and cropping a facial image in each frame, and classifying it with a deep neural network. This project contains two distinct approaches for classifying a video which focus on separate weaknesses in deepfake videos.

The first approach, called the frame based approach, examines individual frames within the video. First, the videos are preprocessed into individual frames centered around the detected face. These frames serve as the training data for an image classifier such as a 2D convolutional neural network. These models treat the videos as a collection of images. In order to classify a single video, the model creates multiple predictions for multiple frames. These predictions are combined with a strategy such as averaging to form a prediction for the entire video.

We propose a novel dual task architecture called Eff-YNet for this task. In order to highlight the differences between an altered face and its background, Eff-YNet performs both image segmentation and image classification in the same network. The image segmentation pathway is designed to segment altered pixels within the image and output a predicted segmentation mask. The image classification branch determines if the image is real or fake and outputs a binary classifier. Combining these two tasks allow the model to focus on visual irregularities. The addition of the segmentation branch aims to improve deepfake classification performance. It also produces segmentation masks that are not commonly present in other deepfake detection methods. These segmentation masks can be used to corroborate the classification prediction with visual evidence as well as provide areas to examine for human observers.

The second approach, named the sequential approach, examines a set of consecutive frames within the video. This method is designed to examine temporal data, or how consistent the video is between consecutive frames. Since the frames are consecutive, the sequential model can examine motion and consistency over time where the frame based model cannot. This approach requires video data preprocessing in order to create data in the correct format. The video is cropped into shorter video clips containing the face. Next, these video clips are loaded as a stack of consecutive frames into the sequential model, creating a single prediction for a set of frames.

These two approaches target different features of the video and are combined to form a more capable ensemble. The ensemble combines the output of the two classifiers with a weighted average, improving the performance over the individual models alone.

1.2 Report Overview

Sections two and three survey existing work in the related fields of facial manipulation, forged video datasets, and detection techniques. The facial manipulation section covers both 3D modeling and deep learning based manipulation techniques. Section three compares and contrasts different forged video datasets from different sources. The fourth section presents detection techniques based on the types of architecture and contributions towards advancing detection methods. Sections five, six, and seven cover the targeted competition and rules, implementation, and results.

2 VIDEO MANIPULATION

Video forgery involves altering a video from its original content. Different video manipulation methods exist to change the identity of a person in digital media. Earlier versions of these techniques require significant knowledge and effort to produce quality results. Recent tools are more automated and produce more realistic images.

2.1 3D Modeling Techniques

Video Face Replacement [4] is one of the first automated techniques to replace faces within videos. The technique uses a 3D model to track two faces in two different videos, with one video being the source while the other is the target. This model is able to copy the source face and paste it over the target face. In order for the source face to fit the new video, the technique accounts for facial angle, motion, and alignment and warps the face into the new position. It also calculates the best seam between the two faces and smoothly blends them together.

The Face2face algorithm [5] aims for automated face reenactment, manipulating the motions and actions of a person in a target video that match the motion and action in a source video. The method tracks the facial expressions in both videos and uses an efficient transfer between the two videos. The mouth is also warped to match the source expression. This new target face is rendered into the target video, blending it into the scene. This algorithm focuses on efficiency, so that it is able to work in a live set up.

Many more 3D modeling techniques exist to transfer faces. However, they require specific post processing to produce photo-realistic output. Additionally, these models often lack smooth motion between frames. These methods have largely favor for more advantageous deep learning models.

2.2 GANs

Deep learning techniques are able to manipulate faces and recreate them accurately. Two popular methods for this goal are Generative Adversarial Networks (GAN) and variational autoencoders (VAE) [2]. These techniques are able to learn how generate a face in a certain pose, eliminating the need for complex modeling and extensive manual input.

Generative adversarial networks are a type of generative model which aims to take a sample of data and estimate its probability distribution [6]. Generative models can either define this distribution explicitly with a function or implicitly with the output of similar data. GANs are designed to do the latter and output synthetic data that matches the sample data distribution. Generative models are used for many tasks including data generation, multi-modal output, and image manipulation.

GANs overcome many disadvantages of other generative models. Generative stochastic networks and Boltzmann machines require the use of markov chains to sample data that matches the target distribution [6], [7]. These markov chains create a new sample by repeatedly applying transitions based on probabilistic rules to an existing sample. This algorithm is computationally expensive and does not scale well to higher dimensions. GANs are able to represent complex models and can generate a sample in a single step once trained [8]. Compared to variational autoencoders, GANs generally produce higher quality output [6]. Also, GANs are able to produce image transformations more efficiently than other methods.

GAN architecture consists of two separate models called the generator and discriminator. These models are typically implemented as a deep neural network such as a convolutional neural network. A diagram of GAN architecture is presented in Fig 2.

The generator, labeled G, is the generative part of the architecture which attempts to generate samples of data similar to the training set, labeled x. Its input is a vector, labeled z, that is sampled from a uniform distribution [8]. This random noise vector allows the generator to produce varied output and capture the full distribution of data belonging to the training set.

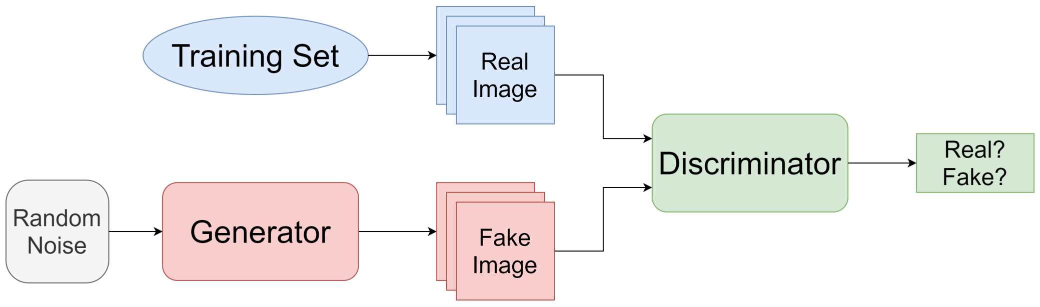

Fig. 2. GAN architecture.

The discriminator is the binary classification part of the architecture which helps train the generator and shape its output. It does so by trying to distinguish between input from the training set, labeled x, and input from the generator, labeled G(z). The input to the discriminator is a sample of data from one of these sources, and the output is a binary classification score. This score is 1 for so called real data and 0 for fake data.

These two parts of the GAN architecture are training in alternating steps. In this process, both the generator and discriminator improve simultaneously. The goal of training is to get the generator to produce realistic samples of data. If the discriminator starts off too strong and instantly rejects data from the generator, it will have no chance to improve. Therefore, the discriminator must start off weak and gradually improve. In the step where the generator improves, the discriminator's parameters, labeled $\theta^{(D)}$, are fixed while the generator optimizes its parameters. In the next step, the generator's parameters, labeled $\theta^{(G)}$, are fixed and the discriminator updates its parameters.

The cross entropy loss that the discriminator tries to minimize is shown in equation 1. The first term on the right side is the discriminator's loss on a batch of data taken from the training set. The discriminator aims to classify training set data as real. When it gives it a label of 1, this loss goes to 0. The second term on the right is the discriminator's loss on data from the generator. It tries to classify this data as fake. When given a label of 0, this term also goes to 0. The discriminator tries to minimize this loss through gradient descent [8].

$$L^{(D)}(\theta^{(D)},\theta^{(G)}) = -\frac{1}{2}\mathbb{E}_{x\sim pdata} logD(x) - \frac{1}{2}\mathbb{E}_z log(1-D(G(z))) \tag{1}$$

The generator tries to maximize this loss by fooling the discriminator [8]. It also optimizes its parameters through gradient descent. This optimization allows the generator to produce more realistic data. The generator never receives samples from the training data directly. All training is done through competition with the discriminator.

Conditional GANs extent the architecture of a plain GAN. It introduces conditions to both the generator or discriminator. These conditions represents additional information related to the data sample, such as a label, a classification, or an image [9]. The generator uses the condition along with random noise to generate data. The discriminator classifies the data by also looking at the condition and seeing if the sample matches. This condition is normally encoded as a vector and fed into the neural network layers. This process is illustrated in Fig. 3.

