Deepfake Detection and Classification

Sam Koenke
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
sskoenke@wpi.edu

Revant Mahajan
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
rmahajan@wpi.edu

Pratik Jawahar
Department of Robotics Engineering
Worcester Polytechnic Institute
Worcester, MA
pjawahar@wpi.edu

***Abstract*—Recent advancement in computer vision techniques have led to a surge in deep fake videos and images flooding. Deep fake is a term coined to represent fake images generated using deep neural networks. These videos and images can have a profound impact on world politics, stock market and personal life. With this paper, our team summarizes past work in this field and then focuses on creating a network architecture to classify these deep fakes.A CNN architecture is implemented and trained and many insights into the problem of deep fake classification are discussed. ¡Will talk about methodology in next revision¿**

***Index Terms*—Deepfakes, Deep Learning, Fake Videos, Deepfake Detection,**

I. Introduction

Social engineering is the planned manipulation of societal behavior, and has been around for as long as people have been able to communicate with each other. As the information age has progressed, it has become easier and easier to reach a larger audience. From the invention of writing, the wide spread use of printing, and now to a modern, digitally connected society; the ability to spread information has become easy and nearly instantaneous. With these changes in the rapid availability of information, one must now, more than ever, have a way to figure out what information is true and what is false.

Reliability theory states that the three most important aspects of information are its confidentiality, its availability and its integrity. Social engineering typically uses information that is released publicly (i.e non-confidential) and highly available through digital means such as the internet and social media. It is the difficult task of determining the integrity of the data that is often left up to the individuals who come in contact with the information.

With so many organizations trying to manipulate social perception and behavior, quite a bit of pressure is being put onto tech companies to minimize the amount of fake information that is being shared via their platforms. While this is certainly an aid to the minimizing the problem, the individuals who come in contact with the information still have a responsibility to determine what they will believe. Often individuals will base their believing of the integrity of information off of the source of the information and the people who provide it. For video and audio content, often this is done based off the visual or audible recognition of someone in the media, but what if you can no longer use your vision or hearing to validate the information? What basis do you now have to verify that piece of information is accurate? It is this vulnerability that deepfakes, a modern social engineering tool, seek to exploit.

Deepfakes, use deep learning to digitally create media content that appears to both look and sound like authentic media. The media created by these deepfakes are sometimes easily detectable, but sometimes good deepfakes are virtually indistinguishable from the real thing.

The potential effects of deepfakes are quite serious. Political manipulation has emerged as one of the most serious types of social engineering. A convincing deepfake video could do damage to political candidates or cause opposing countries to strike one another with military action. In a 2018 speech at the Heritage Foundation, Senator Ricky Rubio gave an indication of how serious of a threat deepfake technology is when he said, “In the old days, if you wanted to threaten the United States, you needed 10 aircraft carriers, and nuclear weapons, and long-range missiles. . . . Today, you just need access to our internet system, to our banking system, to our electrical grid and infrastructure, and increasingly, all you need is the ability to produce a very realistic fake video that could undermine our elections, that could throw our country into tremendous crisis internally and weaken us deeply.” [4]

Other uses of deepfake technology are almost endless. Fake evidence generated for use in the judicial system could lead to incorrect verdicts. Fake media containing non-consensual sexual content could be quite damaging to the personal life and career of individuals. Fake business information could be used to manipulate the stock market.

The problem of social engineering through the use of deepfakes is serious. This paper investigates methods of using deep learning to classify potential deepfake content as real or fake.

II. Background

The recent developments in computer vision and image processing algorithms have turned out to be a double-edged sword, bringing about remarkable augmentation in machine capabilities but concomitantly raising questions of morality as well. They have made significant contributions in a multitude of fields such as cancer detection, autonomous vehicles, augmented reality and state of the art surveillance systems. However, they have also become tools for generating fake digital media to a strikingly realistic degree which poses a

great threat on various levels. A new term has been coined to represent such morphed videos and images, Deepfakes.

