2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)
Galgotias University, Greater Noida, UP, India. Oct 30-31, 2020

Trans-DF: A Transfer Learning-based end-to-end Deepfake Detector

Mohil Patel*, Aaryan Gupta†, Sudeep Tanwar‡, *Member, IEEE*, M. S. Obaidat, *Fellow of IEEE and Fellow of SCS*§
*†‡Department of Computer Science and Engineering, Institute of Technology, Nirma University, Gujarat, India
§College of Computing and Informatics, University of Sharjah, UAE, King Abdullah II School of IT, University of Jordan, Jordan, and University of Science and Technology Beijing, China
Emails: *17bce062@nirmauni.ac.in, †gupta.aaryan8@gmail.com, ‡sudeep.tanwar@nirmauni.ac.in, §m.s.obaidat@ieee.org

***Abstract*—With the advent of information and communication technologies, there have been breakthrough developments in the field of Artificial Intelligence (AI). Moreover, increasing computation power and decreasing processing times, new applications are being developed at great speeds. One such application is Deepfakes, which tackles the increased manipulated and forged media content. But these fake images and videos hamper the security and privacy of individuals and can have large-scale religious, communal, or political implications that may prove to be catastrophic for a nation. The face swapped content at times can be identified by human observation, but with the use of Generative adversarial networks (GANs), such forged content can be developed with is hard to be identified even by humans. Hence, detecting such videos and images is a challenging task for researchers. Motivated from these gaps, in this paper, we propose a pipeline for detecting and extracting human faces from videos, process them to extract features from them, and then classify them as real or fake. The results of the proposed model achieved an accuracy of 90.2% for classifying fake images from real ones.**

***Index Terms*—Deepfakes, Classification, Feature Extraction, Random Forest, VGG, ResNet, Inception, MobileNet, DenseNet**

I. Introduction

Deepfakes are an emerging area of research in the domain of Artificial Intelligence (AI), mostly because of their contradicting nature. They are quite an innovative and creative application of AI but are generally attributed as misuse of disruptive technology. A Deepfake can be defined as any form of multimedia, mainly images and videos, which has been manipulated, altered, or forged using some techniques such as deep learning and image processing or editing, in such a way that it seems very realistic to the human eye. With the advent of digital technologies and the Internet, accompanied by the smoother, faster, and multi-sourced flow of data, the conditions have proved to be perfect for the breeding of fake information. This fake information was previously found to exist in terms of fake news, photo-shopped images, edited videos, etc., but Deepfakes have made this scenario seriously contemplative because of their realistic and promising nature.

Although Deepfakes can be beneficial in many ways such as recreation of voices, movie scenes and even beloved deceased people, creation of a novel virtual fashion try-on, making advertisements and awareness campaigns multilingual, and assisting in a lot of medical innovations [1], they are generally linked with a lot of ethical concerns. Deepfakes are at times synonymous with face-swapping. Most people around the world have witnessed the working of face-swapping algorithms for fun, through applications such as Photoshop and FaceApp. But when face-swapping is not used for fun, it can be used for dangerous purposes. [2] discussed three ways in which face-swapping can pose as a threat - the general face-swap where a face is exchanged with another, the lip-sync where the mouth region of a person is exchanged with another person to alter what is being said, and the puppet-master where the whole face is re-animated with another person's face. These can be used for very dangerous activities such as creating a fake speech of a president to spread political propaganda or even to incite a terrorist riot, defaming a person or celebrity by embedding their face and body in a pornographic act, and tampering with evidence in a criminal case. In the statistical study shown in Fig. 1, it is quite evident that the number of Deepfakes available online is growing exponentially. Hence, there arises an urgent need for detection of these Deepfake images and videos, so that they can be separately identified from the real ones.

Detection of Deepfakes has been growing tougher with every new generator model coming into the research. The differentiating features between Deepfake and real media are getting more complex to identify, primarily because of the Generative Adversarial Networks (GAN) [4]. GANs are used to synthesize new data from scratch and have found immense applications in generating Deepfake data [5][6].

