Deepfake Detection through Deep Learning

Deng Pan, Lixian Sun, Rui Wang, Xingjian Zhang, Richard O. Sinnott
School of Computing and Information Systems
The University of Melbourne, Melbourne, Australia
Contact: rsinnott@unimelb.edu.au

Abstract **- Deepfakes allow for the automatic generation and creation of (fake) video content, e.g. through generative adversarial networks. Deepfake technology is a controversial technology with many wide reaching issues impacting society, e.g. election biasing. Much research has been devoted to developing detection methods to reduce the potential negative impact of deepfakes. Application of neural networks and deep learning is one approach. In this paper, we consider the deepfake detection technologies Xception and MobileNet as two approaches for classification tasks to automatically detect deepfake videos. We utilise training and evaluation datasets from FaceForensics++ comprising four datasets generated using four different and popular deepfake technologies. The results show high accuracy over all datasets with an accuracy varying between 91-98% depending on the deepfake technologies applied. We also developed a voting mechanism that can detect fake videos using the aggregation of all four methods instead of only one.**

Keywords - *DeepFake Detection, Xception, MobileNet, FaceForenscis++, Keras, TensorFlow.*

1 Introduction

With the 2020 US election around the corner, the media has shown a great deal of concern related to deepfake videos. In the era of fake news, people and society more generally are concerned that they can no longer believe what they see online. In this context, Facebook and Instagram announced a new policy in January 2020 banning AI-manipulated "deepfake" videos that are likely to mislead viewers to prepare for the election [1]. The challenge is that this depends on the capability to distinguish between real and fake videos. This is the focus of this paper.

Deepfakes are synthetic media in which a person in an existing image or video is replaced with someone else's likeness [2]. Figures 1 and 2 show screenshots of some famous deepfake works including Barack Obama calling Trump a "*dipshit*" and Nicolas Cage's face being swapped to roles in movies in which he did not appear. These videos drew millions of views on Youtube as people were amazed and concerned about how real they appeared.

The fast evolution of deepfakes has made both the academic field and technology industry put considerable focus on automated detection of deepfake videos, since more and more people are using deepfakes to generate maany forms of fake information from fakes news to hoaxing of content, e.g. celebrity pornography.

Figure 1. Deepfake Video of Barack Obama
https://www.youtube.com/watch?v=cQ54GDm1eL0

Figure 2. Deepfake Video of Nicholas Cage
https://www.youtube.com/watch?v=BU9YAHigNx8

In this paper we consider different deep learning solutions to automatically classify and hence detect deep fake videos. Specifically, we utilise FaceForensics++ [3] as the source video data and used this data to train two neural networks: Xception [4] and MobileNet [5] using pre-processed images. The training of each network produces four models, each corresponding to one of four different mainstream deepfake software platforms. These include Deepfakes [6], Face2Face [7], FaceSwap [8] and NeuralTextures [9]. The result of the model's evaluations demonstrates a high degree of accuracy in distinguishing real and fake videos however this accuracy is also highly sensitive and depends greatly on the deep fake

platform used. To address this, we present a voting mechanism that leverages the outputs of the various models to provide a more robust solution.

2 Related Works

The term deepfakes was coined from the merger of "deep learning" and "fake", referring to the use of state-of-the-art computer vision methods and deep learning techniques to generate fake videos. Fake videos generated using deepfakes consist of two main categories: *face-swapping* and *face-reenactment*.

Face-swapping involves the automatic replacement of a face in a video or image with another face where the identity of the person in the video changes. This original face swapping method can be dated back to a Reddit user post in 2017 [10]. The method used two encoder-decoder pairs, while the two encoders share parameters during training, the training of the decoders are separated. Specifically, for face scenarios, the original face A enters the encoder, then the decoder of B is connected, and the decoding can be done by replacing the face of B with the face of A. This method required a complex training process and consumed a lot of computing resources. This method only worked if there was a large number of target images and video footage to use as training data. This method has however been widely used as it provided the first deepfake method.

Faceswap-GAN is a popular face swap method [8]. Based on the original-deepfakes method, Faceswap-GAN adds antagonistic and perceptual loss to the result of the automatic coding system. Adding counter losses improves the reconstruction quality of the generated image. The addition of perceptual loss improves eye orientation and aligns the face of the generated image with the input image. This method is an optimized version of the original-deepfakes approach [10].

Inspired by image style transferral, Fast Face-swap [11] takes A's face gesture and expression as content and B's identity as style. It then learns B's style while maintaining A's content during the fake image generation. The method used a single image as input. It firstly implements face-alignment and face-segmentation to the image and then uses a transforming network to generate fake images. The conversion network is an image pyramid structure with different branch operations, while each branch contains a zero-padded convolution. The loss function is defined in the feature space of the pre-trained (VGG19) network and contains content loss, style loss and light loss. It also defines total variation regularization to ensure spatial smoothing.

Face-reenactment is the transferral of the expression and pose of a source character to a character in the target video, while the identity of the target character remains the same. Face2Face [7] implements real-time facial reconstruction, which aims to bring the facial expressions of the target video into motion with a source actor and re-render the output video in a realistic manner. Using *Dlib* and *OpenCV*, it first detects the face in the source image with the face detector, finds the key marker points on the face, then fits two 3D models and converts the key marker points into the target face image using the face-specific *pix2pix* conversion model.

Neural Textures [9] proposes a pseudo-video generation method based on neural textures. The method uses raw video data to learn the texture of the target and incorporates information on photometric reconstruction loss versus antagonistic loss. However, the method only focuses on modifying the facial expression corresponding to the mouth region, leaving the eye region largely unchanged.