A. What are Deepfakes

Deepfakes are artificially synthesized digital media, primarily in the form of images and videos, using Machine Learning and Artificial Intelligence. A deep fake may refer to a pre-recorded digital medium (video, image, audio, etc.) of a real person that is morphed either partially or completely, to synthesize an artificial subject that borrows features from the human subject, or a completely artificial digital medium, where every feature is autonomously computer generated with no reference medium. Though the artificial subject has realistic features, the probability of all its features matching those of a single real subject is very low, per the canonical definition. However, the artificial subject may be biased to imbibe features from one or more human subjects based on how the learning algorithm is trained.

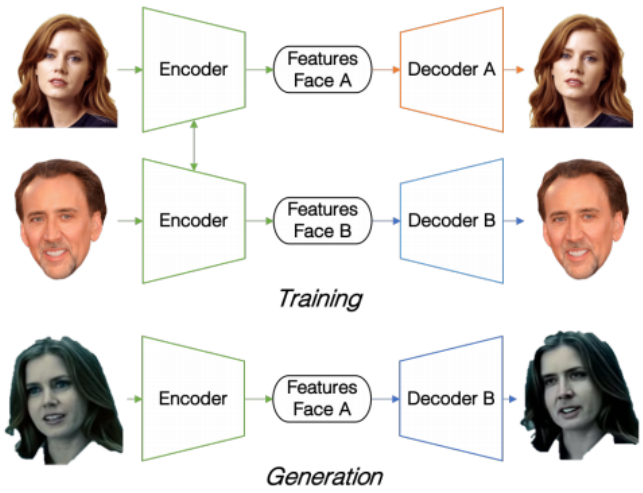

Fig. 1. Fake Images using Auto Enoders

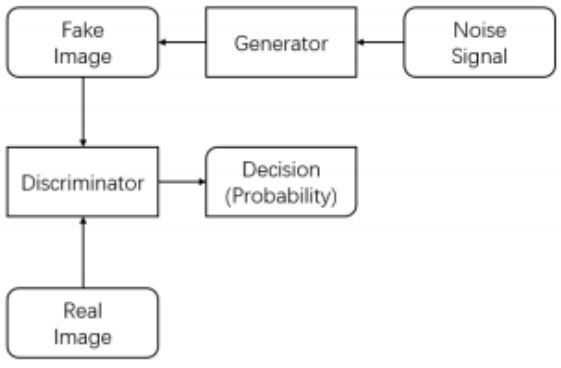

Fig. 2. Fake Images using GAN

B. How are Deepfakes generated

- Auto-encoders are known for dimensionality reduction and compact representations of images. They are able to form a compressed representation of images with a minimized loss function [5][8]. An Encoder Decoder network is used to generate fake images from two different real images. A common encoder is trained to form a compressed representation of the input images. For each image class, a different decoder is trained to get back the original image, or as close as possible from this compressed representation. After the training is done, an input image of one of the known classes is fed into the encoder and then to a decoder, of a different class, to generate the fake image.

- Generative Adversarial Networks are another method by which deep fakes can be created, through two neural networks competing against each other[12]. The first network is called a generator, and it tries to generate fake images. The second neural network is called a discriminator, which tries to classify if an image is fake or not. The feedback is fed back to the generator so it can be trained to produce images, that can fool the discriminator.

C. Detecting Deepfakes

Since the ethical implications of deepfake algorithms gained priority, multiple methods of detecting and classifying a given video as being legitimate or fake have come up. The initial approaches of detection that used weak and imperfect morphing features as indicator flags, have however been countered with improved and more realistic deepfake algorithms, making detection of deepfakes a daunting task. Some methods employed previously include the following methods.