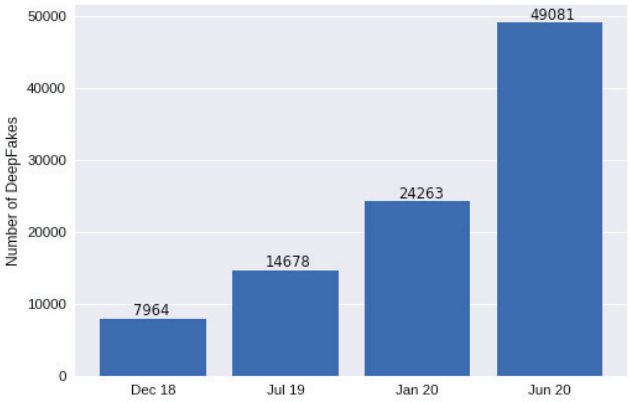


Fig. 1: Number of deepfakes identified online [3]

Simultaneously, detector models have been keeping up with the generators. Thus, both approaches can be viewed as two opposing sides, trying to get better. This paper majorly deals with the detection of Deepfakes from the video data.

There have been several approaches presented for the detection of Deepfake videos. But before the discussion on them, it is important to identify the datasets around Deepfakes. Since Deepfake detection requires the extraction of very complex features, it is necessary to have a large number of high-quality videos of both the fake and the real versions. Such a task has been recently accomplished in the form of datasets such as Celeb-DF [7], Deepfake Detection Challenge (DFDC) [8], and FaceForensics++ [9]. Now, among the most common choices for Deepfake detection, was the Convolutional Neural Network (CNN) with a combination with another model. While [10] used the CNN with optical flow for Deepfake detection and achieved a benchmark accuracy of 81.61%, [11] and [12] proposed CNN architectures combined with Long Short Term Memory (LSTM) networks, with the former achieving accuracy of 94.29%, and the latter achieving an F1 score of 0.95. [13] used an ensemble of CNN models. Another trend visible in the literature analysis is the use of Multi-task Cascaded Convolutional Neural Networks (MTCNN), which has been primarily used for face detection tasks. But researchers modified them with the help of other networks to generate excellent results for Deepfake detection. [14] used MTCNN along with a triplet network-based metric learning architecture to demonstrate the Area Under the Curve (AUC) up to 92.9%. [15] introduced a method based on MTCNN that extracts visual and temporal features from faces to accurately detect manipulations and derived an accuracy of 91.88%. [16] proposed an MTCNN classifier with a You Only Look Once (YOLO) algorithm to detect fake human faces created by both machines and humans. They employed pre-processing techniques and used ensemble methods to detect GANs-created fake images.

Since the features play a big role in Deepfake detection, there have also been various studies around the feature point detection for Deepfake videos. [17] employed a Support Vector Machine (SVM) classifier with various feature point detection algorithms such as Histogram of Oriented Gradients (HOG), Oriented FAST and Rotated BRIEF (ORB), Binary Robust Invariant Scalable Key-points (BRISK), Accelerated Segment Test (FAST), and Speeded-Up Robust Features (SURF) for extracting the key edge and corner feature points which could contribute to Deepfake detection, and got the highest accuracy of 94.5% with the HOG algorithm. [18] presented the use of Haar Wavelet Transform for analyzing the sharpness of edges to detect the Deepfake videos to get the AUC of 90.5%. There have also been several other methods which have resulted in very good results. [19] used the Expectation-Maximization (EM) algorithm and extracted a set of local features specifically addressed to model the underlying convolutional generative process to detect Deepfakes. [20] formulated Deepfake detection as a one-class anomaly detection problem and used a one-class Variational Autoencoder (VAE) to train on real face images and detect non-real images by treating them as anomalies, where they achieved an F1 score of up to 0.982. [21] proposed the application of attribution based confidence (ABC) metric for detecting Deepfake videos to display the accuracy of 96%.

A. Research Contributions

The following are the research contributions of this paper.

- We propose a Random Forest and face detection and feature extraction based model that classifies the input image as real or fake.
- We propose an end-to-end pipeline which inputs the raw video and outputs the classification result.
- Then, we compare various existing methods to detect Deepfakes.
- Finally, the performance of the proposed model is evaluated over various parameters like prediction accuracy, precision, and recall.

B. Organization of the Paper

The rest of the paper is organized as follows. Section II discusses the system model and problem formulation. Then, the workflow is conferred in Section III. Section IV demonstrates the experimental results, and finally, the paper is concluded in Section V.