There have been numerous works considering deepfake video detection methods that consider different aspects. For instance, the blinking rate of a human being is about once every two to ten seconds and the time for each blink is about half or a quarter of a second. People in deepfake videos rarely blink, making deepfake videos slightly more distinguishable from real videos. This was considered by Chawla et al [6].

Apart from blinking frequency, the colour of eyes can also be used for detecting manipulated media. Matern et al. [12] identified that during the generation of fake images, there can be a lack of global consistency exhibited in the high variance of hair and eye colour. They proposed a method for extracting facial regions from an image and detecting iris pixels to determine eye colour features. In this way, deepfake videos or images could be distinguished from their authentic counterparts. However, details like this can always be made up by intentionally collecting the blinking images of the target person and using them for training, which does not guarantee the identification of fake videos.

Apart from the manipulated content itself, other parameters created as by-products of the synthetic process can be used for deepfake detection. For example, swapping a face into another image can cause a discontinuity in pixels and lighting that cannot be sensed by humans. However, such a data-trace, can be used in multimedia forensics to examine whether the video or image has been altered. Such an approach was considered by Gardiner et al [13].

Compared to manual detection done by humans, Convolutional Neural Networks (CNNs) can detect deepfake content via image analysis features [14]. Neural networks allow computers to learn from features that can be barely perceptible to human eyes. Do et al [15] implemented a CNN with a fine-tuning method and achieved an accuracy above 70% using three different image datasets.

3. Dataset

For any deep learing application, the use of a large, feature-rich and highquality data set is essential to amongst other things, ensure that overfitting is avoided, i.e. the application only works on the trained data set. In this work, FaceForensics++ [3] was chosen as the source dataset. There were several reasons for this. Firstly, it is publicly available. It provides detailed documentation for the data and use of

the data. Secondly, almost all extracted images contain a single unobstructed frontal face that can be easily tracked. This typifies many deep fake scenarios as highlighted in Figure 1 and Figure 2, so this was not a limitation. Thirdly, it provides videos that are generated from a comprehensive range of popular deepfake video generation methods. That is, the final deep fake detection solution should work with the outputs of many diverse deep fake generation technologies and not just a single source. Fourthly, FaceForensics++ provides different video quality options for downloading of data depending on time and bandwidth constraints. The varying qualities are important for deep fake detection, i.e. it is easier to distinguish a high-resolution deep fake than a lower resolution fake video.

FaceForensics++ provides a dataset consisting of 1000 original video sequences that have been manipulated based on four automated using different face manipulation methods, namely: Deepfakes, Face2Face, FaceSwap and NeuralTextures. These 5000 videos were downloaded to the University of Melbourne High Performance Computing System (SPARTAN) [24]. All h264 videos were downloaded using a 23x compression rate for time and storage efficiencies. There were five sets of videos in total as shown in Table 1: the original videos, and the deep fake versions of these videos created using Deepfakes, Face2Face, FaceSwap and NeuralTextures.

DataSet	Video Count
Original	1000
Deepfakes	1000
Face2Face	1000
FaceSwap	1000
NeuralTextures	1000

Table 1. Video Count for Each Dataset

For each data set, the data was split into two parts, 80% for training the deep learning models and 20% for subsequent testing and validation, i.e. the testing and validataion data were not used for training the models.

4. Methodology

The ultimate goal of this work was to identify whether a video was real, or whether it had been generated using deepfake technology. As such, the input to the system clearly has to be a video. However, deep learning models use pictures as input, hence conversion of the system (video) input to the model input needs to be done. This is done through a pre-processing module as shown in Figure 3.

In addition to the transformation of input data types, the pre-processing module also needs to take into account the impact that other factors in the video may have on model training. Thus, each frame in the video does not contain just a face. Indeed, the body parts of the person and the background area of the image comprise most of the video frame. These irrelevant features can negatively impact the training of the model. The face area in the image is the focus, and the pre-processing module needs to capture the face in the image as model input. The pre-processing module itself consists of three steps as shown in Figure 4: intercepting frames from the video, detecting faces from these individual frames, and saving face areas as images. Each of these steps is discussed below.

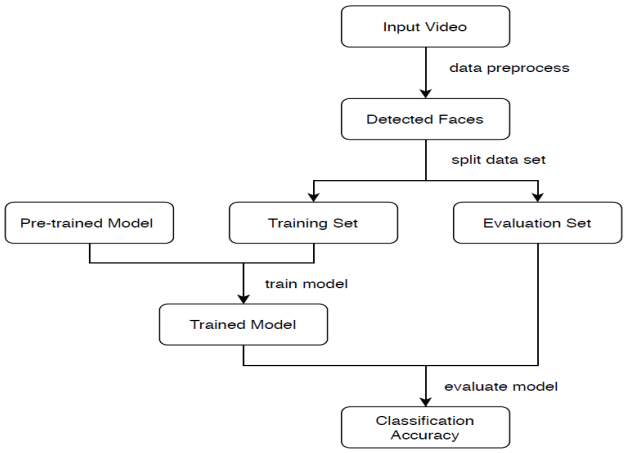

Figure 3. Overall Process Flowchart

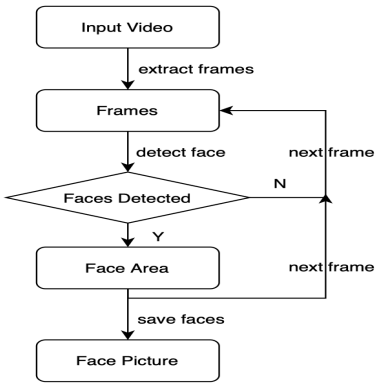

Figure 4. Pre-processing Flow Chart

The first step was to capture the input video into frames. The *video capture* function provided by the OpenCV Python package was used for this purpose. Since the method explored in this project was based on using a single image as input, inter-frame information was not needed. Besides, the similarity between two adjacent frames is too high hence putting them all into a training set will not only reduce the training efficiency but also potentially give rise to issues such as overfitting. The videos selected for this work were all at a frame rate of 30 frames per second. After testing it was identified that selection of one image from every four frames provided a reasonable approach.