- The use of 'softbiometric signatures' has been one of the most primitive techniques used to identify deepfakes. This method involves studying intrapersonal features of the original subject portrayed in the video and forming a class of implicitly distinct stochastic features such as head motion, facial gestures and characteristics while saying a particular phrase or while portraying a particular emotion. This data is used to train the learning algorithm and is then tested against videos of comedic impersonators of the subject and against intrinsically generated Deepfakes. A one-class SVM model is then used to distinguish the set of features from the real subject and those from an impersonator or a deepfake [1]. This method achieved 92% detection accuracy however, generating the softbiometric signatures is a highly complex and personalized process which means each subject must have his own training dataset. Generating such a training dataset is a very arduous task compared to generating a deepfake, meaning this method is quite obsolete with respect to the speed at which deepfakes are being developed.
- Semantic inconsistencies and weakly morphed features are another common and effective means of detecting

deepfakes. This technique exploits the occasional imperfections in image morphing that arise due to excessive movement of the subjects features, or due to swiftly changing image characteristics such as ambient lighting or momentary drops in pixel resolutions. Under these circumstances, the image morphing algorithm cannot find exact matches of facial features in order to morph the subjects features accurately, resulting in errors such as a double chin or a double layered jaw line for that given set of frames. Such frames may also be prone to semantic inconsistencies in the Neural Network that was used to create the deepfake, such as mismatched earrings [14], which are blatant indicators of a fake video.

- Intrapersonal feature analysis such as, analysis of eyes of the subject in a deepfake video has been the most successful and versatile method of detection thus far. This turns out to be an implicit short coming of GAN generated fake videos, where it was noticed that GAN generated video subjects rarely ever blinked [6]. It was also noticed that the eye movement did not accurately match the head pose for a GAN generated video. This is primarily because the GAN used to generate the video is trained using still images of the subject and as it goes with most portrait images of people, the eyes are rarely ever shut. Other intrapersonal features such as head pose, movement, color of the eyes etc. have been effective flags that are significantly difficult for GANs to forge, thereby serve as good indicators for detecting deepfakes [13] [7].
- Speech and audio analysis proves to be yet another effective tool, that strays away from any of the methods mentioned above. Studying audio quality and looking for semantic details in speech are the two main avenues for determining flags that might predict if an audio stream is real or forged. These two parameters however, are not fully incongruous when compared to how image features are analyzed to detect fakes. Looking for abrupt, unnatural changes in audio quality, ambient noise that doesn't match the setting of the video and semantic errors such as unnatural pauses, use of uncommon phrases and the sync between the video and audio feeds are being studied as potential indicators of deepfakes [2].

III. The Database

The data sets used for this project are sourced from the FaceForensics++ Database [cite:FF++], created and provided by Google JigSaw. This data set is created from a selection of 1000 videos containing 509, 914 sourced from www.YouTube.com to imitate real scenarios. These are then manipulated or forged by two computer graphics based methods (*Face2Face and FaceSwap*) and two Neural Network based (*DeepFakes and NeuralTextures*) approaches[11].

A. FaceSwap

FaceSwap is a graphics based approach to transfer the face region from a source video to the target. The face region is detected based on some sparse landmarks. These regions are then used to morph the source features with a 3D template model using shape blending techniques. This model is then fed back onto the target image and a correlation is formed to minimize the difference between the model and the extracted landmarks using the textures of the input image. This is performed for all source and target pairs for all frames in the video.[11]

B. DeepFakes

DeepFakes have become widely popular in recent times as a face morphing or swapping framework and are being studied extensively to develop deep learning algorithms. The implementation used in the FaceForensics++ data set is the *faceswap github*. In this method, two auto encoders and a shared encoder are used to reconstruct features of source and target frames respectively. The images are then cropped and aligned before blending the source features to the target image using Poisson image editing.[11]

C. Face2Face

This is a computer graphics based re-enactment system that projects the source expressions while maintaining features of the target image. This is done by first selecting keyframes manually. These frames are then used to generate a dense reconstruction of the face which enables easier simulation of the face under different illumination conditions or with changed expressions.[11]

D. NeuralTextures

This is a Neural Network based approach that makes use of GANs that rely on tracked geometry of the source and target frames. This information is extracted using *Face2Face*, before being fed into the Generative Network that performs the feature morphing from the source to the target.[11]

Fig. 3. Example from FaceSwap of real (left) and fake (right)

Fig. 4. Example from DeepFake of real (left) and fake (right)

IV. Methodology

Classifying DeepFakes requires a model that is robust enough to learn weights from two seemingly similar subsets, since the morphed frames are composed of human facial features just like the frames that are not morphed. At a cursory glance, it is evident that low level features are of little or no use which implies that our model needs to be deeper. In this paper we implement at two main approaches to perform the classification, Convolutional Neural Networks (CNN) and simple Deep Neural Networks (DNN). We design and train our models using GPU support to speed up the process.

