Lighten the Face Forgery Detection Network by using MobileNet

Shibo Wang
wangshibo1993@tamu.edu

Chih-Peng Wu
chinuy@tamu.edu

Abstract

The maturity of deepfake technology makes it possible for general people to use and create fake video, even with their mobile devices. Great concerns are rising, too. In this work, we extended MobileNetV3, a google open-source deep neural network architecture that is designed to run on mobile device, to perform face forgery detection. Our evaluation result shows that the prediction accuracy of MobileNet can be competitive against MesoNet, a deep neural network that is designed for face forgory detection, while at the same time, require less CPU cycle to predict fake images. We also extend the dataset by extract image frames from google deepfake video repository and showed that more training data will increase the accuracy

1. Introduction

The maturity of the deep learning techniques makes it possible to ”fake” videos that are almost impossible to be identified by human eyes. The so-called *Deepfake* is a technique for image and video synthesis that allows users to replace the face of a celebrity (or anyone else) in a video with another face from another person, regardless of gender, hairstyle, race, and other seemingly impossible shapes, as Fig. 1 demonstrated.

The *Deepfake* is so accessible already that there are productions (e.g., FaceApp [1] and FakeApp [2]) that can easily be acquired by general users who don't know how to control computer programs and can create highly ”real” video and images. Those videos generated by *Deepfake* have cause many social problems. For example, female celebrities were transformed into porn videos as porn stars [14]. There are more penitential misbehaved usages, such as fake news, fake surveillance videos, pornographic.

According to the malicious report by Brundage et al. [8], the growing use of AI systems has led to **Introduction to new threats**, and researchers should be also responsible to study how to mitigate the threats.

Figure 1. Example of *Deepfake* video (source: [18])

1.1. Related work

An authenticated digit media may have tampered when users share to a different website, social media and so on. Using deep convolutional neural networks (CNNs) to identify the authentic of camera images was proved feasible [19][11][12]. For video, progress was made through finding computationally cheap manipulations, for example dropped or duplicated frames [20] or copy-move manipulations [7].

Several recent works have already given different neural networks to make the classification of real and fake pictures and videos. Rossler et al. created and maintained a large scale video dataset for forgery detection [16], providing over 0.5 million edited images. And they used the Xception Net which is a complicated traditional CNN to approach the highest classification accuracy to date. Besides CNNs, another new and more complicated neural network – Capsule network forensics [15] – was also proposed and can have a better performance than some works based on traditional CNNs [9] [6].

1.2. Challenges

There are two challenging problems in current fake face detection. The first problem is there has already existed and maybe emerge more different mechanisms that can transfer one face to another, like *Deepfake*, *Face2Face*, and *FaceSwap*. Thus, the training dataset needs to maintain enough diversity. This problem seems to be solved to some

degree since many open-source codes provide their dataset and also Google has just released a bunch of fake videos to help people construct their dataset. And the second problem seems to be more severe. Those networks with outstanding accuracy are always complicated and need to cost much computing sources. Since several relevant mobile Apps of *Deepfake* have already been launched, our *Deepfake*-detection should also start paying attention to mobile apps and provide some lightweight networks.

2. Approach

To build a face forgery detection system on mobile devices, we tackled the problem with four approaches. First, we collected and created more images as the training dataset. In theory, the more data, the better-trained model would be. Second, we built an integrated pipeline to run both MesoInceptionNet and MobileNet. Third, the model reduction of MobileNet was studied and evaluated. Lastly, we test the run time resource requirement by inspecting the CPU cycle required for the prediction of the models we created.