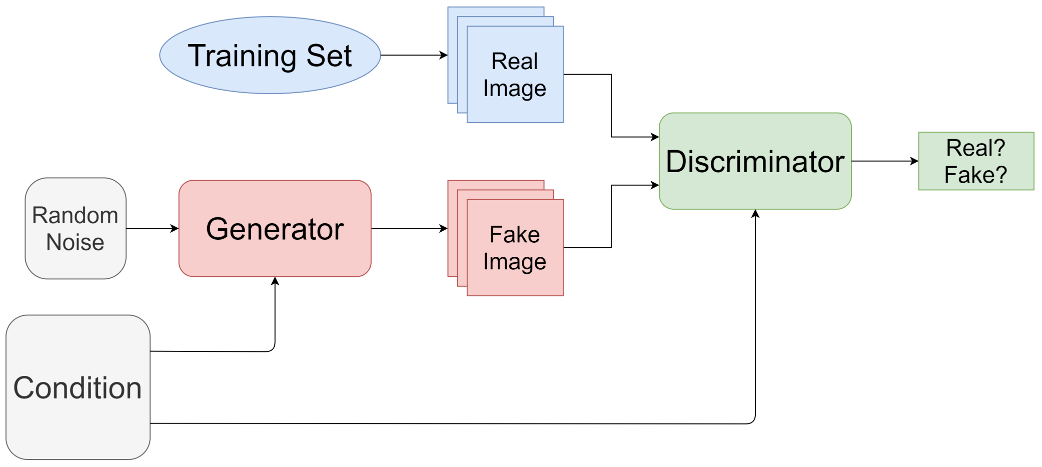

Fig. 3. CGAN architecture.

Conditions allow greater control over the output of a GAN. For example, a plain GAN trained with a set of handwritten numbers would produce a random set of numbers. A conditional GAN can generate a set of the number 3 with a matching condition. This feature along with high quality image output makes conditional GANs popular for manipulating images [10]. Many different architectures have shown success with a variety of conditions such as text, precursor images, and even videos. The following subsection outline GAN architectures that can be used for manipulating faces and producing deepfakes.

2.2.1 Deepfake CGAN Architectures

StyleGAN [11] demonstrates high quality image synthesis on a dataset of face images. The training set that it learns to imitate is a set of high quality real faces. It also learns different styles and features in an unsupervised manner. These features can be altered to control the output face in features such as glasses, skin tone, and eye color.

Two-Pathway Generative Adversarial Network (TP-GAN) [12] is a GAN designed to generate a frontal view of a face from a partial profile view of the face. It consists of a local and global branch in both the generator and the discriminator. These branches are able to preserve both local features such as texture and global features of identity in the final output.

Disentangled Representation learning Generative Adversarial Network (DR-GAN) [13] combines the task of learning a pose invariant face representation with generating a certain pose of a face. The generator uses an encoder-decoder architecture which learns an encoding of a face and decodes it into an image of a posed face. The discriminator judges the generated image against real image of the correct pose. Using this adversarial loss, DR-GAN can create a superior image over traditional methods.

Neural talking heads are another model that utilizes GANs to generate facial videos [14]. It uses two stages of training. The first meta-learning stage learns how to create realistic talking heads from photos of a person. The second fine-tuning stage rapidly learns this task for a small set of images of a person.

2.3 Deepfake Autoencoders

Deepfake autoencoders are another method of producing high quality face swaps [1]. Autoencoders were first utilized as a way to learn efficient encodings of data. The architecture consists of two neural networks labeled the encoder and decoder. The encoder takes in data as input and learns a smaller encoding of the data. This encoding is also called the latent space representation. The decoder learns the inverse operation; it takes in the encoding of the data and learns to reconstruct the original input. This technique is useful for encoding data into smaller dimensions in an unsupervised way. The autoencoder architecture is displayed in Fig. 4.

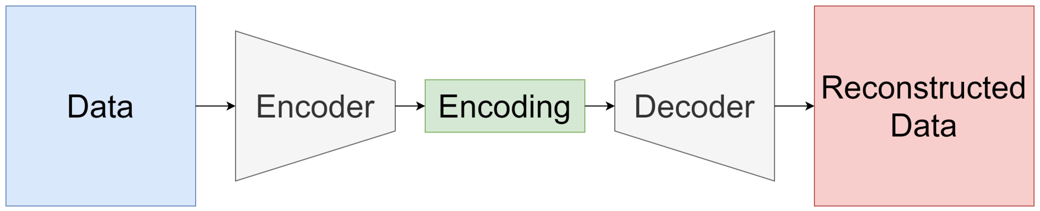

Fig. 4. Autoencoder architecture.

To turn this base autoencoder model into a generative model, variational autoencoders are used [15]. It shares the same basic structure of an encoder and decoder pair with slightly different outputs. The encoder receives the data input and encodes it into a latent space data distribution instead of an exact representation. The decoder takes a sample from the latent space and decodes it into the data's original form. This change into having a data distribution in the latent space allows the latent space variables to be more regular. Using this technique, variational autoencoders are able to vary their latent space variables to generate different data.

Algorithms collectively known as Deepfake utilize a form of variational autoencoders to swap faces. These tools were originally dedicated to creating adult content where an actress's face is inserted into a pornographic video [2]. These methods have no academic source, but their code repositories are often publicly available. These unethically produced videos received strong public backlash and brought attention to AI-manipulated videos. While the original creator is no longer active, community developed implementations of the algorithm persist and are popular for creating videos for entertainment.

Deepfake aims to replace the face in one source video with the face of another person. It contains a pair of autoencoders that are trained simultaneously [1]. The encoders from both networks share weights, so that they encode into the same latent space. The decoders learn to generate their respective faces. Manipulation takes place by encoding the source face image and decoding into the target face image.

2.4 Manipulated Video Datasets

Improvements in deep learning have made tools for video manipulation and synthesis much more accessible. Publicly available applications such as DeepFaceLab [1] allow users to create their own deepfake videos and distribute them. In order to study these deepfake videos and evaluate detection methods, a consistent dataset is necessary to benchmark different solutions.

The availability of forged videos is rather limited in natural settings. Furthermore, the ethics of collecting these videos for research is questionable since the video's subject often cannot consent to their appearance being used in this way. Combined with the computationally expensive production of multiple deepfakes, large datasets remain limited. Three notable datasets in this field include FaceForensics++ [2], CelebDF [16], and DFDC [1].

At the time of release, FaceForensics++ provided the largest dataset of deepfake videos with more than a million extracted images from over 5,000 videos [2]. These deepfakes were created with four different methods including deepfake autoencoders and Face2face. These videos were sourced from public YouTube videos so they have questionable rights. An updated version of the dataset includes about 3,000 more videos of paid, consenting actors contributed by Google.

Celeb-DF is another dataset that attempts to contribute to deepfake research [16]. This dataset aims to provide higher quality data to work on. The authors note that other datasets include poorly made deepfakes that are unlikely to fool human observers. To improve the quality of their dataset, Celeb-DF focuses on creating realistic output. The resolution of generated faces is increased to 256 by 256 pixels. This larger generated face results in smaller distortion and blur when fitting the generated face into the target video. Celeb-DF also uses post processing techniques to match the skin tone and scene lighting. Video motion is improved with smoothing consecutive frames. However, these videos have a public source, so licensing remains questionable.

2.4.1 Deepfake Detection Challenge

The Deepfake Detection Challenge [1] is an open challenge hosted on Kaggle, a website dedicated for data analysis. It is sponsored by AWS, Facebook, Microsoft, and the Partnership on AI to encourage research into.

In order to help Deepfake research, the DFDC has released a significant amount of training data. This data is comprised of over 100,000 files containing 10 seconds of video. This data is organized into 50 folders. The ratio of real to fake videos is approximately 1:4. The fake videos are manipulated through different and unrevealed methods. These methods vary in output quality and represent real world attacks. A sample of various faces from the dataset is shown in Fig. 5.

Any submitted model is tested against a hidden public test set. This test set contains videos similar in source to the training set with additional augmentations. These augmentations include frame rate reduction, resolution reduction, and lower encoding quality. The hidden nature of the test set makes it difficult to properly validate a model using just the training set.

Fig. 5. Sample data from DFDC dataset.

The main limitation placed on the computational model within the competition is a limited run time. The competition encourages training the model outside of Kaggle, but requires the final solution to run inside of a Kaggle Notebook. Once submitted, this notebook will run against a test set of 4,000 videos. The computation is limited to 8 hours, so the inference solution must classify a video under an average of 6 seconds. Furthermore, users can only submit two models a day to prevent error probing. The time and limited amount of submissions make it difficult getting reliable feedback on the private test set.

To judge the final performance at the end of the competition, two selected models from each competitor is judged against a private test set. This test set represents a greater variety of video, meant for simulating real world use cases. These videos are sourced from real, organic videos and are hosted outside the Kaggle platform. For this project, only the publicly available training set is used for training and results.

3 CLASSIFICATION METHODS

Parts of section 3 also appear in this project's conference paper [17], which has been accepted for publication in IMCOM 2021.

Detection methods designed to detect these fake videos are necessary for information security. While deep learning is used to create these videos, it is also used in automated detection. Classification models for forged videos often use a convolutional neural network (CNN) approach due to their state of the art performance in image recognition tasks. There are numerous proposed models aimed at detecting manipulated images ranging from simple to complex.