A. Data Pre-processing

The primary step before building our model is to process the data in our preferred format, which can then be passed to the input layer of the model. For pre-processing, we start by extracting frames from the available videos in an RGB format, since it is the most common input type to train Neural Networks for image classification. All parsed frames are then resized to a 224×224 RGB format in order to ensure dimensional uniformity in the input layer. Also, due to the enormity of the data set and constraints on the computation power at our disposal, we decided to shunt the data set by picking one in every 60 frames parsed, and then discarding the rest. This helped in reducing the size of the data set to make the training process less computationally expensive. To avoid the problem of exploding gradients, we also normalised all the images when loading into our model.

B. CNN Model Architecture

The most common architecture for image processing, using CNNs, is the VGG16 architecture. There is an input layer, followed by pairs of Convolution layers and Pooling layers stacked together, depending on how deep the desired model is. The final convolution layer is followed by a flattening layer, which is in turn followed by a series of fully connected layers that plug out the output nodes.

The convolution layers perform kernel convolutions on the input image, to extract certain properties of the image. For e.g., a Gaussian kernel performs image smoothing on being convoluted with the input image, a Sobel filter is sensitive to edges in the image and so on. Since the image is in the RGB format, there are 3 kernels, one per channel and the size of each kernel used is 3×3.

After each of the convolution layers, a pooling layer is added to group together features learned in that layer. Max Pooling is used to take into account the strongest of the learned features and reject the rest. This output is then passed as an input to the next convolution layer and the number of such layers depends on the depth of the desired model.

Each convolution layer may be associated with a padding that defines how the kernel is convoluted with corner and edge pixels, and an activation function that deals with altering the outputs such that the net output encompasses important learned features. It dies this through deciding which neurons fire in each layer and thereby controlling which weights are updated in each layer.

The net output of all the convolution layers however, is not flat since the image is a multi-channel matrix. For this reason, the output of the convolution layers is flattened before passing it as an input to the fully connected layers. These layers perform in exactly the same manner as a simple Deep Neural Network (DNN).

Layer (type)	Output Shape	Param #
conv2d_15 (Conv2D)	(None, 64, 224, 224)	1792
max_pooling2d_7 (MaxPooling2	(None, 64, 112, 112)	0
dropout_7 (Dropout)	(None, 64, 112, 112)	0
conv2d_16 (Conv2D)	(None, 128, 112, 112)	73856
conv2d_17 (Conv2D)	(None, 128, 112, 112)	147584
max_pooling2d_8 (MaxPooling2	(None, 128, 56, 56)	0
dropout_8 (Dropout)	(None, 128, 56, 56)	0
conv2d_18 (Conv2D)	(None, 512, 56, 56)	590336
conv2d_19 (Conv2D)	(None, 512, 56, 56)	2359808
max_pooling2d_9 (MaxPooling2	(None, 512, 28, 28)	0
dropout_9 (Dropout)	(None, 512, 28, 28)	0
conv2d_20 (Conv2D)	(None, 1024, 28, 28)	4719616
conv2d_21 (Conv2D)	(None, 1024, 28, 28)	9438208
flatten_3 (Flatten)	(None, 802816)	0
dense_9 (Dense)	(None, 128)	102760576
dense_10 (Dense)	(None, 64)	8256
dense_11 (Dense)	(None, 32)	2080
dense_12 (Dense)	(None, 1)	33

Total params: 120,102,145
Trainable params: 120,102,145
Non-trainable params: 0

Fig. 5. CNN model architecture Iteration 1

Simple DNNs are composed of an input layer, followed by stacks of fully connected, hidden layers of neurons that finally feed out to the output layer. Each input node is passed through each node of the first hidden layer and the corresponding weights extract the input feature vectors. The output of each node of the first hidden layer is fed as an input to each node of the second hidden layer and this process is repeated for each layer.