2.1. Extended dataset

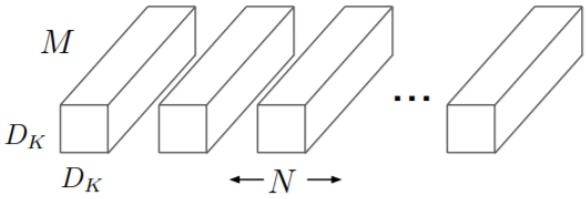
The authors of MesoNet released their dataset of 12353 images for training and 7104 images for testing. To get more images for training model purpose, we downloaded video collections (FaceForensics++ [16]) published in this September. The FaceForensics++ is a curated repository for face forgery video. We only used the one that was deepfaked by Google [10] for video quality issues.

We wrote a script to extract faces in videos. In hoping to balance the number of faces in the real versus fake faces, we only extract 31217 face images as training data and 12014 images as testing data. In other words, we tripled the dataset size. We named the new dataset as the extended dataset and will be used to evaluate model performance in the next section.

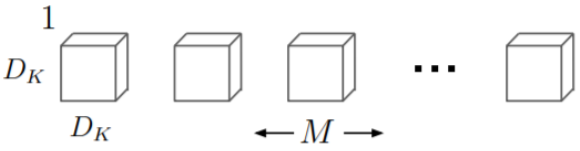
2.2. Pipeline integration

To lighten the network, we propose using the MobileNet [13][17] to replace traditional CNN, which splits the traditional CNN into depth-wise convolution and point-wise convolution to build light weight deep neural networks. The structure of MobileNet is shown in Fig. 2. Based on [13], the ratio of the computational volume of MobileNet to the traditional CNN is only about 0.1189. Besides using depthwise separable convolution to reduce the number of calculations and parameters, a recent and more advanced version, MobileNet V2 [17], also introduces the shortcut that has already succeeded in many popular networks like Resnet and Densenet. To adapt the MobileNet to shortcut, two tricks, inverted residuals, and linear bottlenecks are introduced to mitigate feature degradation. The difference between the MobileNet V2 and the ResNet is shown in Fig. 3.

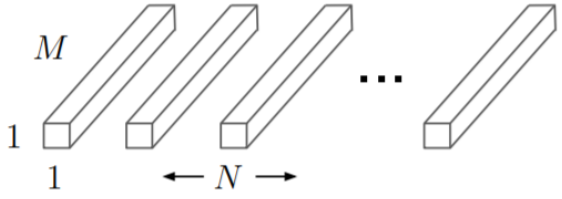
Inverted residuals and linear bottleneck are the main innovations of MobileNet. It adopts the strategy of first mapping the original information to high dimensions, then performing convolution feature extraction, and finally compressing, thereby retaining as much of the original information as possible. At the same time, to prevent Relu from further damaging features during compression, Relu was replaced with Linear bottlenecks. The model provided by the original paper contains an 11-layer *Inverted residuals + linear bottleneck* structure, which is a complicated neural network. Considering that our problem is a binary classification problem, an overly complex model will bring overfitting. At the same time, to improve the CPU calculation speed and reduce memory usage, so that the model can run smoothly on the mobile terminal, we did a series of experiments to observe the relationship between model complexity, accuracy and model resource consumption while reducing the model complexity. These results will be shown later in the next section.



(a) Standard Convolution Filters



(b) Depthwise Convolutional Filters



(c) 1×1 Convolutional Filters called Pointwise Convolution in the context of Depthwise Separable Convolution

Figure 2. The architecture of MobileNet [13]. The standard convolutional filters in (a) are replaced by two layers: depth-wise convolution in (b) and point-wise convolution in (c) to build a depthwise separable filter.

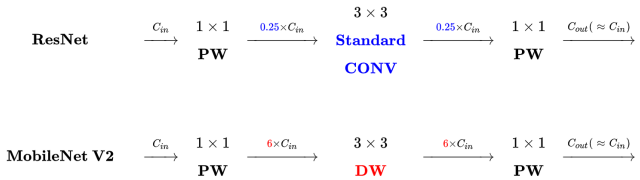


Figure 3. ResNet and MobileNet V2 comparison

Both MesoNet and MobileNet are available on github. For MesoNet, the authors released a source code on github [3], and Liu released another version using pyTorch [4]. We used the PyTorch version of MesoNet in our experiment. For MobileNet, we used Kuan Wang's PyTorch implementation [5] as our starting point.