A task related to image classification is image segmentation. Instead of assigning a label to the entire image, an image segmentation model outputs a prediction for each individual pixel. This output has the effect of annotating the image with different labeled regions. We introduce U-Net as a modern image segmentation architecture.

Since these forgeries consist of video data, multiple methods exist for approaching this problem. One approach described as frame by frame is to process the video into individual frames and treat them as images for an image classification model. The model outputs a classification score for each image. These scores are combined to form the classification for the entire video.

Another method is to use a sequential approach to the data. The video is processed into a sequence of frames. A model is able to take this sequence of frames and make an overall classification. This approach has the advantage of using temporal data within the video.

3.1 Frame by Frame Architectures

These frame based architectures share significant overlap with image recognition models. While image recognition tasks typically classify multiple classes, this problem simplifies it to binary classification. This task still remains challenging as the visual

Table 1
Selected Deepfake Detection Methods

Model Name	**Architecture Type**	**Input**
MesoNet [3]	Custom 2D CNN	Images
HeadPose [18]	SVM	Images
Xception-c23 [2]	Generic 2D CNN	Images
Multi-Task [19]	Custom 2D CNN	Images, Masks
Emotion [20]	DFDC	Video, Audio
ResNet 3D [21]	3D CNN	Video
Eff-YNet (New)	2D CNN with U-Net	Images

difference between real or fake is more subtle than object recognition. These detection methods are designed to recognize small differences and artifacts of facial manipulation methods. The generated faces may contrast with the background. They may also have blending artifacts between the face and the rest of the person. A few popular 2D CNN architectures include MesoNet, XceptionNet, and EfficientNet. A general overview of the 2D CNN architecture is shown in Fig. 6.

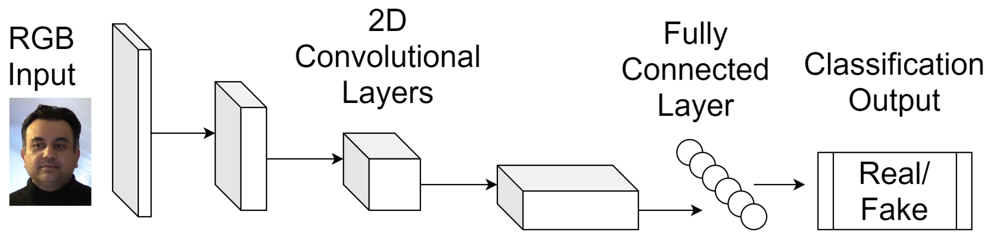

Fig. 6. 2D CNN architecture.

MesoNet is a CNN based detection method designed for detecting Deepfake and Face2Face manipulations. The design of MesoNet comes from analyzing mid-level features and only examines cropped faces from the source video. Small details of the forged video may be lost in compression while high level features will generally be correct even after manipulation. The architecture uses a deep neural network with a small number of layers to focus on the right level of detail.

A subsequent researcher uses a more complex model with higher performance called XceptionNet, which uses depth-wise separable convolutional layers with residual connections. This architecture was originally designed for general image classification, so it can receive the entire video frame as input. When adapted for facial manipulation detection, this network performed better than MesoNet [2]. However, it suffers from long training time and higher model overhead.

EfficientNets are a newer family of CNN models aimed at providing efficient scaling of resources. By balancing model width, depth, and resolution, EfficientNets outperform other models with a comparable number of parameters. There are eight different sizes of EfficientNets ranging from B0 at the smallest to B7 at the largest. These CNN models perform well on general image classification so it is natural to adapt these to detect Deepfakes.

ForensicTransfer is a more specific approach to classifying Deepfakes. This solution emphasized generalization between different Deepfake datasets. Instead of a CNN based model, it uses an autoencoder architecture consisting of an encoder and decoder. This model learns a feature representation of a facial image and compares it to a cluster of real faces. If the features are sufficiently distant from the cluster, it is classified as fake. This model performs comparably well to 2d CNN models when trained and tested on the same dataset. When tested on a new dataset, ForensicTransfer has a significant advantage.

3.2 Sequential Architecture

Sequential methods utilize a sequence of frames to detect fake videos. These methods make use of temporal data as well as expected consistency between frames. These approaches are much more varied in architecture. Some popular methods include 3D CNNs and recurrent neural networks (RNNs).

A model with a recurrent neural network can target certain weaknesses within the video movement and behavior [22]. A RNN considers previous frames as well as the current frame to generate output. The final output of the model is based on each frame of the entire video, allowing it to examine behavior over time. The basic form of the RNN uses the output from the previous data point as an additional input for the current datapoint. Fig. 7 illustrates a single RNN cell.

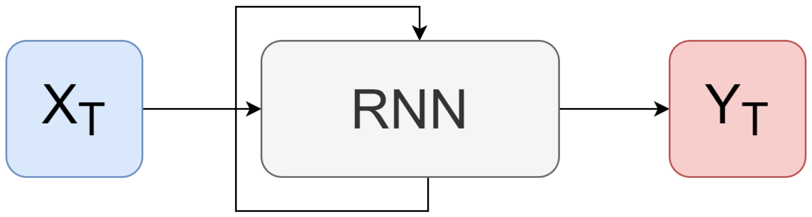

Fig. 7. RNN architecture.

One problem in the basic RNN architecture is that it does not model long range dependencies well. Earlier data points may be important, but their effect is reduced over time. Long short term memory (LSTM) is a type of RNN architecture that is able to retain data within a sequence. In addition to the output that is propogated between data points, it also keeps track of a cell state. This cell state undergoes fewer functions than the model output, allowing data to be retained.

For instance, early versions of Deepfakes failed to account for the human behavior of blinking. Using cropped images of the eyes in a video, a model that utilizes long-term recurrent convolutional network (LRCN) can be trained to detect the number of times the subject blinks in the video [23]. Next, the model can compare the detected blinking to an estimated rate of 17 blinks per minute. The model decides if the video is real or fake based on this comparison.

Other models utilizing recurrent neural networks examine the general temporal consistency within the video. The frame-to-frame transition of generated faces may have artifacts from the generation process. Examining these with an appropriate model can differentiate real and fake videos.

Head Pose [18] is a model that examines inconsistencies in 3D head poses within videos. The model crops and detects facial landmarks within the video. Next, the model estimates the 3D pose of the facial landmarks. In poorly made deepfake videos, the 3D pose of the face does not align correctly with the rest of the head. The estimated 3D pose is fed into an SVM classification model to determine if the video is real or fake.

3D CNNs have proven to be useful in video classification [24], [25]. These models take in a short clip of video and automatically classifies the action being performed in them. Recent work has adapted these networks for deepfake detection [21]. These networks focus on temporal features present in videos such as consistency between frames and human movement. An example of 3D CNN architecture is illustrated in Fig. 8.

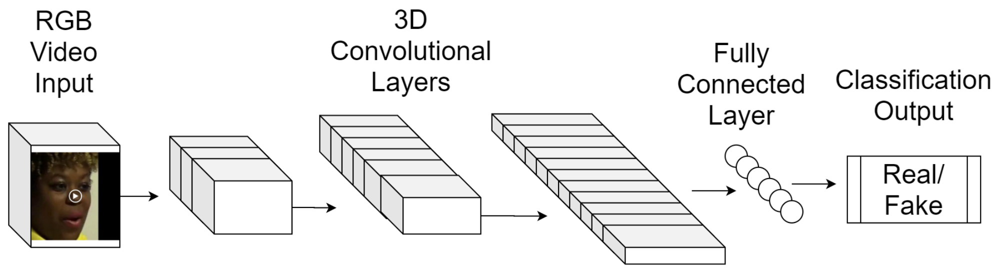

Fig. 8. 3D CNN architecture.

A model that examines human emotion [20] within video and audio has also shown promising results. Two paired networks examine the emotions present within the face and audio. The output of these two models are compared. If they are different enough, the model classifies the video as fake.

3.3 Segmentation Architecture

Image segmentation is a closely related field to image classification. This task involves identifying manipulated regions within an image. Instead of outputting a singular label, these models can produce a bounding box containing an object or a semantic mask of the image. This mask represents the image, where different classes are shaded in different colors. Image segmentation is commonly used in fields of medical imaging and traffic object detection.

U-Net [26] is one popular model for image segmentation. Earlier segmentation methods used a sliding window approach, where the image was divided into different overlapping patches. A deep learning model would then classify the center of each patch based on its surroundings. The sliding window approach was costly and slow. The U-Net architecture attempts to address these problems with an autoencoder, consisting of a downsampling encoder and an upsampling decoder as illustated in Fig. 9.