The weights are tracked and a bias may be introduced to lead the net output to converge to a particular vector. The output nodes are then back-propagated, depending on the error function associated with the output vector of the first pass. This permits efficient updating of weights making the model more accurate for applications such as clustering or classification.

Layer (type)	Output Shape	Param #
conv2d_10 (Conv2D)	(None, 128, 112, 112)	3584
max_pooling2d_5 (MaxPooling2	(None, 128, 56, 56)	0
dropout_3 (Dropout)	(None, 128, 56, 56)	0
conv2d_11 (Conv2D)	(None, 256, 56, 56)	295168
conv2d_12 (Conv2D)	(None, 256, 56, 56)	590080
max_pooling2d_6 (MaxPooling2	(None, 256, 28, 28)	0
dropout_4 (Dropout)	(None, 256, 28, 28)	0
conv2d_13 (Conv2D)	(None, 512, 28, 28)	1180160
conv2d_14 (Conv2D)	(None, 512, 28, 28)	2359808
max_pooling2d_7 (MaxPooling2	(None, 512, 14, 14)	0
conv2d_15 (Conv2D)	(None, 1024, 12, 12)	4719616
conv2d_16 (Conv2D)	(None, 1024, 10, 10)	9438208
max_pooling2d_8 (MaxPooling2	(None, 1024, 5, 5)	0
conv2d_17 (Conv2D)	(None, 2048, 3, 3)	18876416
conv2d_18 (Conv2D)	(None, 4096, 1, 1)	75501568
flatten_2 (Flatten)	(None, 4096)	0
dense_5 (Dense)	(None, 512)	2097664
dense_6 (Dense)	(None, 64)	32832
dense_7 (Dense)	(None, 32)	2080
dense_8 (Dense)	(None, 1)	33

Total params: 115,097,217
Trainable params: 115,097,217
Non-trainable params: 0

Fig. 6. CNN model architecture Iteration 2

When an optimal model has been trained, it can be tested with a different set of inputs and the accuracy of classification is generally commensurate to the training accuracy, since back-propagation is a strong weight optimization algorithm.

The major drawback with using simple DNNs however, is that training large data sets drastically affects the computation cost, since each input node is passed through all hidden layer nodes in the fully connected setup. This is where CNN has a comparitive advantage as it shares its weights, allowing it be much deeper, with the same number of tune-able parameters. It becomes evident that CNN stacks followed by fully connected layers perform better.

This CNN based model architecture is robust to learning image features and the depth of the model ensures feature maps with higher level features learned. These higher level features are crucial when it comes to classifying images with features that are virtually indiscernible on a cursory glance. The lower level features help identify basic discrepancies that are equally important for classification.