2.3. Slim MobileNetV3

As one of our goals is to reduce the overhead of prediction, we want to compress the current MobileNet after which the prediction accuracy only fluctuates slightly but CPU calculation speed and memory usage of the model is optimized to run smoothly on the mobile terminal. As is shown in Fig 4[1], we can continue to compress the model, cutting out several *Inverted residuals + linear bottleneck* layers. In the evaluation section, We would show a series of experiments to demonstrate the relationship between model complexity, prediction accuracy, and model resource consumption while reducing the model complexity.

3. Evaluation

In this section, we wanted to answer following questions:

1. Does extended training dataset help?
2. What the performance (prediction) difference between MesoInception4 and MobileNetV3?
3. How much speed up that MobileNetV3 can bring?

3.1. Performance improvement using extended dataset

Our first experiment was testing the performance improvement because of more training data. We followed the same experiment setup of MesoNet [6], where the authors used 10% of the training data as the validation set and the testing dataset was never used in model training.

We drew the error of training and validation of using original dataset (i.e., the dataset from MesoNet), and our extended dataset in Fig. 5 and Fig. 6. We also drew the same plots of MobileNetV3 using original dataset and our extended dataset in Fig. 7 and Fig. 8.

Based on the figures, we concluded that epoch=20 is our optimal number; therefore, we used epoch=20 in all following experiments. The performance improvement using more training data reduced the error of MesoIncep-