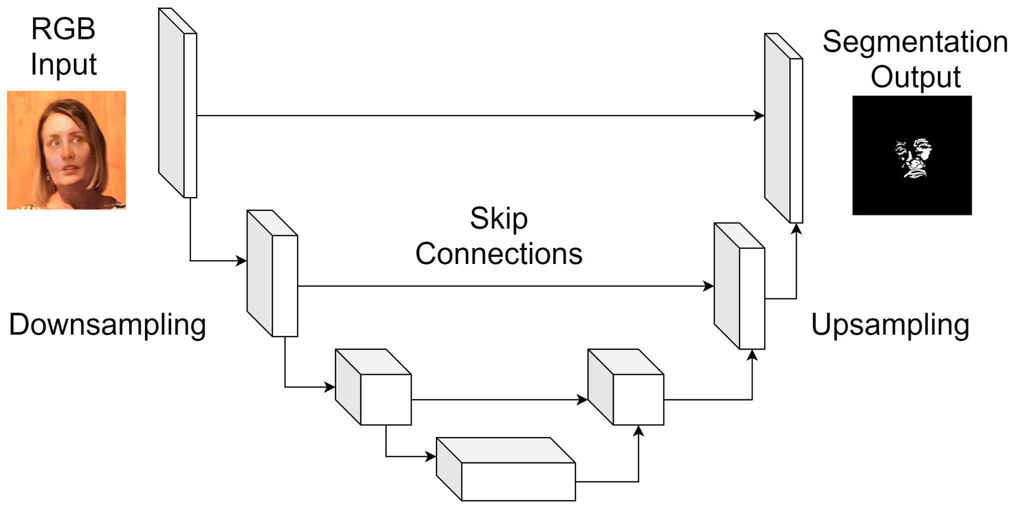

Fig. 9. U-Net Architecture.

Since the decoder increases the size of the output to match the input, these sliding windows were no longer needed. The U-Net also contains skip connections between the encoder and decoder. These connections carry information between encoding and decoding layers of the same size, ensuring structural information is not lost within the bottleneck of the encoder. This feature allows the decoder to upsample the encoded features into a segmentation mask more accurately. This architecture represents a U shape with the encoder descending on the left and the decoder ascending on the right.

Since EfficientNets show state of the art performance in image recognition tasks, it is beneficial to include them as an encoder. Eff-UNet [27] is a recent model combining the effectiveness of an EfficientNet B7 model in a U-Net for fine grained segmentation. This model was tested on road traffic images and won first place in the IDD lite segmentation challenge for AutoNUE 2019. Our proposed model adapts Eff-UNet to deepfake detection and segmentation.

Y-Net [28] is an extension of the U-Net architecture which adds a classification branch to the end of the encoder. This combined architecture was originally created for segmenting and classifying breast biopsy images. The U-Net portion segments the image into cells of different type. The classification branch takes the encoded features and classifies the image as benign or malignant. The authors found that this model was effective in handling both tasks at the same time. We adapt this model for our deepfake detection architecture.

3.4 Multi-Task Learning

Incorporating a segmentation model into a detection method has been shown to improve performance before in detecting Deepfakes. Information gained through the segmentation branch and classification branch is shared throughout the model, improving the performance of both branches [19]. One multi-task learning approach combines three different tasks to identify and segment manipulated faces in video. The multi-task model is designed to output a classification, segmentation, and reconstruction of the image. The loss for all three of these tasks are summed up with equal weights to form the total loss. This model is trained to minimize the total loss, so it will learn all three tasks simultaneously.

Multi-Task learning has promising results, but more research is needed to advance the model. The existing model uses a simple segmentation and classification branch, leaving room for improvement. Using a more advanced segmentation model would improve results. My proposed model would combine a segmentation model that utilizes U-Net with a classification branch. These two tasks are related for detecting Deepfakes and altered digital media. Using an advanced CNN within the U-Net could strengthen the model as well. EfficientNets are state-of-the-art and can be adapted for this use.

4 ARCHITECTURE IMPLEMENTATION

We developed two distinct pipelines to detect and classify deepfake videos. The first pipeline is the frame based approach, which divides the video into individual frames. These frames serve as input to a 2D CNN. The second pipeline is the sequential approach which uses a short sequence of frames as the preprocessed input for each video. This sequence of frames

Each architecture contains two main parts. The first part processes videos within the dfdc dataset into face centered data used as input. This process requires reading in each frame, detecting and aligning the faces with a face detector, and saving the result in the correct format. The second part of the architecture uses

4.1 Frame Based Implementation

The frame based approach transforms the videos into a set of frames. Deep learning classification models take these individual frames as input and generate a classification. The classification for an entire video is made from the average of its frames. In this subsection, we describe a pipeline that generates both frames and masks that highlight the altered pixels. The pipeline then uses a novel dual task architecture called Eff-YNet to predict the classification and segmentation mask based on the frames. The pipeline is illustrated in Fig. 10.

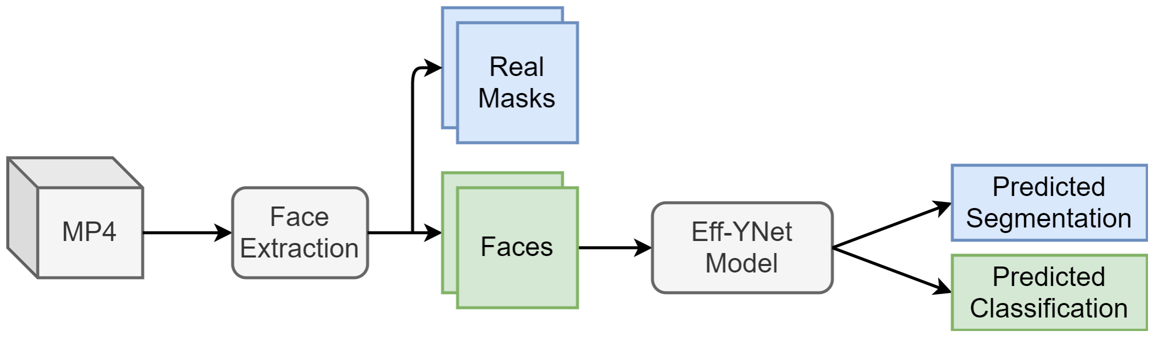

Fig. 10. Eff-YNet detection pipeline.

4.1.1 Face Extraction

The face extraction module is designed to process the video into both frames and masks which are used by the Eff-YNet model. We use the opencv package to open and read the frames from each video. Since it would be impractical to extract all frames of each video for training, we extract 30 frames per video. There are many ways to sample 30 frames out of the 10 second video. One method is to choose 30 consecutive frames out of the video. These frames would contain information on motion over time, but offers a limited portion of the video to view. For the frame based method, we sample 30 evenly spaced frames from the video.

We used the MTCNN face detector in the facenet-pytorch package to identify and crop faces within a video frame. This detector was chosen for its combination of speed, accuracy, and integration with PyTorch. This face detector uses cascaded CNNs in a three stage process. The first stage rapidly produces candidate windows or bounding boxes. The second stage refines these candidates and rejects false positives. The third stage outputs the detected face as well as facial landmarks. MtCNN allows the use of batches as input, so we collect 30 frames to input into the face detector at one time.

Once the face is detected, the pipeline calculates a bounding box with a margin that adds 30 percent of the original length to each side. This margin allows the crop to include more of the background, which is useful in identifying out of place altered pixels. Next, these bounding boxes are cropped and are saved as a png file in its original resolution. The model resizes these crops at training and test time.

Real videos and their corresponding fakes share identical frames at the same timestamps. Only the faces are modified between videos. To efficiently process the data and ensure consistency, bounding boxes are only calculated for real videos. The corresponding fake videos use the same bounding box at the same time stamp to generate a crop of the face.

4.1.2 Segmentation Mask Generation

For the Eff-YNet architecture, we aim to create a model that segments altered pixels from real ones. We train the model with target segmentation masks to achieve the goal. These masks are generated through finding the pixel-wise differences between the original and manipulated videos. The segmentation target for an original video is an empty mask. The segmentation target for a fake video is a mask of the altered pixels.

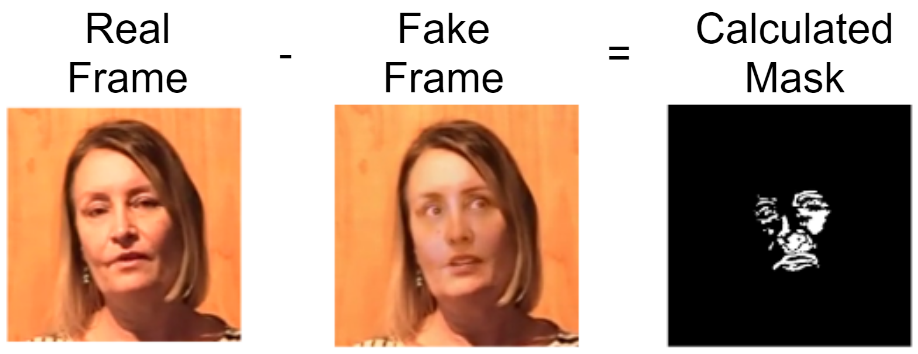

Fig. 11. Segmentation mask generation.