II. System Model

Fig. 2 shows the system model of the proposed scheme. The main focus of the scheme is to identify if the video has been altered by face swapping or not. The scheme works in 3 phases - (i) face detection and extraction, (ii) feature extraction, and (iii) classification.

The first stage is to detect face from the video ($\mathcal{V}$) and extract it as an image ($\mathcal{I}$) which can be further preprocessed to extract important features ($\mathcal{F}$) to classify the image. Usually, whenever an image is swapped in a video there are blending problems such as non-uniform edges, non-coherence between the face and background image, etc. which can be targeted to detect the genuineness of video.

The extracted face images are passed through feature extractors to extract features that can be fed to classifiers to detect whether the image is real or fake. The quality of generated features depends on the feature extractor utilized. Better the feature extractor, the better will be the quality of features extracted and hence easier to classify the images. The feature extractors convolve over images selecting important features and passing them forward. The final output of extractors is a set of features which is utilized for classifying the images.

The extracted features set is utilized for classification. The classifier is trained using annotated data previously fed into the classifier. The classifier learns to differentiate between real and fake features during training and then new images can be tested for their authenticity.

III. The Proposed Approach

This section demonstrates the workflow of the proposed system. The system has three phases as shown in Fig. 3 - (i) face detection and extraction, (ii) feature extraction, and (iii) classification.

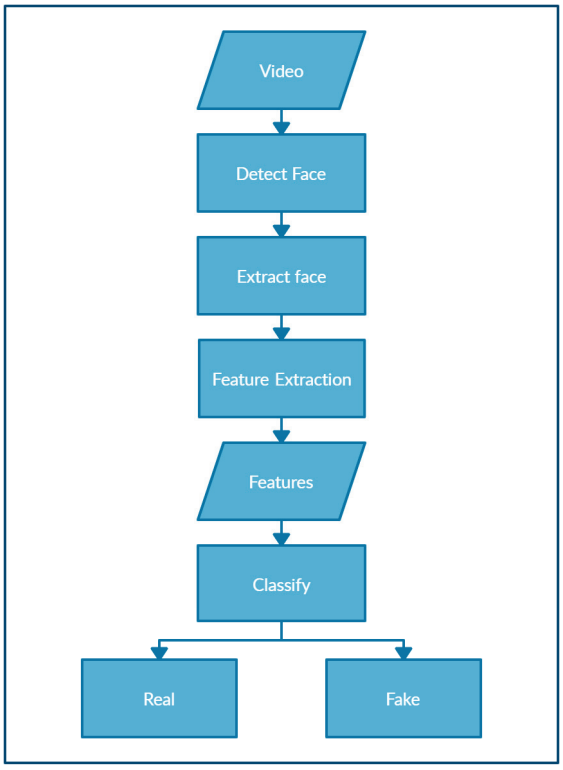


Fig. 2: System Model

A. Face detection and extraction

The first phase involves the detection of the face from the video and extracting it. A face detector module detects the face from certain frames from the video and then extracts and saves it as an image. For implementation purposes, the MTCNN model from the facenet-pytorch library was used to extract faces from the videos. Multi-task Cascaded Convolutional Networks (MTCNN) [22] is a model specifically developed for face and facial landmark detection. It is one of the most popularly used models for face detection because it is fast and works at good speeds even on CPU. It works in three stages, wherein each stage uses a CNN of different configurations to refine the results to the best extent possible. The dataset used in our study is the Deepfake Detection Challenge. Algorithm 1 shows the algorithm for extracting faces from the videos.

B. Feature Extraction

The second phase is feature extraction. Feature extraction will convert the face images into important features that will be useful for classifying the images as real or fake. The features can be extracted using various techniques. In this implementation, we are using pre-trained models as feature extractors. The reason to use pre-trained models is that they have been trained on thousands of images and are very efficient in extracting features that simple feature extractors may not be able to comprehend. The used feature extractors are VGG16, ResNet50, InceptionV3, MobileNet, and DenseNet. Each feature extractor was fed with an input image of size 128x128.