[1] Visuzlized using https://github.com/lutzroeder/Netron

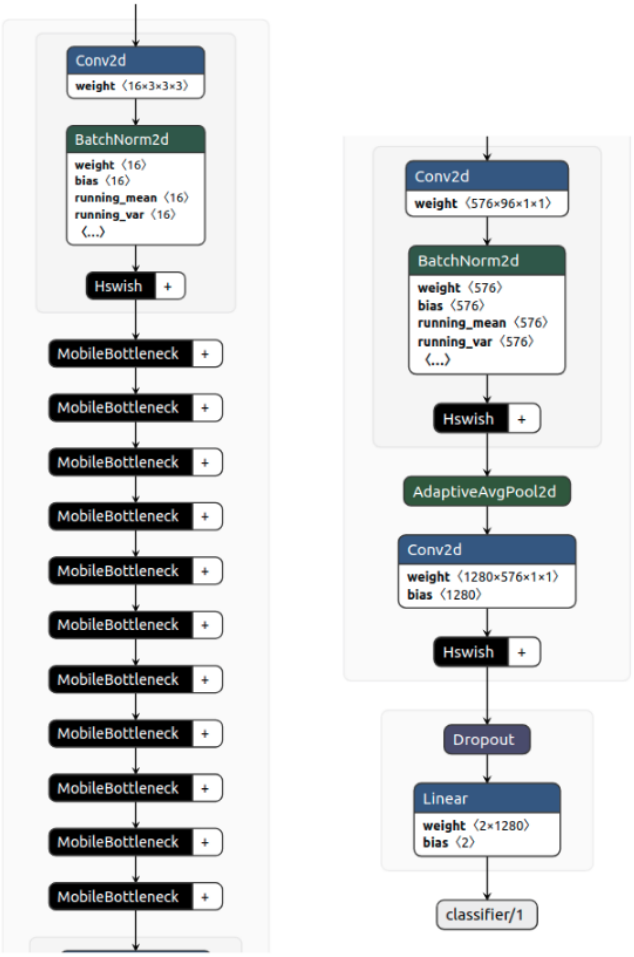


Figure 4. MobileNet V3 architecture

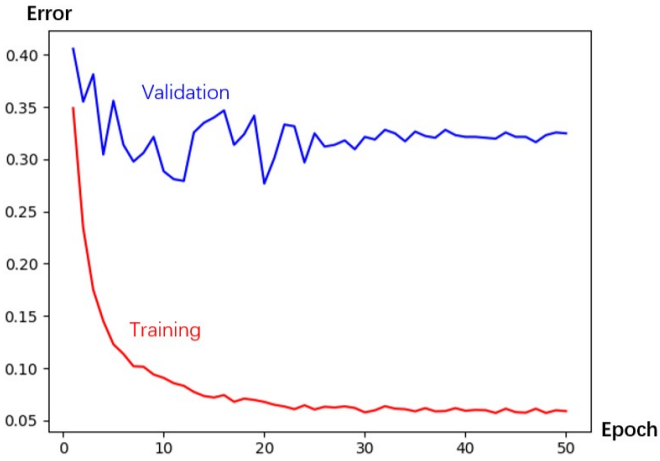


Figure 5. Cross-validation result of MesoInception4 using origin dataset as training

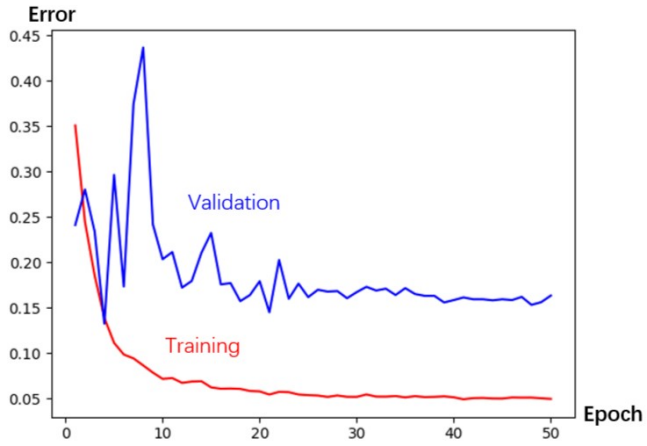


Figure 6. Cross-validation result of MesoInception4 using extended dataset as training

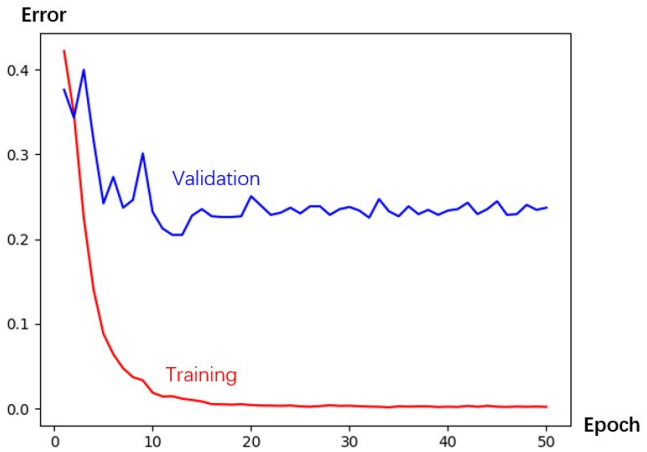


Figure 7. Cross-validation result of MobileNetV3 using origin dataset as training

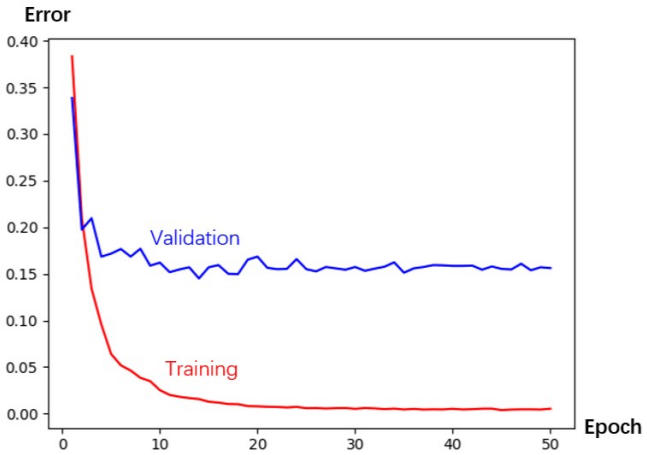


Figure 8. Cross-validation result of MobileNetV3 using extended dataset as training