The second step was to detect the faces that appear in the image and label them. The *cascade classifier* provided by *OpenCV* was used for this purpose. After testing a variety of classifiers, the *haarcascade_frontalface_alt* classifier was

chosen as it creates the most accurate area of the face. There were some non-face selection issues. For example, after testing, it was found that the area of the misjudged human face was generally smaller than the area of the correct face area, so only the largest area was retained so that the correct face area could be detected.

The third step was to save the detected area of the face as a new image. Before storing, the facial images needed to be resized uniformly. The image size required for the Xception model is 299*299, while the image size for the MobileNet model is smaller at 224*224. Examples of the facial images from the extracted frames are shown in Figure 5.

Figure 5. Extracted Frames (Left) & Processed Images (Right)

4.1 Deep Learning Models

There are many deep learning models and frameworks that are now available. Xception and MobileNet were chosen as models for the experiments in this paper for the following key reasons. Firstly, Xception has a high performance based on its benchmarking performance on the FaceForensics test environment. FaceForensics provides a test environment for researchers to test their trained models. The performance of models trained by different teams are displayed in Figure 6 along with the methods they chose. Among these methods, Xception shown a relatively good performance over four different datasets and more importantly, it is open source with extensive documentation, making it easier for model training and tuning.

MobileNet was chosen as it shares a similar structure to Xception. They are both based on convolutional neural networks (CNNs) and both employ depthwise and pointwise convolutional layers. The difference between them is that MobileNets has fewer features to increase the efficiency of the model.

FaceForensics Benchmark

This table lists the benchmark results for the Binary Classification scenario.

Method	Info	Deepfakes	Face2Face	FaceSwap	NeuralTextures	Pristine	Total
Two-stream-SRM-RGB1		**0.982**	**0.927**	0.942	**0.953**	0.174	0.562
eff-b7-v3		0.973	0.912	0.913	0.807	0.198	0.546
Sentinel		0.964	0.905	0.883	0.867	0.624	0.763
face single model		0.964	0.869	0.942	0.460	0.738	0.760
LGSC_Lite		0.964	0.839	0.922	0.660	0.812	0.821
Xception	[P]	0.964	0.869	0.903	0.807	0.524	0.710
Andreas Rössler, Davide Cozzolino, Luisa Verdoliva, Christian Riess, Justus Thies, Matthias Nießner: FaceForensics++: Learning to Detect Manipulated Facial Images. ICCV 2019							
simple policy	[B]	0.955	0.869	0.942	0.387	0.744	0.751
eff-b7-att epc11		0.955	0.788	0.864	0.547	0.592	0.680
EfficientNet-b4		0.955	0.796	0.825	0.827	0.712	0.779
ATDETECTOR		0.955	0.781	0.903	0.760	0.808	**0.823**
fakeface		0.955	0.766	0.883	0.747	0.806	0.816
RealFace		0.945	0.766	0.864	0.813	0.790	0.815
faceClassify1		0.945	0.854	**0.981**	0.793	0.730	0.806

Figure 6. FaceForenscis Benchmark

4.1.1 Xception

Xception, stands for Extreme Inception, was presented by Google researchers in 2017 [4]. It provides a deep convolutional neural network architecture that involves Depthwise Separable Convolutions [16]. The Depthwise Separable Convolutions deal with both spatial (image width and height) and depth dimensions. The depth dimension focuses on the number of channels. An image typically has three channels: red, green and blue, although the number of channels may increase after a few convolutions [17].

In the original Depthwise Separable Convolution (i.e. the Inception Module), the Depthwise Convolution comes before the Pointwise Convolution. Depthwise convolution is the channel-wise n*n spatial convolution while Pointwise convolution is 1*1 convolution to change the dimension [18]. Both operations are followed by a Rectified Linear Unit (ReLU) non-linearity.

Xception applies a modified version of Depthwise Separable Convolution with a few key differences. The first difference is the reverse order, i.e. Pointwise Convolution comes before Depthwise Convolution. A second difference is that Xception does not use non-linearities in its implementation.

The researchers of Xception replaced Inception modules with modified depthwise separable convolutions. The hypothesis was that the mapping of cross channels correlations and spatial correlations in the feature maps of convolutional neural networks could be entirely decoupled [4]. Xception has a feature extraction base consists of 36 convolutional layers, which are structured into 14 modules. These are optionally followed by fully connected layers. The last layer is a logistic regression layer that is used for image classification.

The Keras framework [19] has made using Xception straightforward by wrapping the Xception model as a high-level package with pre-trained weights. It is lightweight, stable and time-efficient to implement. This means that

detailed model implementation is not required and instead the focus can be on valid feature extraction and detecting deepfake videos.

4.1.2 MobileNets

As the name suggests, MobileNets are efficient neural network architectures, which are suited for mobile and embedded vision-based applications that lack computational power. This architecture is based on a depthwise separable convolution. Depthwise separable convolution is a form of convolution that decomposes standard convolutions into 1*1 pointwise convolutions [5].