We use structural similarity index (SSIM) to generate the real segmentation masks. The SSIM metric aims to measure visually perceivable differences between two images based on the surroundings and value of each pixel. More different pixels are giving a greater value in the range of 0 to 255 in the mask. For use in our segmentation model, We use a threshold of 50 to convert the SSIM mask into a binary mask with 0 for unaltered and 1 for altered. An example of mask generation is presented in Fig. 11.

4.1.3 Eff-YNet Implementation

We propose Eff-YNet to classify Deepfakes. This model aims to identify the pixelwise differences between real and fake. Eff-YNet derives its name from the combination of a strong 2D CNN called EfficientNet B4 [29] with a UNet segmentation model [26]. A classification branch is added to this combination, creating a YNet. A diagram of the Eff-YNet model is presented in Fig. 12. The model takes in an input image of size 384 x

384 and predicts both a classification and segmentation mask. While other input sizes are usable, we chose this size since it is close to EfficientNet B4's native resolution of 380 x 380. This size is also divisible by 32, a requirement of the UNet implementation.

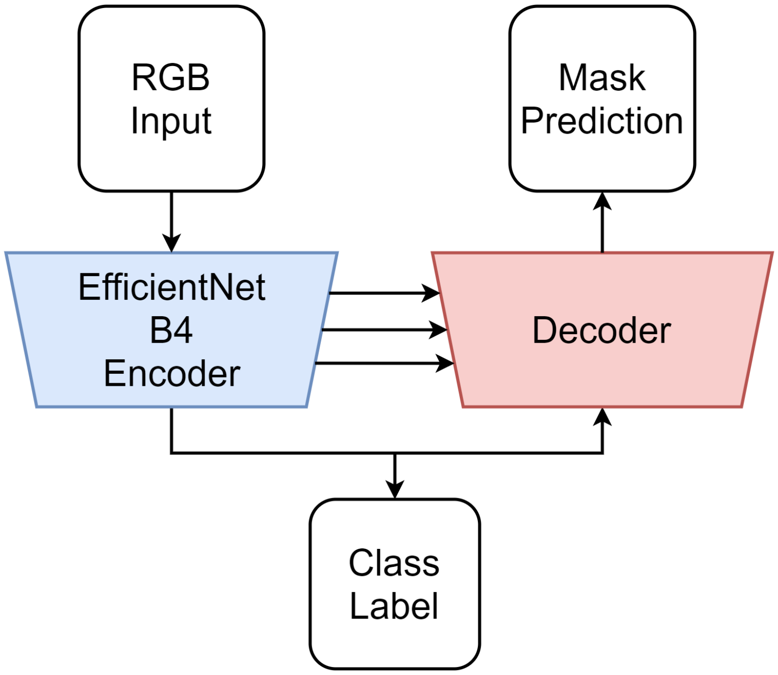

Fig. 12. Eff-YNet model architecture.

For this model, we start with an EfficientNet B4 CNN model as the encoder. This encoder takes in the image as the input and outputs a number of features which represent an encoding. These features are fed into a classification branch and a segmentation branch. The classification branch consists of a fully connected layer which takes this features and makes a binary classification. The encoder and classification branch combined are identical to the baseline architecture, which also uses EfficientNet B4. On its own, this 2D CNN is able to classify deepfake images with reasonable accuracy. The addition of the segmentation branch improves training and allows the model to focus on pixel differences between altered areas and the original image.

The segmentation branch referenced above is also called the decoder. It performs the nearly the inverse task of the encoder. Starting with encoded features, it recreates the original image as a segmentation mask. This decoder is attached to the EfficientNet CNN within the U-Net architecture. The decoder receives additional intermediate features with skip connections. These skip connections connect corresponding layers in the encoder and decoder which allows the decoder to use contextual information in its recreation of the original image. This architecture is implemented with the segmentation models pytorch library.

The goal of this model is to train the model to identify altered pixels in order to generalize better. Simply training a 2D CNN model may cause over fitting due to recognizing faces. By adding the segmentation masks, the model can learn features of altered areas and segment them, rather than looking at the identity of the person. These tasks are able to learn from each other due to the shared encoder. We also test an EfficientNet B4 model as a baseline for comparison.

4.2 Video Based Implementation

The video based approach processes each video into smaller video clips that are cropped around the detected face. Next, consecutive frames in these clips are stacked to create a 3D input. The classification model for this 3D input is a 3D CNN called ResNet 3D. We also test a similar, related architecture called ResNet(2+1)D. These 3D CNNs operate similarly to the 2D CNN classification models, but also add time as an additional dimension. This addition allows the model to examine spaciotemporal features such as consistency and motion between consecutive frames. These 3D CNNs can take a variable number of frames. We chose to use 32 consecutive frames of 112 x 112 resolution each to balance batch size and training time.

4.2.1 Video Clip Extraction

The 3D CNNs expect a video-like stack of consecutive frames as the input. However, we also wish to emphasize the face in each video, cropping out the irrelevant areas. If we use consecutive crops generated with the frame based face extraction, it would cause the background to move while the face stays in the center of the frame.

In order to generate a cropped video clip, we use regions of interest that are available on github [30]. These bounding boxes are made with a face extraction algorithm that samples frames throughout the video. These frames serve as the input to an MtCNN face detector, which generates a list of bounding boxes. The algorithm then detects stacks of bounding boxes which overlap, which represents a single person's face moving over time. Then, it calculates a larger bounding box which contains the entire overlapping stack. This large bounding box is cropped for consecutive frames, so the background stays static while the face moves between frames. The crops generated from the video are saved as an mp4 clip.

4.2.2 ResNet 3D Implementation

For video classification, we implement ResNet 3D and ResNet (2+1)D as provided by the torchvision library. These models have shown success in video task recognition. We adapt these models for binary classification, training them to distinguish between real and fake videos.

The video clips produced by the algorithm above contain various lengths, resolutions and aspect ratios. For use in the model, these clips are altered at training and testing time. We extract 32 frames per video clip to standardize the length. Each frame is padded into a square aspect ratio and resized to 112 x 112.

These 3D CNN models are designed to examine consecutive frames in a sequence. It examines consistency and smooth motion that deepfake videos have difficulty demonstrating.

5 TRAINING

5.1 Validation Split

The training data underwent an 80-20 validation split based on folders. Folders 37-47 were chosen for validation. The remaining folders numbered from 0 to 50 were chosen for training. In the training data, the source and manipulated videos are constrained to the same folder. However, the actors and actresses in the video may appear in multiple folders. This property causes a minor validation leak, but it is minimized through the folder-wise validation.

5.2 Augmentations

We found that augmentations are necessary to prevent the model from overfitting. The high tendency to overfit may be due to the model learning how to recognize faces rather than identity-independent features. To combat this, we use a strong set of augmentations as well as selective cutout.

Augmentations include flips, rotations, contrast adjustments, and image compression to simulate variety in collected videos. These augmentations allow the model to train for many epochs and avoid overfitting. The different augmentations are applied randomly for each image. In the video dataset, a consistent augmentation is applied to frames from the same video so it does not affect the consistency within a video.

The cutout augmentation is designed to reduce the model's attempts to learn specific identities in the model. We use both landmark cutout as inspired by the first place solution [31] and random cutout. The landmark cutout removes one of the nose, eyes, or mouth as bounded by the face detection model. The random cutout removes a 128 x 128 region from the image. The landmark cutout was applied at a higher rate than the random cutout, 60 percent and 10 percent respectively.

5.3 Hyperparameters and Loss

We found a low learning rate of 1e-4 was effective for our models. Too high of a learning rate prevented the model from converging, possibly due to the separation of the encoder and decoder. The training loss would oscillate too wildly for the model to make a confident prediction. We used an Adam optimizer to train the models.

For the U-Net model, both the segmentation and classification branches were trained simultaneously. The segmentation loss is dice loss between the predicted and actual segmentation mask. The classification loss is binary cross entropy loss between the predicted and actual classification. The total loss for the model is the average of these two losses.

5.4 Batch Sizes and GPU Memory

All models were trained on a system with a Nvidia Geforce 2080 TI GPU with 11GB of memory. In order to train them with GPU acceleration, the models, input data, and training gradients must fit into the GPU's memory. Due to this limitation, model size and training batch sizes must be considered to balance training time with performance. A table of model memory requirements during training is presented in Table 2.