MobileNetV3	positive	negative
positive	2334	511
negative	858	3401

MesoInception4	positive	negative
positive	2325	520
negative	103	4156

Table 1. Confusion matrices of MobileNetV3 and MesoInception4 using the original dataset as training

MobileNetV3	positive	negative
positive	2298	547
negative	1208	3051

MesoInception4	positive	negative
positive	2294	551
negative	298	3961

Table 2. Confusion matrices of MobileNetV3 and MesoInception4 using the extended dataset as training

tion4 from 0.32 to 0.17; while the error reduction of MobileNetV3 was from 0.22 to 0.15.

One thing surprised us was that MobileNetV3 has surmounted MesoInception4 in the error rate. One possible reason is the MobileNetV3 was released in 2019 and MesoInception4 was in 2018. The nonlinearity function, i.e., the *h-swish* function, used in MobileNetV3 may fundmentally improve a lot.

3.2. Prediction on testing dataset using MesoInception4 and MobileNetV3

Now we tested the prediction accuracy of the testing dataset using the model trained by the original dataset and the extended dataset. The number of testing images was 7104, which were downloaded from the MesoNet GitHub [3]. We showed the confusion matrices in Table 1 and Table 2.

We also drew the ROC curve of MesoInception4 using original and extended dataset in Fig. 9 and Fig. 10. And the ROC curve of MobileNetV3 using original and extended dataset were the Fig. 11 and Fig. 12.

Comparing the confusion matrices of MobileNetV3 and MesoInception4, we found that MobileNetV3 tends to predict images as ”fake”, which resulted in a more false-negative rate. Overall, the MobileNetV3 and MesoInception4 has similar performance.

3.3. Slim MobielNetV3

We want to reduce the overhead of the MobileNetV3.

3.4. Running time analysis

One of the goals of this work is to build a face forgery detection model using fewer CPU resources. This is par-