Standard convolutions filter the inputs and merges them into a new set of outputs. Standard convolution operations filter functions based on the convolution kernel and combine these functions to create a new representation.

The depthwise separable convolution is divided into two layers: separate layers for filtering and separate layers for merging. Deep and separable convolution consists of two layers: depthwise convolution and pointwise convolution. Depthwise convolution is used to apply a single filter to each input channel (input depth). Pointwise convolution then uses a simple 1*1 convolution to create a linear combination of depth layer output.

This factorization has the effect of significantly reducing the calculation and model size. Specifically, in the MobileNets model, each input channel has a filter. After the filtering process, the outputs of 1*1 pointwise convolution and depthwise convolution are combined.

MobileNets has 28 layers in total. A comparison between standard convolutional layer and depthwise convolutional layer is shown in Figure 7.

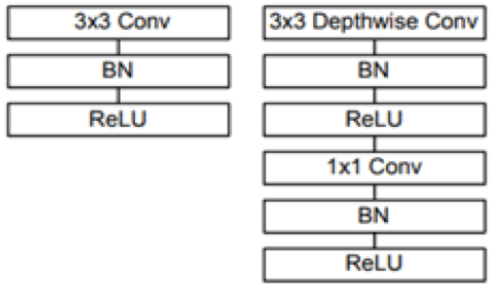

Figure 7. Comparison of Standard Convolutional Layers and Depthwise/Pointwise Convolutional Layer with Batchnorm (BN) and ReLU [5]

The model structure of MobileNets places all the computations into dense 1*1 convolutions using highly optimized general matrix multiply (GEMM) functions, so that most of the additional parameters end up in the fully connected layer.

The MobileNets model is trained in Tensorflow using asynchronous gradient descent. This is similar to Inception V3 [20]. However, unlike InceptionV3, which trains large models with regularization and data augmentation techniques, MobileNets intentionally does not tackle images with sides of heads for example. It also reduces image distortion by limiting the size of images that are allowed because smaller models are less likely to have problems with overfitting. Other training parameters remain the same irrespective of the size of the model though.

4.2 Virtual Environment Configuration

The University of Melbourne SPARTAN cluster was chosen as the platform to perform all experimental tasks including video downloading, image preprocessing, model training and model evaluation. SPARTAN is a high-performance computing (HPC) system operated by Research Computing Services at The University of Melbourne. It combines a high-performance bare-metal compute cluster with flexible cloud infrastructure and associated GPU cluster to suit a wide range of use-cases.

As with many HPC systems, SPARTAN does not allow users to install software themselves, instead, it has a pre-installed set of modules that are configured by the system administrators. Since there were some required packages not available on Spartan, a targeted virtual environment was created. The virtual environment was activated in each slurm script submitted by the user.

GPUs are essential for machine learning training due to their efficiency. Therefore, the configuration of a compatible set of TensorFlow, CUDA and cuDNN was crucial to prepare for the training.

4.3 Training

Since pictures from the same video can have a degree of similarity, it is preferential that pictures extracted from the same video are in the same data set. If the model has been trained using pictures from one video, it will have a larger chance to correctly predict other pictures from that video. However, the objective is to identify common features shared between different videos. Thus, datasets were split by the video ID instead of by the picture ID. The length of videos also differed. As a result, the training data set and test test data set had a different ratio of pictures.

4.3.1 Data Splitting

When pre-processing videos, the path of output files was in the form '*pictures/fake/Deepfakes/00132-00121.png*' where 00121 was the picture ID. This implies that this picture was the 121st picture from video 00132. 00132 is the video ID and this implies that the video was the 132nd video in the folder '*Deepfakes*'. All video IDs were put into a list. As noted previously, this list was split into two parts with 80% used for training and 20% used for testing.

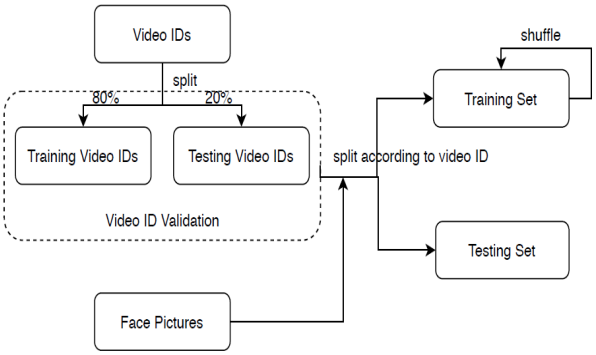

Figure 8. Flowchart for Data Preprocessing

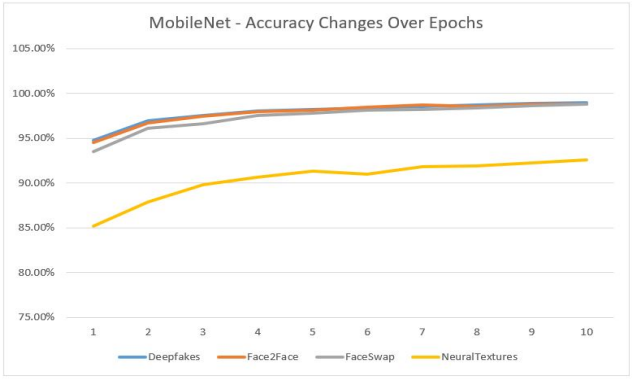

Figure 9. MobileNet - Accuracy Change Over Epochs

The picture paths were put into a list of training set paths based on the video ID of the picture path. For instance, if 00132 is a video ID for training, all paths of pictures from this video were put into the training paths list. Before training, the paths list used for training was shuffled. To save memory, a generator was created to yield samples instead of reading all pictures into memory. Each image was converted into a 299*299*3 numpy array when generating samples. Labels were used for fake pictures (1) and for non-fake pictures (0).