Table 2
Model Parameter and Memory Size

Model	**Trainable Parameters**	**Input Size (MB)**	**Total Size (MB)**
Eff-YNet	19,476,890	1.69	2066.79
EfficientNet 2D	17,550,409	1.69	1627.18
ResNet3D	33,166,785	4.59	1221.37
ResNet(2+1)D	31,300,638	4.59	3117.69

The 2D pathway models of Eff-YNet and the baseline models have a batch size of eight, consisting of eight RGB images of size 384 by 384. Larger architectures such as using EfficientNet B7 as the encoder were possible, but it took much longer to train when reducing the batch sizes.

The 3D pathway models of ResNet3D and ResNet(2+1)D had a batch size of 4, with each input consisting of 32 consecutive images of size 112 x 112. We chose this resolution since pretrained models were available with this input size. Increasing the number of consecutive frames and the input size may increase performance, but since the video clips must be processed as a whole, it is more difficult to fit into memory compared to the 2D pathway.

6 DEVELOPMENT OF MODEL

In developing the final solution, the overall pipeline went through multiple iterations of classification models, training methods, and datasets. Since it would be difficult to directly compare all the methods without extensive ablation studies, the development of detection techniques are presented here. These intermediate results were essential in developing our final, proposed models and methods.

6.1 Initial Baseline Models

6.1.1 Deepfake Faces Dataset

While all the datasets used in this project originate from the DFDC video dataset, the process of extracting faces and frames affect their quality. Differences in margin, resizing, and frame spacing can influence how a model fits to the data.

THe first dataset used in developing a detection method is the deepfake faces dataset publicly available on kaggle. It contains a face crop from the first frame of each video as detected by the python face recognition package based on dlib. It excludes about 20 percent of the training videos where no face or multiple faces were detected. These crops are then resized to 224 x 224 and saved as a png. This collection was chosen for its early availability and compatibility with pretrained models.

However, it has many downsides compared to future models. The dlib face extractor processes single frames at a time and had slow performance. Furthermore, the crops were resized without regard to the original aspect ratio. This resizing resulted in stretched faces, making it harder for the model to have a consistent crop. The faces also included no margin, so the image often cuts off the edges of the face.

Since the DFDC dataset is heavily unbalanced, containing far more fake faces than real faces, we experimented with balancing the data by adding more real faces. To do this, we added real faces from the Flicker Faces High Quality (FFHQ) dataset to the deepfake faces dataset. Initially, this technique seemed promising as we could use all the fake videos within the DFDC while adding even more real faces to train on. Training the model on this combined dataset also produced good results, as the training and validation loss decreased compared to the original dataset. However, testing these models on the original DFDC dataset showed worse results. We suspect that the FFHQ dataset has major differences compared to the fake faces in the DFDC dataset, allowing the model to easily detect these real faces. This attempt at balancing inflates the score while decreasing actual performance.

6.1.2 2D CNN Model

The initial classification models consisted of 2D CNNs to act as a baseline and quick proof of concept that deep neural networks fit the data suitably. The first model is ResNext from the ResNet family of CNNs. The input size of this model matches the 224 x 224 resolution of the deepfake faces dataset, the first easily available dataset that we collected. The second model, XceptionNet, was also tested on this model. This model was chosen for its use as a baseline in other publications and increased resolution of 299 x 299.

These models started to weakly fit the training data after many epochs and performed poorly on validation. We suspect that the performance can be improved with increased resolution, more data, and more complex models.

6.2 LRCN Pipeline

The next class of model that we implemented is the Long-term Recurrent Convolutional Network (LRCN) which aims to add examining temporal consistency to the 2D CNN model. The LRCN model was originally designed for video activity classification. It consists of two parts: a 2D CNN and a long short-term memory model (LSTM). The 2D CNN base converts each frame of size 224 x 224 into a set of 1792 features. Next, the LSTM model takes in a list of frame features and outputs a list of outputs, one for each frame. We feed the final output representing the last frame into a fully connected layer to create the final prediction. An overview of the LRCN architecture is presented in Fig. 13.

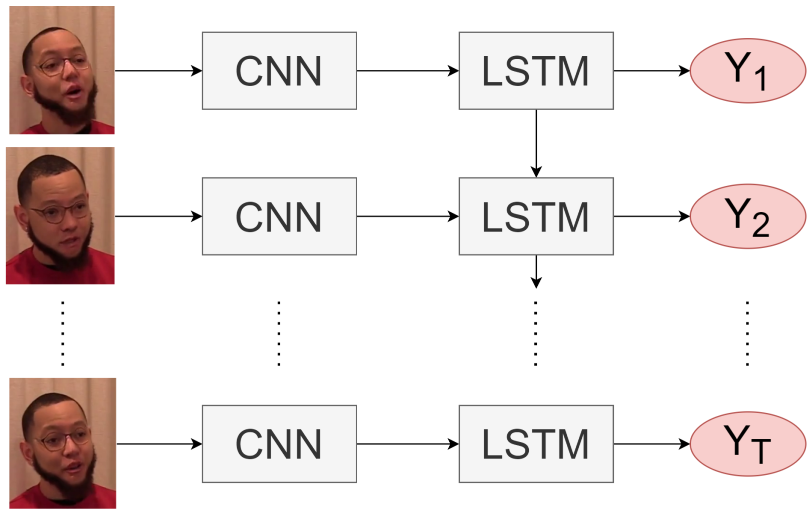

Fig. 13. LRCN pipeline.

6.2.1 LRCN Dataset

The second dataset is the lrcn dataset which was designed for use in sequential models. Sequential models use multiple frames to create a prediction, so this dataset contains 10 faces per video. This dataset uses the MtCNN face detector implemented in the facenet pytorch package. The video is sampled at 10 equally spaced frames, and the first face returned by the face detector is cropped. If less than 10 faces were cropped, the video is resampled at different points to obtain the requisite number of frames. These crops are resized into 224 by 224 pixels.

This dataset has small improvements over the previous one. It uses MTCNN as the face detector, which performs faster and more accurately than the dlib face detector. It also has multiple frames per video and includes videos with multiple faces detected. However, these crops use little margin, resulting in a zoomed image of each face.

6.2.2 LRCN Model

The LRCN model aims to model long range temporal dependencies over multiple frames. We chose to train the model using 8 frames per video due to space and time constraints. The model consists of a 2D CNN encoder and a RNN decoder. The 2D CNN is a pretrained EfficientNet B4 model with an input size of 224 by 224.

EfficientNets are a family of models that are designed for efficient scaling. There are eight different models varying in size, labeled from B0 to B7. For this project, EfficientNet B4 was chosen for its balance of parameters and accuracy. The encoder is initially pretrained on the ImageNet dataset. When we tested the model with all weights randomized, we found that it would not converge in a reasonable amount of time.

The encoder outputs 1792 features per image which is then fed into the RNN decoder. The RNN is a LSTM model which takes in a stack of features from 8 frames. The final prediction is made with the LSTM's output on the final frame, after it has seen the features of the prior 7 frames. The LSTM outputs 256 features at the final frame. These features are fed into two sequential fully connected layers. These layers combine the features and output the final prediction. This LSTM adds temporal modeling capability over the baseline model. This model hopes to classify real and fake videos based on consistency of faces as well as quality over time.

We found that the LRCN model was able to fit the training data well, but overfit rapidly. The improvements in validation loss stagnated after a few epochs. We suspected that this overfitting was inherent in the LRCN model, since it relies on a 2D CNN to first extract the features. The LSTM decoder which models time dependent features only works on the extracted features, so it is limited to what it can see. Furthermore, the data generation is mismatched since the frames are not consecutive and rely on the face detector for framing.

7 RESULTS

Parts of section 7 also appear in this project's conference paper [17], which has been accepted for publication in IMCOM 2021.

For frame level inference, evaluate the Eff-YNet model as well as a baseline EfficientNet B4 model with area under ROC curve (AUC) for the validation set. The ROC curve is illustrated in Fig. 14. The baseline model trained quickly and fit the training set well, but produced slightly worse results for the validation set. The Eff-YNet model took longer to train, but yielded improvements in classification over the baseline model.

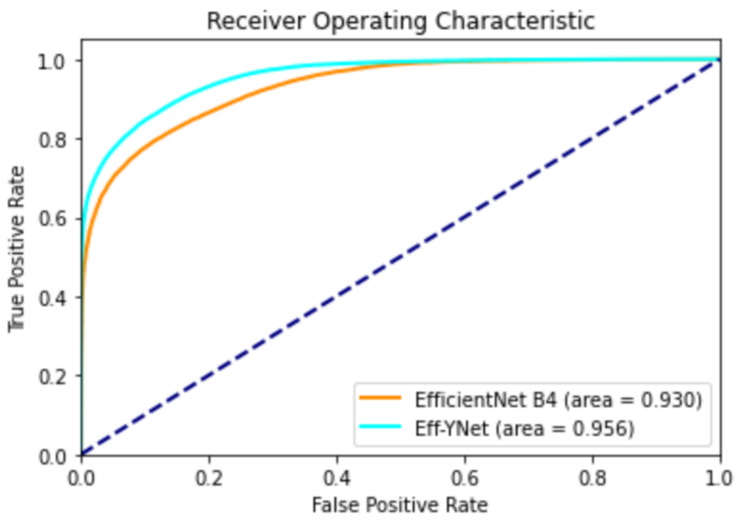

Fig. 14. Frame level ROC for tested architectures.