Algorithm 1 Face detection and extraction

Input: Video ($\mathcal{V}$)
Output: Images ($\mathcal{I}$) of faces extracted from different frames of the video

```
procedure GENERATEIMAGES(V)
    V ← loadVideo(V)
    model ← MTCNN()
    numOfFrames ← number of frames
    frameNum ← linspace(0, V.length(), numOfFrames)
    for num in frameNum do
        frame ← loadFrame(V, num)
        face, confidence_score ← model.detect(frame)
        if confidence_score > threshold then
            saveImage(face)
        end if
    end for
end procedure
```

VGG16 [23] is a CNN model which was proposed by K. Simonyan and A. Zisserman. The model achieved 92.7% top-5 test accuracy in ImageNet, a dataset consisting of over 14 million images of 1000 classes. The model architecture consists of several convolutional layers stacked with filters of size 3x3 and max-pooling layers. These stacks are repeated several times to build the whole model. The model outputs a feature map of (4,4,512), which is flattened to get a feature vector of shape (1,8192).

ResNet [24] is another highly accepted model for image classification. Resnet (residual neural network) is based on the fundamental idea of identity shortcut connections. It follows the hypothesis that simply creating a stack of identity mappings will perform similarly as a shallow architecture. Therefore, fitting a residual mapping with stacked layers is easier than directly fitting the desired underlying mapping. The model processes the input image and outputs a feature map of (4,4,2048), which is flattened to a feature vector of (1,32768).

Inception [25] is another CNN model that was developed by Google. It was an important milestone in CNN classifiers. Most CNN architectures just stacked convolutional layers to make a deeper model and increase accuracy. But this was a huge compromise on computational cost. The inception architecture aimed at increasing speed as well as accuracy. It suggested the use of sparsely connected network architecture. It uses 1x1 convolutions to limit the number of input channels to the next layers and hence reduce computation costs. The inception model processes the input image and outputs a feature map of (2,2,2048), which is flattened to a feature vector of (1,8192).

MobileNet [26] is a lightweight model developed for image recognition. It uses Depth-wise Separable Convolution to reduce the size and complexity of the model. It is useful for mobile and embedded applications. MobileNet processes the input image and outputs a feature map of (4,4,1024), which

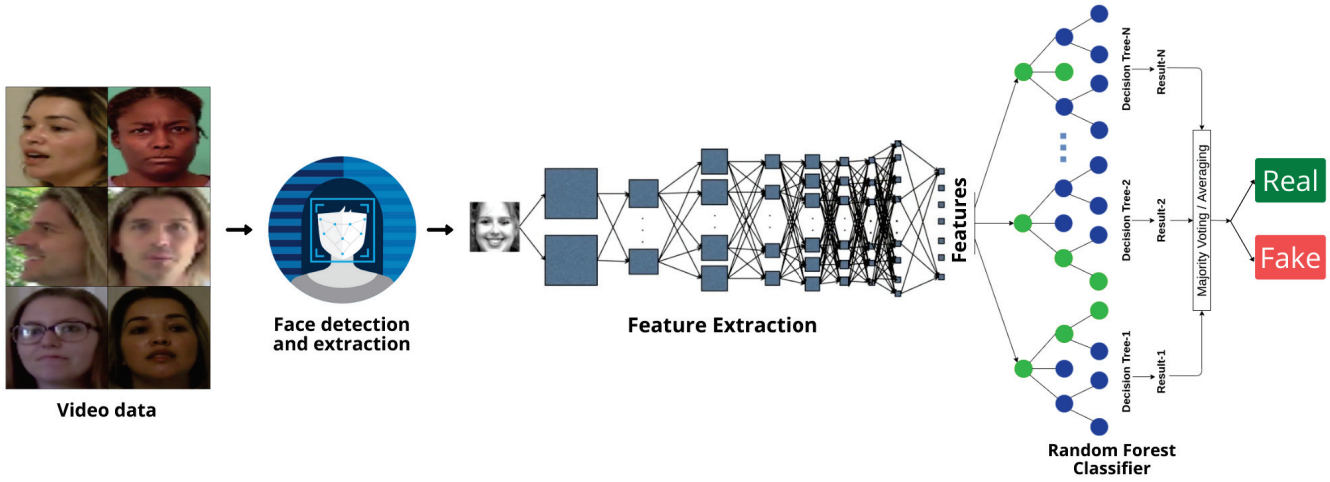


Fig. 3: System Architecture

is flattened to a feature vector of (1,16384).

DenseNet [27] is another breakthrough model in the image processing task. DenseNet is like an extension to ResNet but with a difference that each layer has inputs from all preceding layers and passes its output to all following layers. This makes the connections dense, increasing the accuracy with fewer parameters. DenseNet outputs a feature map of (4,4,1024), which is flattened to a feature vector of (1,16384).