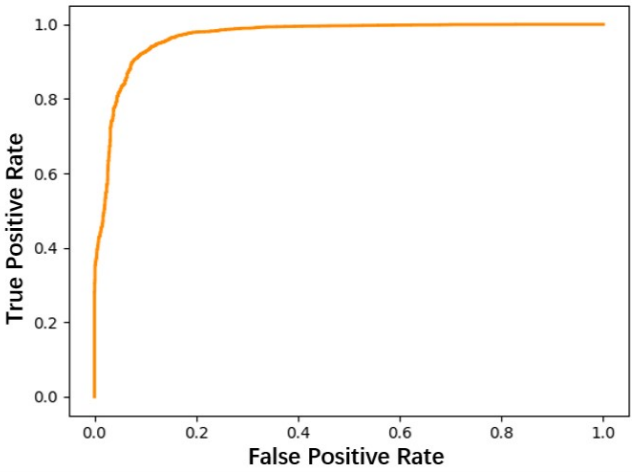


Figure 9. ROC curve of MesoInception4 using origin dataset as training

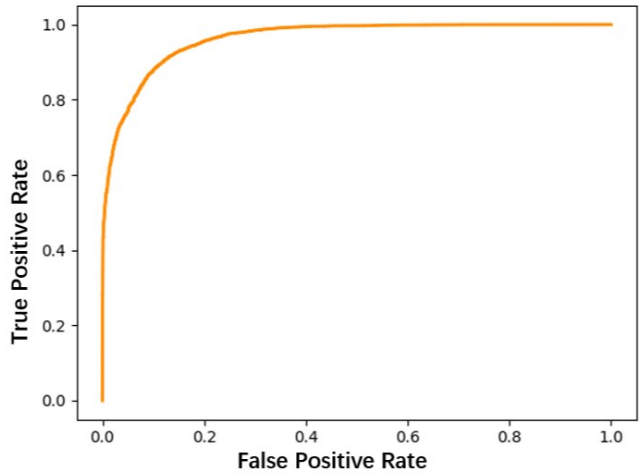


Figure 10. ROC curve of MesoInception4 using extended dataset as training

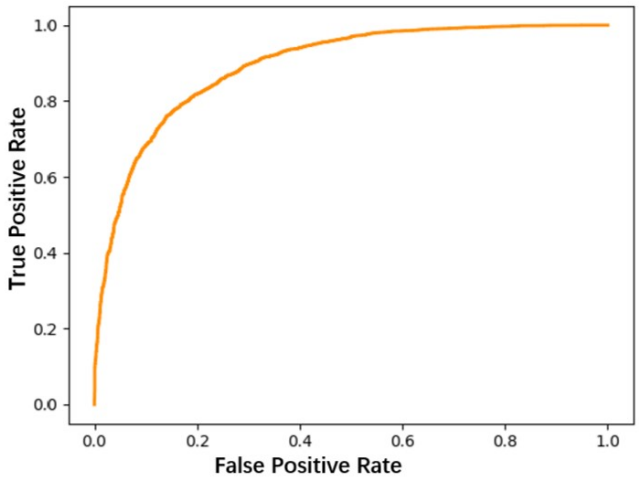


Figure 11. ROC curve of MobileNetV3 using origin dataset as training

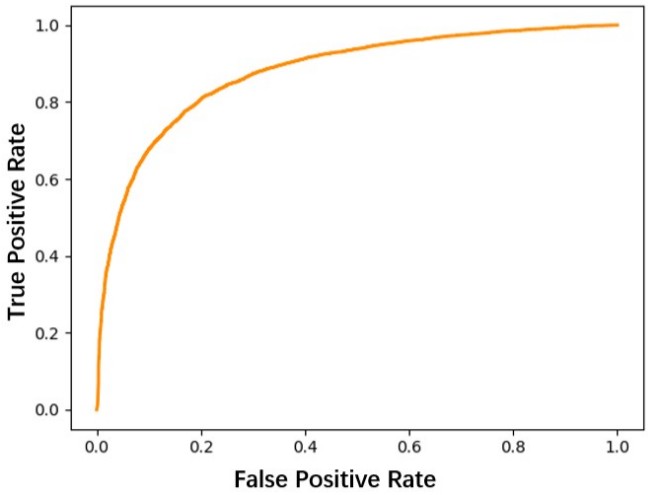


Figure 12. ROC curve of MobileNetV3 using extended dataset as training

ticularly important for a mobile device as there is no GPU support. The MobileNetV3 has done many optimizations for CPU environment, such as using h-Swish non-linearity function, and network architecture search.

We were interested in the CPU and memory usage of the model. For CPU usage, we used the Linux command **perf**[2] to get the CPU cycle to do the prediction. For memory consumption, we used **guppy3**[3] – a python package for heap memory usage – to get the heap memory usage.

For our 5 trained models: 1) MesoInception4, 2) MobileNetV3, and 3) MobileNetR1, 4) MobileNetR2, 5) MobileNetR3, where Rx means that we removed x number of layers in the original MobileNetV3. We ran a prediction task of 1000 images and measured the number of CPU cycles and memory usage. The result is depicted in Fig. 13.

[2]http://man7.org/linux/man-pages/man1/perf.1.html
[3]https://github.com/zhuyifei1999/guppy3

	Speed up	Speed up (wall clock)
MesoInception4	–	–
MobileNetV3	6.78%	8.26s
MobileNetR1	19.56%	23.84s
MobileNetR2	23.75%	28.96s
MobileNetR3	29.44%	35.89s

Table 3. Speed up in CPU cycle for the assigned prediction task.