For video level validation, we adapt the two frame based methods above by averaging their predictions on 16 frames per video. We also examine the 3D CNN models of ResNet 3D and ResNet (2+1)D as well as an ensemble. This ensemble consists of the weighted average of Eff-YNet and ResNet 3D. Eff-YNet is weighted as twice the weight of ResNet 3D due to its greater performance.

We benchmarked our results against other methods that test the DFDC dataset. These results are summarized in Table 3. Since the DFDC dataset is fairly new, direct

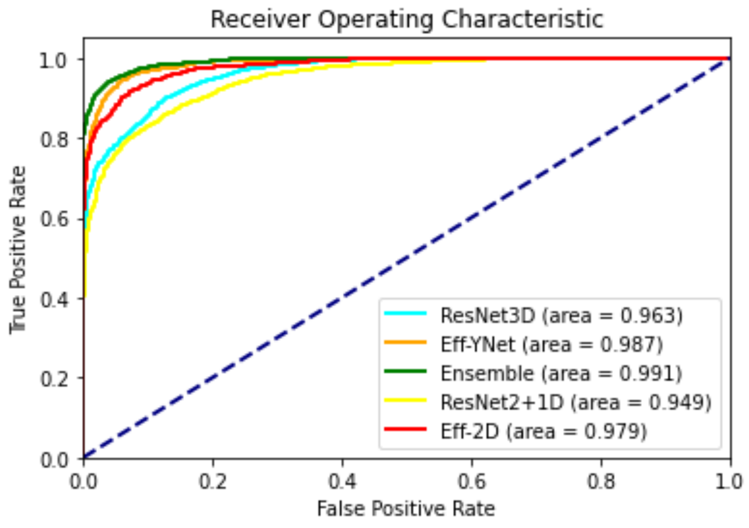

Fig. 15. Video level ROC for tested architectures.

comparisons are limited. We describe 4 methods which are tested in [16] as well as a recent video method [20] that utilizes emotional cues.

1) MesoNet [3] is a 2D CNN based detection method. It is specifically designed to detect deepfake videos with a midsized network.
2) HeadPose [18] uses an SVM classifier to detect inconsistencies in head poses and angles.
3) Xception-c23 [2] is a general purpose 2D CNN adapted to detect deepfakes on the FaceForensics++ dataset.
4) Multi-Task [19] utilizes a 2D CNN with a segmentation and reconstruction branch. It uses these tasks to aid in training and detecting deepfakes.
5) Emotion [20] uses Siamese networks to evaluate both face and speech emotion at the same time and detects deepfakes on how well the emotions match. The AUC score for this model is based on video-level predictions.

It is important to note the differences in training and testing environments in order to make a fair comparison. The results for methods 1-4 were created using publicly available

Table 3
AUC Scores for DFDC

Model	Training Dataset	Inference Type	AUC %
MesoNet [3]	Unpublished	Frame	75.3
HeadPose [18]	UADFV [18]	Frame	55.9
Xception-c23 [2]	FaceForensics++ [2]	Frame	72.2
Multi-Task [19]	FaceForensics++	Frame	53.6
Audio-Visual [20]	DFDC	Video	84.4
EfficientNet B4	DFDC	Frame	93.0
Eff-YNet (Frame)	DFDC	Frame	95.6
Eff-YNet (Video)	DFDC	Video	98.7
ResNet3D	DFDC	Video	96.3
Eff-YNet/ResNet 3D Ensemble	DFDC	Video	99.2

code and weights in [16]. Since the tests are not done by the original authors of each model, we rely on the faithful adaptions to compare our results. Furthermore, these models are trained on different datasets such as FaceForensics++ [2] and tested on a subset of the DFDC Dataset. Our model was both trained and tested on subsets of the DFDC dataset, so we would expect better results. Both our model as well as models 1-4 report a frame-level AUC, where results for each video frame are included.

The Emotion model [20] operates on video-level results due to using video input. The authors note that they select 18,000 random samples from the DFDC dataset. Since this model both trains and tests on the DFDC dataset, we believe it serves as a fairer comparison compared to the other benchmark models.

7.1 Segmentation Results

Our model also produces valuable segmentation results that are not present in other models. Since most work in deepfake detection focuses on classification, we could not find a comparison for our quantitative segmentation results such as dice loss. Instead, we examine the predicted segmentation masks qualitatively.

We find that the segmentation mask prediction generally matches the classification prediction. If the predicted mask is empty, the model predicts the image to be real. If

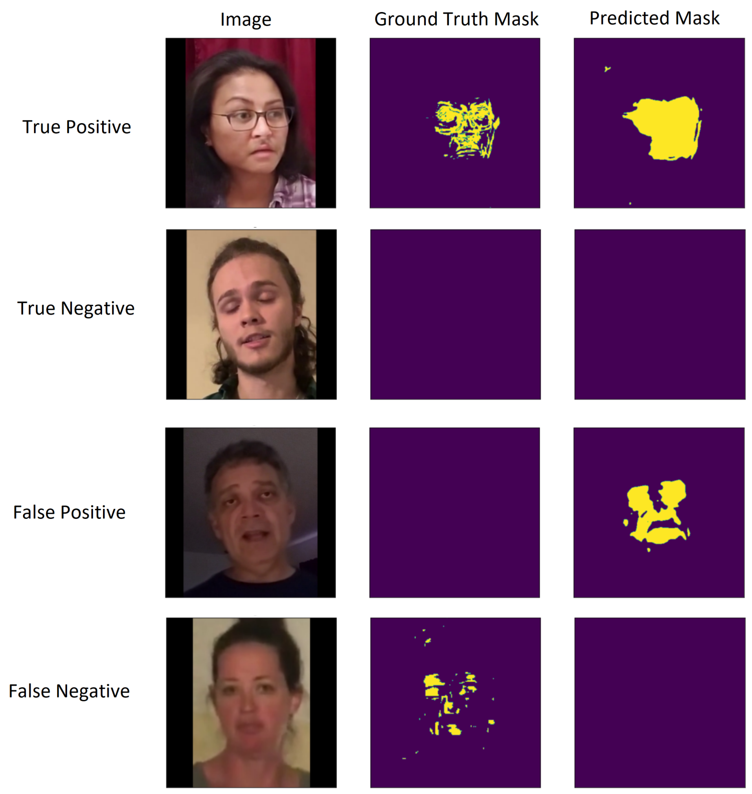

Fig. 16. Sample segmentation results.

there is a large predicted mask, the model is confident that the image is fake. We present selected results of the segmentation output in Figure 16. The model tends to produce correct results on larger and clearer images, while incorrect results seem to stem from blurry and small source images.

These segmentation masks are useful in understanding how the image influences Eff-YNet's classification label. It is able to identify the areas in the image that may be altered by a deepfake algorithm. These results may be useful in real-world applications as well. If a video is suspected of being a deepfake, the segmentation output can identify the faces and areas that are manipulated. This feature poses an advantage over other classification schemes.

7.2 Discussion of Results and Limitations

The Eff-YNet approach seems promising as it generalizes well and avoids overfitting the data. The Eff-YNet approach detects deepfakes based on differences between real and fake regions and produces useful segmentation masks. We used a multitude of augmentations ranging from random crops, contrast and brightness adjustment, and cutting out part of the image. These augmentations encourage data diversity and generalization since the same image can be augmented in different ways. Cutout seemed to slow the training rate, but also improved the final result. This result suggests that preventing the model from identifying specific faces allow it to focus on pixel level differences.

This model is only trained on the DFDC dataset, which includes both real and fake versions of each video. We utilize these pairs of videos to create segmentation masks, but pairs of videos may not be always available in different datasets. The core concept of segmenting altered regions from real regions may also be more difficult as deepfakes improve. Detection methods that utilize harder to model features such as temporal consistency [21] or emotional cues [20] would have an advantage with more complex deepfake generation.

While Eff-YNet uses visual features to identify deepfake videos, it does not take temporal features and consistency into account. In order to incorporate these features, we also include 3D CNN models trained on the DFDC dataset, namely ResNet 3D [25] and ResNet(2+1)D [24]. These models rely on short clips of video to detect any digital alterations. The ResNet 3D model performs 3D convolutions while the ResNet(2+1)D model imitates 3D convolutions with separate 2D and 1D pathways. These two pathways represent spatial and temporal features respectively.