Algorithm 2 shows the algorithm for extracting features from the image and assign labels and develop the dataset for classification. The dataset is fed to the classifier model for training and testing.

Algorithm 2 Feature Extraction

Input: Path ($\mathcal{P}$) to the extracted images, Model ($\mathcal{M}$) used for extraction

Output: Feature list ($\mathbf{F}$) of faces extracted along with labels

procedure EXTRACTFEATURES($\mathcal{P}$, $\mathcal{M}$)
  $\mathbf{I} \leftarrow$ readImagesFromPath($\mathcal{P}$)
  model $\leftarrow$ loadModel($\mathcal{M}$)
  $\mathcal{N} \leftarrow$ length($\mathbf{I}$)
  $\mathbf{F} \leftarrow$ emptyList()
  for image in $\mathcal{I}$ **do**
    features $\leftarrow$ model.extract(image)
    label $\leftarrow$ image.label
    $\mathbf{F}$.append(features, label)
  end for
  return $\mathbf{F}$
end procedure

C. Classification

The last step is to classify the features generated as real or fake. Many machine learning and deep learning algorithms are available for this purpose. One such algorithm is Random Forest. Random Forest is an ensemble tree-based algorithm. It uses a set of decision trees that are created from random subsets of training data and then combines the results from all trees to output the final class of the text input. Ensemble algorithms combine more than one algorithms, which may be the same or different. The advantage of this is that it produces high accuracies for classification while preventing overfitting on data. This is because the output is the average output of multiple algorithms, hence the error by one algorithm is balanced by others providing better results. Algorithm 3 shows the algorithm for classifying images after their features have been extracted. The data is split into training and test set and training data is fed to the model. The test data is used to compute the performance of the classification.

Algorithm 3 Classification

Input: Features ($\mathbf{F}$) and labels

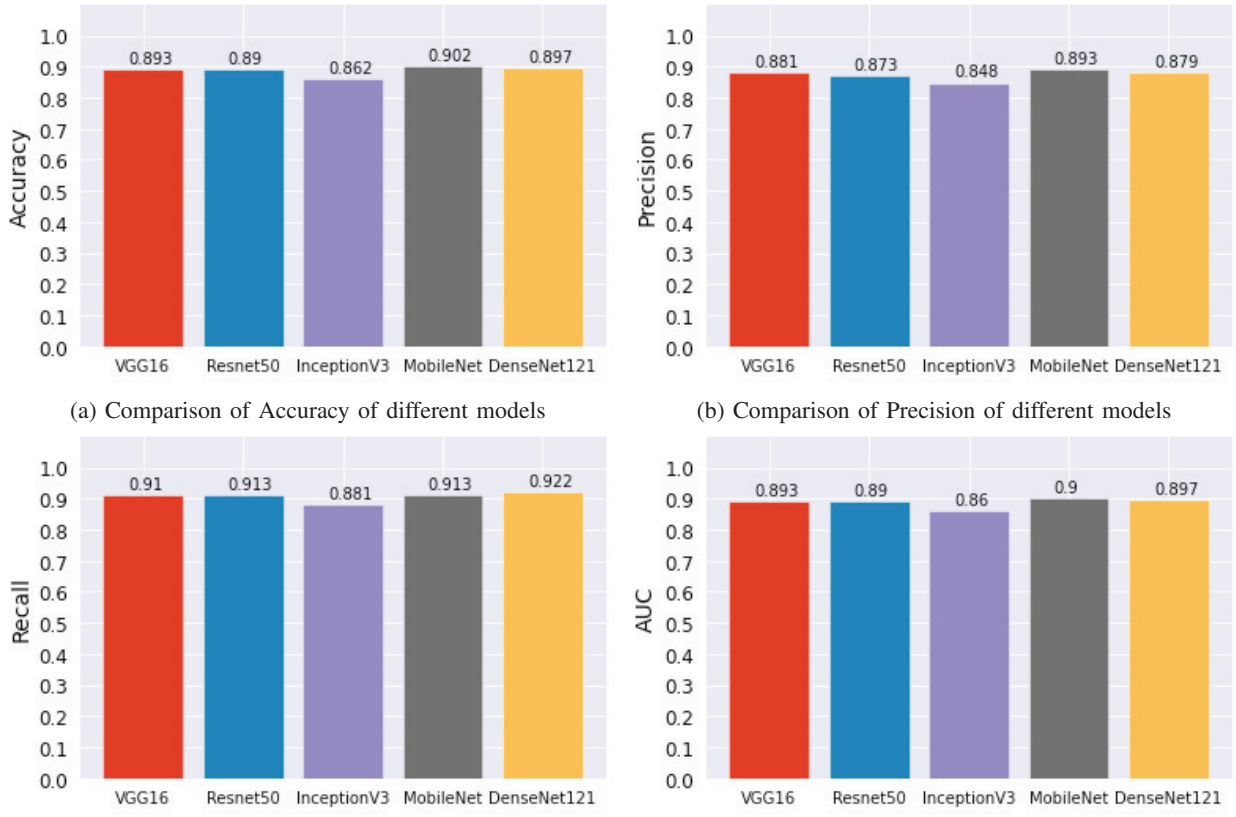
Output: Predicted classes ($\mathcal{P}$) of images as real or fake