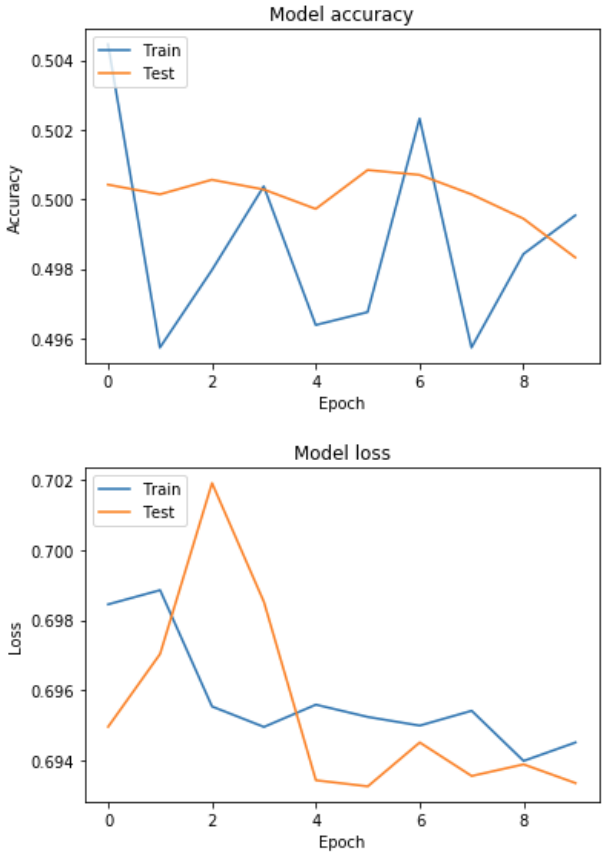

Fig. 7. CNN model with batch size of 32 and 10 epochs on DeepFake

C. Training on Amazon Web Services

To train and prototype different models, we decided to setup two different AWS EC2 instances with Nvidia Tesla K80 12 GB GPUs. One was used to run the CNN prototypes on DeepFake images and the other was used to run FaceSwap images. ACE cluster was used to quickly prototype these models and then they were deployed on AWS to train.

V. Results

For our CNN model, we took a similar approach to the VGG network. Unfortunately due to the limitation of GPU memory, we could not make it as deep as the VGG. The model architecture used is shown in figure 5

The results of this model with 224×224 sized images, batch size of 32 and runs of 10 and 30 epochs gave an accuracy of about 50% as shown in figure 7 and figure 8 respectively. Due to lack of more GPU memory, the deeper models could not be trained with this configuration.

To train it for more epochs and with more filters, the inputs were resized to 112×112, more convolution layers were added and the model was trained with 50 epochs to observe how it performed. This architecture is shown in figure 6 This model was trained on two different subset of the manipulated data, DeepFake and FaceSwap to compare the accuracy of the same