According to the Table 3, MobileNetV3 required less CPU cycle to accomplish the same prediction task, while our reduced MobileNet model can further reduce the required CPU cycle. Considering the CPU specification of the iPhone 11 Pro Max is 2.7 GHz, the wall clock time speed

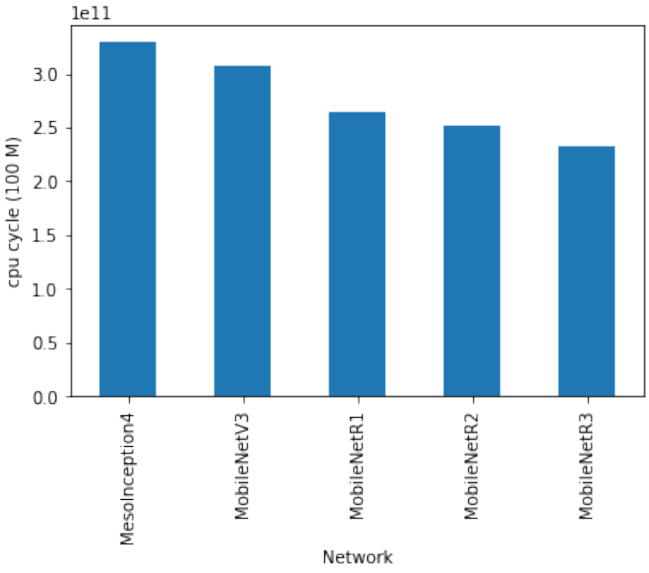


Figure 13. CPU cycle required for predicting 1000 images.

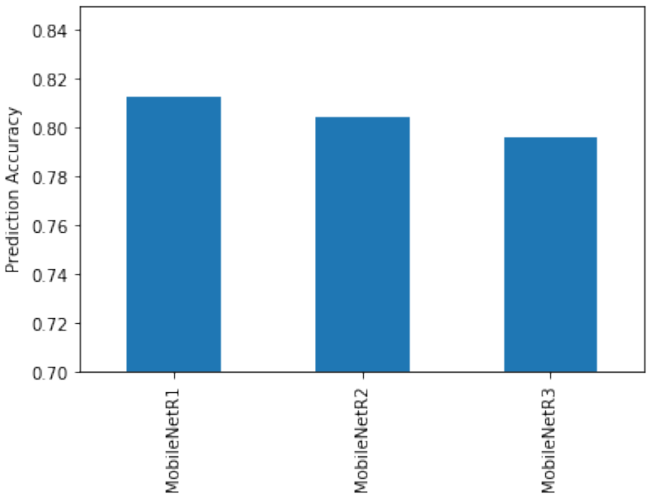


Figure 14. MobileNetV3 reduced models accuracy. R1, R2 and R3 represent the number of layers removed.

up of using the original MobileNetV3 models was 8.26s. Without reduced models, we can achieve 23.84s, 28.96s, and 35.89s, respectively.

As for the memory testing, we found all three models consumed mostly the same amount of heap memory (25 MB) so we ignore the result here.

We also showed the prediction accuracy was barely impacted due to reduced layers in Fig. 14. The prediction accuracy was reducing from 81% to 80% to 79%. The changes in prediction accuracy have been controlled within a small range, while the CPU cycle required to predict the same number of images has been reduced, which means the compressed model achieves our original optimization goal.

4. Discussion

4.1. Recognize face or distinct fake face

The original CNN model was designed for face recognition. In our early stage of working, we accidentally put the faces of the same people into both training and testing datasets. The prediction rate was over 90%. Though the prediction process was incorrect, we also learned that both models were a good fit for face recognition.

4.2. Further improvement idea

Currently, we only use CNN-like neural network architecture to perform the training and prediction. If we can integrate the nature of the video where images are sequence input and use LSTM or like neuron layers, we can foresee the prediction rate increase. Of course, the high cost of LSTM will deduct our original objective, i.e., design a deep neural network to predict fake faces on mobile devices.