4.4 Training Configuration

The adaptive moment estimation (*adam*) was employed as the optimizer [21]. The learning rate needs to be dynamically adjusted, which makes Adam a suitable optimizer. Other parameter settings used include: a loss function based on cross entropy and the batch size.

Sambhu et al. [22] identified that selection of the batch size is often a trade-off between training efficiency and model accuracy. Although a larger batch size can reduce the time needed for training, the accuracy of the model will often decrease. In order to achieve higher accuracy, a batch size of 32 was chosen for the implementation. The number of iterations (epochs) which was set to 10.

From our experiments on the effect of epochs on the accuracy of the model, as shown in Figure 9, it was observed that the model's accuracy increased gradually from epoch 1 to epoch 8, then eventually converged approaching eopch 10. In addition, the multi-processing option was set to True.

5. Results

For each dataset related to the deepfake video generating approaches, i.e. Deepfakes, Face2Face, FaceSwap and NeuralTextures, 20% of the videos from the dataset were tagged and reserved for the final evaluation. Additionally, the model trained with the dataset Deepfakes was evaluated with the other three datasets to determine if the detection performance of the model had any overfitting problems with the fake video generation method.

Instead of focusing on just the accuracy of the model as a binary classification, the evaluation was based on a true positive rate (TPR) and true negative rate (TNR) of the classification result. The real video and fake video detection accuracy and the overall detection accuracy were computed based on the following equations.

$$Real\ Video\ Detection\ Accuracy = \frac{TN}{TN + FN} \times 100\%$$

$$Fake\ Video\ Detection\ Accuracy = \frac{TP}{TP + FP} \times 100\%$$

$$Overall\ Detection\ Accuracy = \frac{TN + TP}{TP + TN + FP + FN} \times 100\%$$

The results of the experiments are shown in Figures 10-14. The following observations from these figures can be made. Generally speaking, the results show a high degree of accuracy of classification for all measurements across all experiments including true positive rates (TPR), true negative rates (TNR) and the overall accuracy in identifying deep fakes or classifying videos as real/authentic. All rates were above 85% and some rates could achieve over 98% on all datasets (with the exception of NeuralTextures).

In the Xception model, the TNR was approximately 1% higher than the TPR in almost all datasets. This difference was larger in the MobileNet model. With regards to NeuralTextures, the TPR was 91.57% and the TNR 85.97%, giving a difference in prediction accuracy of 5.6%.

Combining the results obtained from both the Xception and MobileNet models, we found that both methods had relatively low detection rates on fake video datasets

generated by the NeuralTexture method with the detection accuracy generally 6-8% lower than those of the other datasets. This is due to the fact that fake videos generated through the NeuralTexture method were much more realistic and could often not be distinguished by manual comparison.

Comparing the results obtained from the Xception and MobileNet models, it can be observed that MobileNet had a higher detection rate for the FaceSwap-based deepfake videos compared to Xception (by approx. 1%). For the other three datasets, MobileNet's detection capabilities had a drop of 2-3% in detection accuracy compared to the Xception model, however the benefits gained from use of a much smaller model may outweigh these differences, e.g. if the end application is to be a mobile application for example.