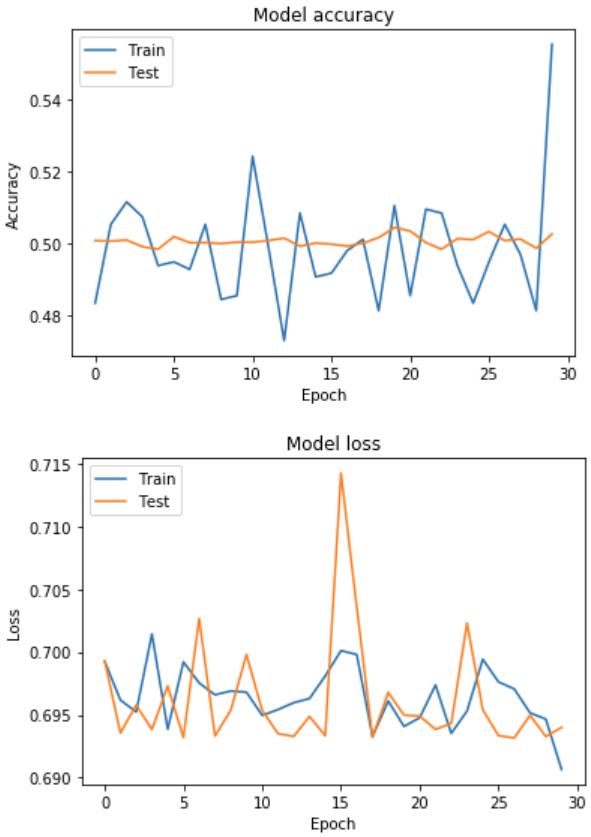

Fig. 8. CNN model with batch size of 32 and 30 epochs on DeepFake

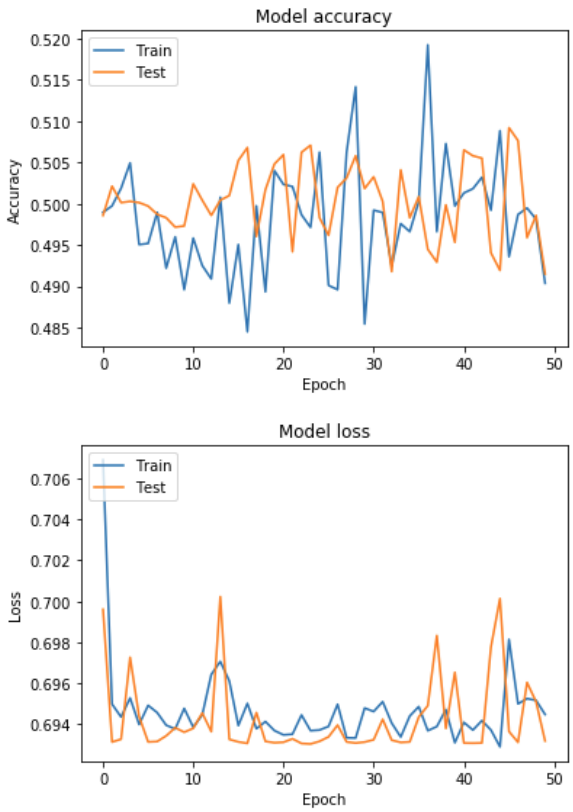

Fig. 9. CNN model with batch size of 128 and 50 epochs on DeepFake

model on both these DeepFake types. The model performed much better on FaceSwap getting about 55% as compared to 50% on DeepFake as seen in figure 10 and figure 9 respectively .

VI. Analysis

Detecting DeepFakes and distinguishing them from pristine images involves studying the semantic inconsistencies of the algorithm that generated the morphed output. The input to our classifier is available as time series data and since the CNN model is not capable of registering time series semantics, the classification relies purely on individual frame semantics. This is one major drawback of the CNN model and one of the main reasons that drives the accuracy down, since time series inconsistencies are some of the strongest outliers with respect to DeepFakes.

The semantic inconsistencies on a frame-to-frame basis however are modelled well by our model. The initial convolutional layers are robust at extracting low level features that may correspond to image features such as edges, boundaries, sharpness details, illumination etc. These features are valuable in cases where the morphing is imperfect in terms of inaccurately modelling facial landmarks. The most common example of this imperfect morphing is a multilayered jawline or facial boundary. This is the reason for incorporating a smaller number of neurons in the initial layers and it also makes extracting the strongest learned feature using MaxPooling much simpler.

The mid-level features are rather unimportant to us since these usually include subject semantics. In our case, the mid-level features of both pristine and morphed images are somewhat the same. Mid-level features in general correspond to common subject features like components of the face such as eyes, nose etc. These are present in both morphed and pristine images with approximately the same probability and feature space definitions, rendering such features useless to our classifier.

The higher level features that are learned more significantly by the deeper layers tend to be essential since these contain crucial semantic information. These features would correspond to real world semantics, such as the average distance between two eyes, presence of two eyes above a single central nose and average proportionality in position and orientation of mid-level features. These are crucial for detecting DeepFakes since the biggest problem with morphing images comes with modelling and aligning the source features to the target image. This leads to huge semantic inconsistencies in position and orientation of crucial subject features.

Despite our model being robust to picking up such crucial features, some of the main reasons for poor performance of our model are, small size of the dataset owing to hardware constraints, frequently changing subjects leading to rapidly