4.3. Conclusion

We proposed a new angle to deepfake detection, that is, performing detection on mobile devices that have no GPU resource. In this work, we proposed a new approach of using MobileNetV3, a Google-developed deep neural network for mobile devices, and gain similar detection accuracy against the baseline model – MesoInception4. While keeping an acceptable prediction rate, the reduce MobileNet can reduce the CPU resource requirement and reduce the wall clock time by about 27 seconds when predicting 1000 images.

References

[1] Faceapp. `https://www.faceapp.com/`. Accessed: October 2019.

[2] Fakeapp. `https://www.fakeapp.org/`. Accessed: October 2019.

[3] Mesonet. `https://github.com/DariusAf/MesoNet`. [Online; accessed October-2019].

[4] Mesonet-pytorch. `https://github.com/HongguLiu/MesoNet-Pytorch`. [Online; accessed October-2019].

[5] A pytorch implementation of mobilenetv3. `https://github.com/kuan-wang/pytorch-mobilenet-v3`. [Online; accessed October-2019].

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

[7] P. Bestagini, S. Milani, M. Tagliasacchi, and S. Tubaro. Local tampering detection in video sequences. In *2013 IEEE 15th International Workshop on Multimedia Signal Processing (MMSP)*, pages 488–493. IEEE, 2013.

[8] M. Brundage, S. Avin, J. Clark, H. Toner, P. Eckersley, B. Garfinkel, A. Dafoe, P. Scharre, T. Zeitzoff, B. Filar, et al.

The malicious use of artificial intelligence: Forecasting, prevention, and mitigation. *arXiv preprint arXiv:1802.07228*, 2018.

[9] D. Cozzolino, G. Poggi, and L. Verdoliva. Recasting residual-based local descriptors as convolutional neural networks: an application to image forgery detection. In *Proceedings of the 5th ACM Workshop on Information Hiding and Multimedia Security*, pages 159–164. ACM, 2017.

[10] N. Dufour and A. Gully. Contributing data to deepfake detection research. `https://ai.googleblog.com/2019/09/contributing-data-to-deepfake-detection.html`, 09 2019. [Online; accessed October-2019].

[11] D. Güera, Y. Wang, L. Bondi, P. Bestagini, S. Tubaro, and E. J. Delp. A counter-forensic method for cnn-based camera model identification. In *2017 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 1840–1847. IEEE, 2017.

[12] D. Güera, F. Zhu, S. K. Yarlagadda, S. Tubaro, P. Bestagini, and E. J. Delp. Reliability map estimation for cnn-based camera model attribution. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 964–973. IEEE, 2018.

[13] 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.

[14] L. Matsakis. Artificial intelligence is now fighting fake porn. `https://www.wired.com/story/gfycat-artificial-intelligence-deepfakes/`, 2018. Accessed: October 2019.

[15] H. H. Nguyen, J. Yamagishi, and I. Echizen. Capsule-forensics: Using capsule networks to detect forged images and videos, 2018.

[16] A. Rössler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Nießner. FaceForensics++: Learning to detect manipulated facial images. In *International Conference on Computer Vision (ICCV)*, 2019.

[17] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4510–4520, 2018.

[18] H. Schellmann. Deepfake videos are getting real and that's a problem. `https://www.wsj.com/articles/deepfake-videos-are-ruining-lives-is-democracy-next-1539595787`, 2018. Accessed: October 2019.

[19] A. Tuama, F. Comby, and M. Chaumont. Camera model identification with the use of deep convolutional neural networks. In *2016 IEEE International workshop on information forensics and security (WIFS)*, pages 1–6. IEEE, 2016.

[20] W. Wang and H. Farid. Exposing digital forgeries in interlaced and deinterlaced video. *IEEE Transactions on Information Forensics and Security*, 2(3):438–449, 2007.