We found that these 3D models fit the data better than LRCN models in our initial testing, supporting other research results [21]. These 3D models do not outperform

Eff-YNet and the baseline in video level AUC scores. However, combining ResNet 3D and Eff-YNet in an ensemble yielded improvements, despite the lower score. It is interesting to note that combining Eff-YNet with the baseline EfficientNet model did not improve the score. These results suggest that it is advantageous to rely on multiple approaches to detect deepfake videos.

7.2.1 Model Size Limitations

Training complexity and memory efficientcy must also be taken into account. In the field of CNN-based classification models, the number of trainable parameters is positively correlated with model performance. Therefore, we should expect performance increases with larger models.

The baseline EfficientNet B4 model produces comparable classification results at a much smaller model size and training time. If deepfake classification is the only desired result, a larger 2D CNN model may prove to be better than an Eff-YNet of comparable size. 3D models offer a unique approach, but require much more space since they must process an entire video clip instead of individual images. It is likely that they underperform 2D solutions of comparable size.

7.2.2 DFDC Competition Winners

The DFDC competition winners are the best performing models in this area of deepfake detection. Even though they are not formally published and not benchmarked to a public data set, the winning architectures are described in detail online. It would be beneficial to compare proposed models to these winning solutions.

The fifth place winning architecture [30] used an ensemble of multiple 3D CNN models and a few 2D CNN models. By using differently trained models with different input, they were able to create a well rounded ensemble capable of generalizing well.

The first place solution [31] used an ensemble of five EfficientNet B7 models. This surprisingly simple architecture shows the power of using larger models that utilize space efficiently. The raw computing power of these 2D CNNs were able to detect deepfakes from single images alone, averaged over a whole video. This solution's performance shows that scaling up simple solutions is comparable to using more complex methods.

8 CONCLUSION AND FUTURE WORK

Facial manipulation methods utilize traditional 3D modeling and computer graphical methods to change the facial identity of people in videos. Deep learning models can also alter facial images to a high degree of accuracy. This technology may possibly be used for malicious intent such as misrepresenting a political figure. Current manipulation techniques are difficult for humans to identify, so automated detection methods are needed to identify fake videos. Currently, convolutional neural network based classification models are able to identify facial manipulation with reasonable accuracy.

Classification models exist for known manipulation methods, but these models are vulnerable to newer attacks. Models need to detect a wide range of facial manipulation techniques to implement effective detection. Benchmarks and challenges such as the Deepfake Detection Challenge promote competition and stimulate research into this field.

In this paper, we present and evaluate a novel Eff-YNet architecture for both classifying and segmenting video frames. This model utilizes inherent differences between the altered faces and the background image to identify deepfake videos. This model is evaluated on the Deepfake Detection Challenge dataset and compares favorably to recent detection methods. Furthermore, our model produces valuable segmentation results that identify possible altered regions and explain the classification label. We also evaluate a ResNet 3D model that also performed well on videos. An ensemble of these two methods yielded better results than each model alone.

Further research is needed to create a robust model that can handle different deepfake generation algorithms. The Eff-YNet model is limited to examining visual differences within individual frames. Other models [20], [21] can incorporate more complex features such as audio, motion, temporal consistency, and emotion. Currently, our 3D CNN model of ResNet 3D can model temporal features and provides reasonable accuracy. This model addresses weaknesses in the Eff-YNet model and improves the performance in an ensemble. In the future, we would also like to test our Eff-YNet model on different datasets and compare the results with other state of the art methods.

The winners of the Deepfake Detection Challenge also provide promising research direction. The top five solutions in the competition all had varying model architectures and briefly presented their results at CVPR2020. The first place solution [31] surprisingly used a relatively simple architecture of ensembled 2D CNN EfficientNets. These top solutions created for DFDC will prove useful in creating robust detection methods.

Literature Cited

[1] B. Dolhansky, J. Bitton, B. Pflaum, J. Lu, R. Howes, M. Wang, and C. C. Ferrer, "The deepfake detection challenge dataset," *arXiv preprint arXiv:2006.07397*, 2020.

[2] A. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 1–11, 2019.

[3] D. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "Mesonet: a compact facial video forgery detection network," in *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pp. 1–7, IEEE, 2018.

[4] K. Dale, K. Sunkavalli, M. K. Johnson, D. Vlasic, W. Matusik, and H. Pfister, "Video face replacement," *ACM Trans. Graph.*, vol. 30, pp. 1–130, dec 2011.

[5] J. Thies, M. Zollhofer, M. Stamminger, C. Theobalt, and M. Nießner, "Face2face: Real-time face capture and reenactment of rgb videos," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 2387–2395, 2016.

[6] I. Goodfellow, "Nips 2016 tutorial: Generative adversarial networks," *arXiv preprint arXiv:1701.00160*, 2016.

[7] Y. Bengio, E. r. Thibodeau-Laufer, G. Alain, and J. Yosinski, "Deep generative stochastic networks trainable by backprop," in *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, p. II–226–II–234, JMLR.org, 2014.

[8] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[9] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[10] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 1125–1134, 2017.

[11] T. Karras, S. Laine, and T. Aila, “A style-based generator architecture for generative adversarial networks,” *CoRR*, vol. abs/1812.04948, 2018.

[12] R. Huang, S. Zhang, T. Li, and R. He, “Beyond face rotation: Global and local perception gan for photorealistic and identity preserving frontal view synthesis,” in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2439–2448, 2017.

[13] L. Tran, X. Yin, and X. Liu, “Representation learning by rotating your faces,” *CoRR*, vol. abs/1705.11136, 2017.

[14] E. Zakharov, A. Shysheya, E. Burkov, and V. S. Lempitsky, “Few-shot adversarial learning of realistic neural talking head models,” *CoRR*, vol. abs/1905.08233, 2019.

[15] Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, and L. Carin, “Variational autoencoder for deep learning of images, labels and captions,” in *Advances in neural information processing systems*, pp. 2352–2360, 2016.

[16] Y. Li, X. Yang, P. Sun, H. Qi, and S. Lyu, “Celeb-df: A large-scale challenging dataset for deepfake forensics,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3207–3216, 2020.

[17] E. Tjon, M. Moh, and T.-S. Moh, “Eff-ynet: A dual task network for deepfake detection and segmentation,” in *Accepted for 2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)*, 2021.

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

[19] H. H. Nguyen, F. Fang, J. Yamagishi, and I. Echizen, “Multi-task learning for detecting and segmenting manipulated facial images and videos,” *arXiv preprint arXiv:1906.06876*, 2019.

[20] T. Mittal, U. Bhattacharya, R. Chandra, A. Bera, and D. Manocha, “Emotions don’t lie: A deepfake detection method using audio-visual affective cues,” *arXiv preprint arXiv:2003.06711*, 2020.

[21] O. de Lima, S. Franklin, S. Basu, B. Karwoski, and A. George, “Deepfake detection using spatiotemporal convolutional networks,” *arXiv preprint arXiv:2006.14749*, 2020.

[22] D. Güera and E. J. Delp, “Deepfake video detection using recurrent neural networks,” in *2018 15th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, pp. 1–6, IEEE, 2018.

[23] Y. Li, M.-C. Chang, and S. Lyu, “In ictu oculi: Exposing ai generated fake face videos by detecting eye blinking,” *arXiv preprint arXiv:1806.02877*, 2018.

[24] D. Tran, H. Wang, L. Torresani, J. Ray, Y. LeCun, and M. Paluri, “A closer look at spatiotemporal convolutions for action recognition,” in *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pp. 6450–6459, 2018.

[25] K. Hara, H. Kataoka, and Y. Satoh, “Learning spatio-temporal features with 3d residual networks for action recognition,” in *Proceedings of the IEEE International Conference on Computer Vision Workshops*, pp. 3154–3160, 2017.

[26] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation,” in *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241, Springer, 2015.

[27] B. Baheti, S. Innani, S. Gajre, and S. Talbar, “Eff-unet: A novel architecture for semantic segmentation in unstructured environment,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 358–359, 2020.

[28] S. Mehta, E. Mercan, J. Bartlett, D. Weaver, J. G. Elmore, and L. Shapiro, “Y-net: joint segmentation and classification for diagnosis of breast biopsy images,” in *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pp. 893–901, Springer, 2018.

[29] M. Tan and Q. V. Le, “Efficientnet: Rethinking model scaling for convolutional neural networks,” *ArXiv*, vol. abs/1905.11946, 2019.

[30] J. Howard, “Dfdc - our solution for kaggle’s deep fake detection challenge.” GitHub Repository, 2020.

[31] S. Seferbekov, “Deepfake detection (dfdc) solution by @selimsef.” GitHub Repository, 2020.