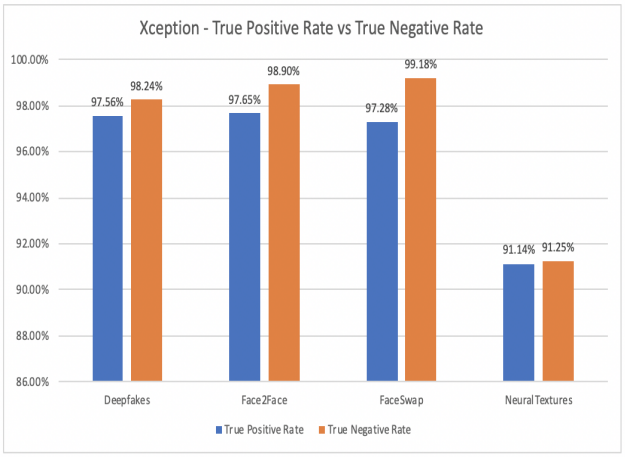

Figure 10. Comparison of TPR vs TNR using Xception

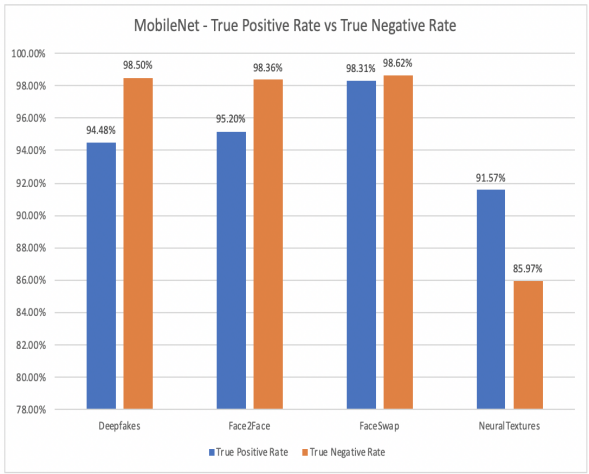

Figure 11. Comparison of TPR vs TNR using MobileNet

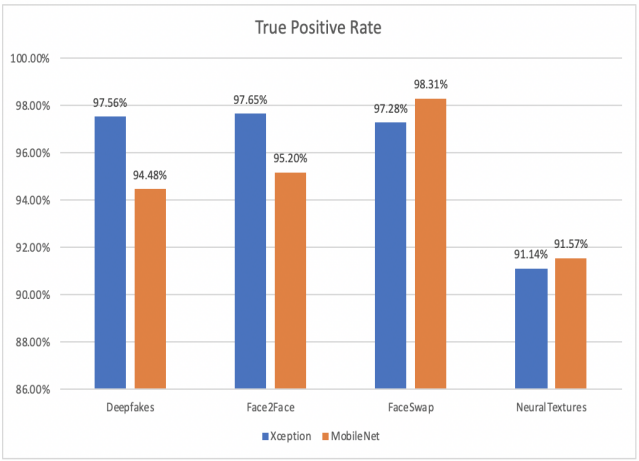

Figure 12. TPR Comparison between Xception and MobileNet

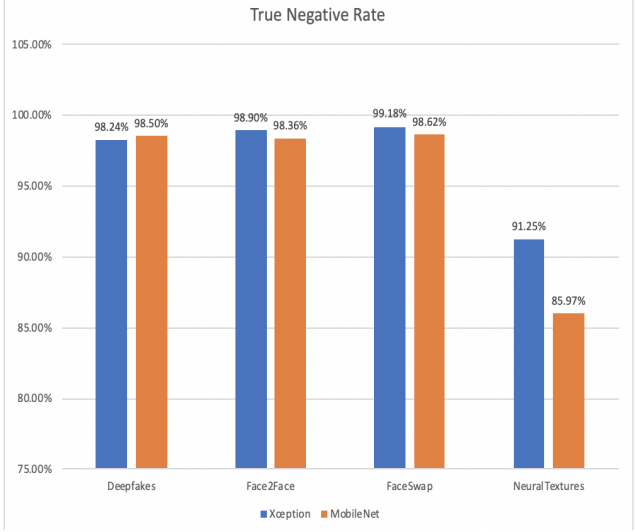

Figure 13. TNR Comparison between Xception and MobileNet

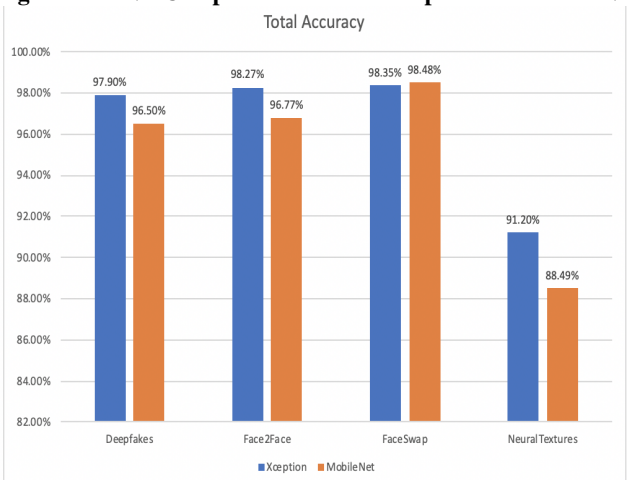

Figure 14. Total Accuracy Comparison between Xception and MobileNet

In addition to the Face Forensics++ video data sets, other deep fake videos were retrieved from youtube and tested

using a combined voting mechanism. Firstly, a manipulated video of former president of the US, Mr Obama, was used to test the functionality of the models (see Figure 1). The video was successfully classified as fake. According to the results presented in Table 2, the video was classified as fake by NeuralTextures while other models classified the video as real ('raw' in the table). Since the voting mechanism can only identify fake videos generated by the given methods, these results indicate that the video is very likely to be generated using the NeuralTexture method.

Model	Fake Count	Raw Count	Model Decision
NeuralTextures	897	774	Fake
Face2Face	0	1671	Raw
Deepfakes	1	1670	Raw
FaceSwap	0	1671	Raw

Table 2. Detection Results on Fake Obama Video

Secondly, a video generated by StyleGAN2 method [23] was used to exemplify the restrictions of the voting mechanism. This video is available at: https://www.youtube.com/watch?v=6E1_dgYlifc. The StyleGan2 method is a deepfake generation method proposed by the Nvidia team in 2020 that is well known for generating vivid and realistic manipulated facial images [23]. Since no fake images were generated or used in the video above, it is expected that the four models would all identify the video as real. The results were indeed confirmed as shown in Table 3.

Model	Fake Count	Raw Count	Model Decision
NeuralTextures	6	598	Raw
Face2Face	7	597	Raw
Deepfakes	0	604	Raw
FaceSwap	15	589	Raw

Table 3. Detection Results of StyleGAN2 Produced Video

6. Discussion

From the results data, it can be observed that when the Xception model was used to detect fake videos generated by the four mainstream deepfakes methods, all of them had high detection accuracy. However, this was based on the premise that the corresponding detection model was used for detection of each type of fake video. If the correspondence between the model and the testing data was not followed, for example, using a model trained by the Deepfakes method to detect fake videos based on the Face2Face dataset, we see from Figure 15 that the accuracy of detection is massively impacted (in some cases down to 1% accuracy).