procedure CLASSIFY($\mathbf{F}$)
  $\mathcal{D}_{train}$, $\mathcal{D}_{test} \leftarrow$ splitData($\mathbf{F}$)
  $\mathcal{M} \leftarrow$ RandomForestClassifier()
  $\mathcal{M}$.train($\mathcal{D}_{train}$)
  $\mathcal{P} \leftarrow$ emptyList()
  for ε in $\mathcal{D}_{test}$ **do**
    features $\leftarrow$ ε.features
    prediction_class $\leftarrow$ $\mathcal{M}$.predict(features)
    $\mathcal{P}$.append(prediction_class)
  end for
  return $\mathcal{P}$
end procedure

IV. Experimental Results and Discussion

This section discusses the performance of the proposed scheme with varying feature extractors. All the variations were tested on common data utilized for training and testing. The training set had around 7000 images and test set had 2000 images for classification.

Fig. 4 shows the comparison of the various metrics of



(a) Comparison of Accuracy of different models

(b) Comparison of Precision of different models

(c) Comparison of Recall of different models

(d) Comparison of AUC for different models

Fig. 4: Comparison of Various evaluation metrics for different feature extractor

the performance of the classification task. Fig. 4a shows the comparison of accuracies of the different feature extractors. Of all the models, MobileNet shows the highest accuracy of 90.2% followed by DenseNet121 with an accuracy of 89.7%. The least accuracy is shown by InceptionV3 (86.2%).

For DeepFake identification, identifying an image or video as Deepfake, while it is not is less costly than classifying a video which is a Deepfake as a real one. Hence the goal should be to optimize or decrease false negatives. Therefore, maximizing recall should be a priority. Fig. 4c shows the comparisons of recall obtained. The highest recall obtained is 92.2% by DenseNet121. The least recall of 88.1% is obtained by InceptionV3.

After optimizing false negatives, false positives should be optimized to reduce misclassification of real media. This involves optimizing precision. MobileNet shows the highest precision of 89.3%, followed by DenseNet121. Fig. 4b shows the precision obtained by all models.

The area under the curve (AUC) of the receiver operating characteristics (ROC) curve is an overall measure of the goodness of the classifier. Higher the AUC, the better the classifier. MobileNet has the highest AUC of 90% and InceptionV3 has the lowest 86% as evident from Fig. 4d.

For the given sampled data and methods, the best overall results are shown by MobileNet. However, DenseNet121 beats MobileNet in terms of recall which is rather more important, therefore DenseNet121 is preferred.

V. Conclusion

Deepfakes are relatively new in the research community but have opened new horizons of possibilities for AI. But the good always comes with some evil, and thus it has become crucial to building models for detecting Deepfake videos. This paper aims to present a simple end-to-end model, which can be used as a benchmark model for researchers and developers who want to work with Deepfake detection. The paper also contributes to the concept of transfer learning and how it can prove to be quite useful when we have limited computational resources and cannot train a deep learning model for several days on GPUs or TPUs. Various metrics such as accuracy, precision, recall, and AUC, have been used in the paper to conduct a more detailed analysis of how the models are performing. After observation of the results, it is safe to conclude that among all the models used in the study, MobileNet outperforms the other models. But it is also a critical observation that the other models are not far behind the MobileNet, and thus this implies the possibility that the other models can be further tweaked and developed into more accurate detectors.