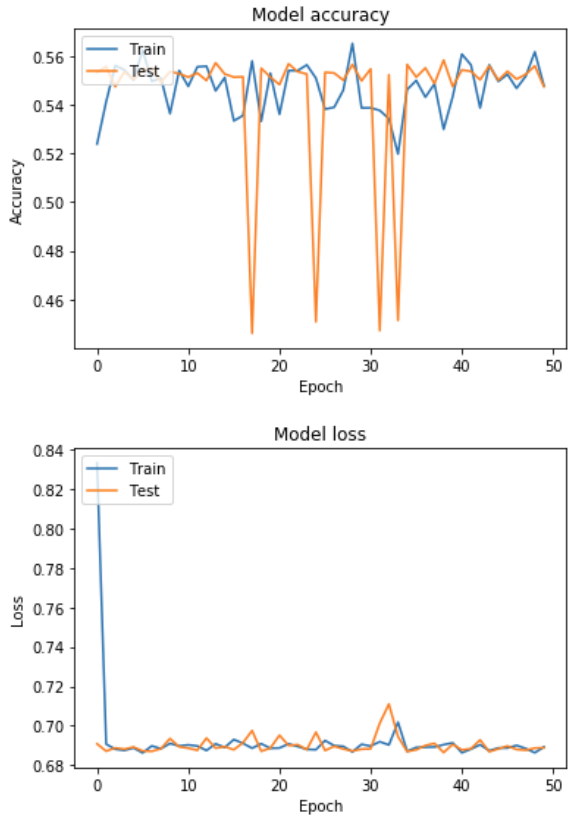

Fig. 10. CNN model with batch size of 128 and 50 epochs on FaceSwap

changing features in pristine images causing a lack of commonality in crucial features at the MaxPooling layer, and changing backgrounds from video to video, causing a further decrease in the commonality in learned weights. Also, the fact that the background remained the same in both morphed and pristine videos may have dominated over low level semantic discrepancies causing significant misclassification.

VII. Conclusion and Future Work

As indicated in the results and analysis sections, CNNs and DNNs do not seem to well represent the feature differences across a temporal dimension. Since many inconsistencies show up across different frames of a deep fake video, treating frames as a sequence will likely provide more insight into the differences between subsequent frames. The deep neural network architecture seemingly suited to treating time series data is the Long-Short Term Memory (LSTM) network.

The changing background is also a big factor that drives the classification accuracy, so creating a bounding box around the subject and passing only the contents of the bounding box to the model should improve the detection accuracy significantly.

The proposed architecture to work with combines the CNN, LSTM and DNN networks to make a general deep fake classifier that can trace the discrepancies between frames in a DeepFake video. An example of such an architecture is shown in 11. This combined with hardware capability to process larger datasets should be more robust to classifying and detecting DeepFakes.

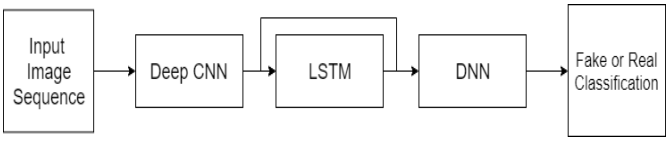

Fig. 11. A LSTM network with CNN input layers and DNN output layers

The deep CNN is for low-level feature extraction from an image. After the image features are processed they will be input to the LSTM to be have sequence descriptors computed. The sequence descriptors will be flattened and passed to the DNN for the classification task to be performed.

This architecture has been built but errors dealing with shape mismatches kept popping up during training. Completing the training and evaluation of this network will be left for future work.

References

[1] Shruti Agarwal, Hany Farid, Yuming Gu, Mingming He, Koki Nagano, and Hao Li. Protecting world leaders against deep fakes. page 8.

[2] Ehab A AlBadawy, Siwei Lyu, and Hany Farid. Detecting AI-synthesized speech using bispectral analysis. page 6.

[3] Ronit Chawla. Deepfakes : How a pervert shook the world. page 5.

[4] 2018 / 9 Comments. 'deep fake' technology is a threat to national security, politics, and the media, marco rubio says.

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

[6] Yuezun Li, Ming-Ching Chang, and Siwei Lyu. In ictu oculi: Exposing AI created fake videos by detecting eye blinking. In *2018 IEEE International Workshop on Information Forensics and Security (WIFS)*, pages 1–7. IEEE.

[7] Yuezun Li and Siwei Lyu. Exposing DeepFake videos by detecting face warping artifacts. page 7.

[8] Thanh Thi Nguyen, Cuong M. Nguyen, Dung Tien Nguyen, Duc Thanh Nguyen, and Saeid Nahavandi. Deep learning for deepfakes creation and detection.

[9] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. Deep face recognition. In *BMVC*.

[10] Andreas Rossler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics: A large-scale video dataset for forgery detection in human faces. page 21.

[11] Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, and Matthias Nießner. FaceForensics++: Learning to detect manipulated facial images.

[12] Tianxiang Shen, Ruixian Liu, Ju Bai, and Zheng Li. "deep fakes" using generative adversarial networks (GAN). page 9.

[13] Turek. Media forensics, https://www.darpa.mil/program/media-forensics.

[14] Turek. Semantic forensics, https://www.darpa.mil/program/semantic-forensics.