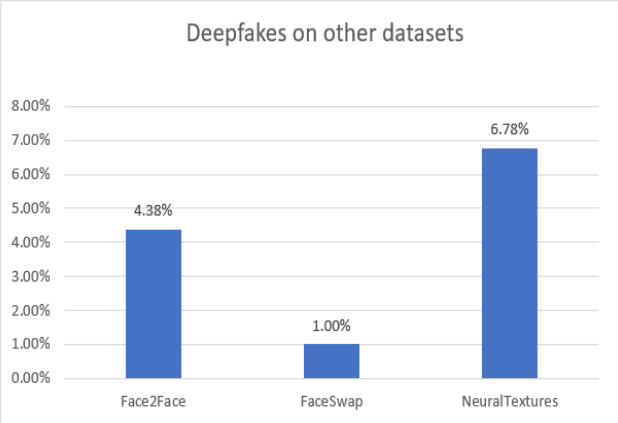

Figure 15. Deepfake Model Results on Other Datasets

This is not due to an overfitting of the model to the training set since the images inside each model test set have not appeared in the training set at all, and the model still achieves more than 94% detection accuracy with each testing set. Rather each model is incapable of detecting other types of deepfake videos because they have been trained with four bi-categorical classifiers, thus the classifiers are extremely sensitive to certain types of deepfake videos.

Existing deepfake video detection methods are designed to build video authenticity detection systems suitable for different kinds of deepfake videos. These detection methods usually select features which are common between each type of deepfake video, but the variety of these features is small and will not be present in every video. As a result, generic classifiers do not achieve a high degree of accuracy.

Instead, we chose to train four dichotomous classifiers based on four dominant deepfakes methods. We then used a voting mechanism to determine whether or not a video was a deepfake video. This detection method was able to detect videos generated by all four mainstream deepfake methods with a high degree of accuracy. Although this combined model worked well on the fake videos produced by the four applications studied, the model may fail on other types of fake videos generated from other approaches that are not included in the voting mechanism. Thus, none of the platforms could identify the StyleGAN2 videos as being fake. However, training a new model and adding it to the voting mechanism, should be able to detect this type of fake video as well.

The trained model is certainly capable of image detection however a voting mechanism is needed for video prediction when there are sensitivities based on the different model's performance at identifying fake videos generated using different deep fake platforms. The voting mechanism uses the outputs of all four trained models. For example, given 300 images produced by a video, the Face2Face model sets a threshold such that if over 50% of the images are predicted fake, then the video is classified fake by the Face2Face

model. If any of the four model outputs are classified as fake, then the video is considered fake. In principle, every model might have its own threshold for predicting whether a video is fake or not. For every model, the number of pictures classified as fake from a real or fake video should be separated by the recommended threshold based on experimental validity to avoid erroneous classifications.

In this work we set the threshold to 50% for all four models. There are two reasons for this. Based on the result section, both the TPR and the TNR for each model are high (>90%), therefore crests on a binomial distribution graph sit at each edge and look symmetric. We therefore take the symmetric axis (~50%) as the threshold. Figures 16 and 17 show the distribution of the number of fake / real (raw) pictures respectively detected from a fake/real video created on NeuralTextures using the Xception model. Although the accuracy of this model is not very high, we can see there is no overlap between the two distributions. With such a binomial distribution we may assume that the possibility of each picture being detected is independent. However, in real life, there are many highly realistic but fake videos and others that are completely unrealistic and hence easy-to-detect. Thus, we do not want the threshold to be too strict because each video may have more or less pictures that are detected/undetected.

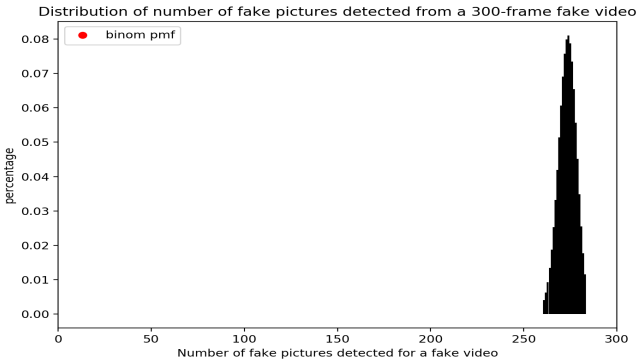

Figure 16. Distribution of Fake Pictures Detected from a 300-frame Fake Video

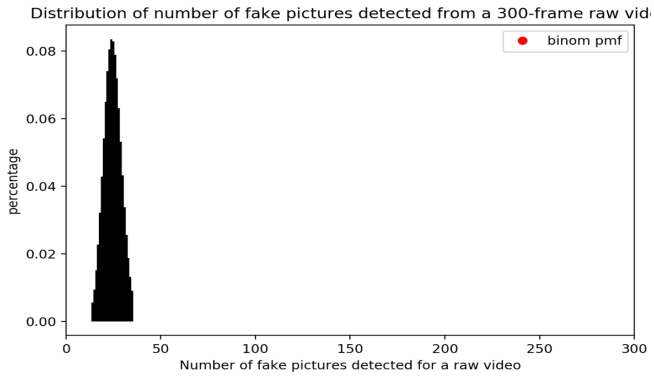

Figure 17. Distribution of Fake Pictures Detected for Real (Raw) Video

7. Conclusions

In this paper a total of eight deepfake video classification models were trained, evaluated and compared based on four fake video generation methods and two state of the art neural networks. Each model exhibited satisfactory classification performance over the corresponding dataset used to train it. Specifically, for the Xception models, the overall fake detection accuracy was above 90% and the model performed slightly better at detecting real videos compared to fake videos with a 2-3% increase in accuracy. The NeuralTextures model was an outlier in this test with the same detection rate of 91% for both true positive rates and true negative rates. For the MobileNets model the overall detection accuracy was above 90% for videos based on the Deepfakes, Face2Face and FaceSwap platforms but only 88% for videos produced on the NeuralTextures platform. The models have a similar detection accuracy for fake and real videos. The model trained with NeuralTextures was again an outlier with a 91% true positive rate and 86% true negative rate.