References

[1] W. Mika, "The emergence of deepfake technology: A review," *Technology Innovation Management Review*, vol. 9, pp. 39–52, 11 2019.

[2] A. A. Maksutov, V. O. Morozov, A. A. Lavrenov, and A. S. Smirnov, "Methods of deepfake detection based on machine learning," in *2020 IEEE Conference of Russian Young Researchers in Electrical and Electronic Engineering (EIConRus)*, pp. 408–411, 2020.

[3] H. Ajder, "Deepfake threat intelligence: a statistics snapshot from june 2020." "https://sensity.ai/deepfake-threat-intelligence-a-statistics-snapshot-from-june-2020/".

[4] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[5] A. Brock, J. Donahue, and K. Simonyan, "Large scale gan training for high fidelity natural image synthesis," 2019.

[6] J. Thies, M. Zollhöfer, and M. Nießner, "Deferred neural rendering: Image synthesis using neural textures," *ACM Trans. Graph.*, vol. 38, July 2019.

[7] 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.

[8] 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.

[9] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner, "Faceforensics++: Learning to detect manipulated facial images," 2019.

[10] I. Amerini, L. Galteri, R. Caldelli, and A. Del Bimbo, "Deepfake video detection through optical flow based cnn," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, Oct 2019.

[11] I. Amerini and R. Caldelli, "Exploiting prediction error inconsistencies through lstm-based classifiers to detect deepfake videos," in *Proceedings of the 2020 ACM Workshop on Information Hiding and Multimedia Security*, IH&MMSec '20, (New York, NY, USA), p. 97–102, Association for Computing Machinery, 2020.

[12] M. F. Hashmi, B. K. K. Ashish, A. G. Keskar, N. D. Bokde, J. H. Yoon, and Z. W. Geem, "An exploratory analysis on visual counterfeits using conv-lstm hybrid architecture," *IEEE Access*, vol. 8, pp. 101293–101308, 2020.

[13] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo, "Gan is a friend or foe? a framework to detect various fake face images," in *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, SAC '19, (New York, NY, USA), p. 1296–1303, Association for Computing Machinery, 2019.

[14] A. Kumar and A. Bhavsar, "Detecting deepfakes with metric learning," 2020.

[15] D. M. Montserrat, H. Hao, S. K. Yarlagadda, S. Baireddy, R. Shao, J. Horváth, E. Bartusiak, J. Yang, D. Güera, F. Zhu, and E. J. Delp, "Deepfakes detection with automatic face weighting," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2851–2859, 2020.

[16] S. Tariq, S. Lee, H. Kim, Y. Shin, and S. S. Woo, "Detecting both machine and human created fake face images in the wild," in *Proceedings of the 2nd International Workshop on Multimedia Privacy and Security*, MPS '18, (New York, NY, USA), p. 81–87, Association for Computing Machinery, 2018.

[17] F. F. Kharbat, T. Elamsy, A. Mahmoud, and R. Abdullah, "Image feature detectors for deepfake video detection," in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*, (Los Alamitos, CA, USA), pp. 1–4, IEEE Computer Society, nov 2019.

[18] M. A. Younus and T. M. Hasan, "Effective and fast deepfake detection method based on haar wavelet transform," in *2020 International Conference on Computer Science and Software Engineering (CSASE)*, pp. 186–190, 2020.

[19] L. Guarnera, O. Giudice, and S. Battiato, "Deepfake detection by analyzing convolutional traces," 2020.

[20] H. Khalid and S. S. Woo, "Oc-fakedect: Classifying deepfakes using one-class variational autoencoder," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 2794–2803, 2020.

[21] S. Fernandes, S. Raj, R. Ewetz, J. S. Pannu, S. Kumar Jha, E. Ortiz, I. Vintila, and M. Salter, "Detecting deepfake videos using attribution-based confidence metric," in *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pp. 1250–1259, 2020.

[22] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016.

[23] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition. corr abs/1512.03385 (2015)," 2015.

[25] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision. corr abs/1512.00567 (2015)," 2015.

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

[27] G. Huang, Z. Liu, and K. Q. Weinberger, "Densely connected convolutional networks. corr abs/1608.06993 (2016)," *arXiv preprint arXiv:1608.06993*, 2016.