A voting mechanism was implemented to utilize the four models together to detect all four types of videos generated by the four mainstream fake video generating methods. This followed a simple model whereby any video classified as fake by any method was ultimately classified as fake. Other models are also possible, e.g. based on ranking and consensus, but given the sensitivity of the models to the specific platforms as shown in Figure 15, this approach was the most sensible to adopt.

In future work it would be worth exploring the impact of different loss functions and different optimizers on the results. Some researchers have also explored specific facial features as the dataset to feed in the model, e.g. the eyes, nose, ears or mouth. It would be interesting to compare model performance between a model trained with a whole face and models trained with partial facial features.

The classification described here is based on isolated pictures obtained from videos. No inter-frame pattern correlations were considered. Therefore, we did not consider video-oriented detection techniques, e.g. how many times the character blinks or identify any eye colour differences throughout the video. This would certainly be an area for future work.

Ideally this work would benefit from an easy to use front-end to draw more attention and garner interest from people. Ideally, the interface should allow users to upload a picture or video, raw or fake, and get a result from the model within a few seconds.

The Github link to this project software can be found at: https://github.com/BabyDelphin/COMP90055_Research_Project

Acknowledgments
The authors acknowledge use of the combined HPC/GPU facility (SPARTAN) at the University of Melbourne. SPARTAN was part-funded by an Australian Research Council LIEF grant.

References

[1] Alex Hern. (2020). *Facebook bans 'deepfake' videos in runup to US election.* Retrieved from https://www.theguardian.com/technology/2020/jan/07/facebook-bans-deepfake-videos-in-run-up-to-us-election

[2] WITNESS. (2018). *Mal-uses of AI-generated Synthetic Media and Deepfakes: Pragmatic Solutions Discovery Convening*. Retrieved from https://lab.witness.org/projects/synthetic-media-and-deep-fakes/

[3] Ondyari. FaceForensics. (2018). https://github.com/ondyari/FaceForensics

[4] Chollet, F. (2017). *Xception: Deep learning with depthwise separable convolutions*. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 1251-1258).

[5] Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., ... & Adam, H. (2017). *Mobilenets: Efficient convolutional neural networks for mobile vision applications*. arXiv preprint arXiv:1704.04861.

[6] Chawla, R. (2019). *Deepfakes: How a pervert shook the world*. International Journal of Advance Research and Development, 4(6), 4-8.

[7] Thies, J., Zollhofer, M., Stamminger, M., Theobalt, C., & Nießner, M. (2016). *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).

[8] Faceswap-GAN. (2018). GitHub repository, https://github.com/shaoanlu/faceswap-GAN

[9] Thies, J., Zollhöfer, M., & Nießner, M. (2019). *Deferred neural rendering: Image synthesis using neural textures*. ACM Transactions on Graphics (TOG), 38(4), 1-12.

[10] r/deepfakes. (2017). Reddit Post, https://www.reddit.com/r/deepfakes/

[11] Korshunova, I., Shi, W., Dambre, J., & Theis, L. (2017). *Fast face-swap using convolutional neural networks*. In Proceedings of the IEEE International Conference on Computer Vision (pp. 3677-3685).

[12] Matern, F., Riess, C., & Stamminger, M. (2019, January). *Exploiting visual artifacts to expose deepfakes and face manipulations*. In 2019 IEEE Winter Applications of Computer Vision Workshops (WACVW) (pp. 83-92). IEEE.

[13] Gardiner, N. (2019). Facial re-enactment, speech synthesis and the rise of the Deepfake

[14] Güera, D., & Delp, E. J. (2018, November). *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.

[15] Do, N. T., Na, I. S., & Kim, S. H. (2018). Forensics face detection from gans using convolutional neural network.

[16] Fabien, M. (2019). *XCeption Model and Depthwise Separable Convolutions*. Retrieved from: https://maelfabien.github.io/deeplearning/xception/#

[17] Wang, CF. (2018). *A Basic Introduction to Separable Convolutions*. Retrieved from https://towardsdatascience.com/a-basic-introduction-to-separable-convolutions-b99ec3102728

[18] Tsang, SH. (2018). Review: Xception – *With Depthwise Separable Convolution, Better Than Inception-v3(Image Classification).* https://towardsdatascience.com/review-xception-with-depthwise-separable-convolution-better-than-inception-v3-image-dc967dd42568

[19] Gulli, A. and Pal, S., 2017. *Deep learning with Keras*. Packt Publishing Ltd.

[20] Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. arXiv preprint arXiv:1502.03167.

[21] Zhang, Z., 2018, June. Improved adam optimizer for deep neural networks. In *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)* (pp. 1-2). IEEE.

[22] Sambhu, N., & Canavan, S. (2020). *Detecting Forged Facial Videos using convolutional neural network*. arXiv preprint arXiv:2005.08344.

[23] Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., & Aila, T. (2019). *Analyzing and improving the image quality of stylegan*. arXiv preprint arXiv:1912.04958.

[24] Lafayette, L., Sauter, G., Vu, L., Meade, B., S*partan Performance and Flexibility: An HPC-Cloud Chimera*, OpenStack Summit, Barcelona, October 27, 2016. doi.org/10.4225/49/58ead90dceaaa