

DeepFake Detection in Deep Learning

Wen Li, Jianing Sun, Yi Xiang, Yanou Yang

Abstract – In this paper, we implemented three deep learning techniques to complete the deep fake detection challenge to detect manipulated media among all the mixed video. We applied feature engineering to the input video, which we should convert to image by meaningfully capturing frames in each video at first. For model part, we firstly built robust 2D-CNN model with Meso-Net neural network layers. In this model, we use the image-level input as our training data and took the 150th frame from our test video as our test dataset. Then, we implemented a 3D-CNN with applying feature extraction and background reconstruction, such as rescaling and changing background color to highlight emphasis. Overall, we achieved around 0.9 percent accuracy on our dataset.

Introduction

Deepfake (stemming from “deep learning” and “fake”) is a technique that can superimpose face images of a target person to a video of a source person to create a video of the target person doing or saying things the source person does. The underlying mechanism for deepfake creation is deep learning models such as autoencoders and generative adversarial networks, which have been applied widely in the computer vision domain.[1] These content generation and modification technologies may affect the quality of public discourse and the safeguarding of human rights—especially given that deepfakes may be used maliciously as a source of misinformation, manipulation, harassment, and persuasion. Identifying manipulated media is a technically demanding and rapidly evolving challenge that requires collaborations across the entire tech industry and beyond.[2]

The goal of our group project is to build robust models like 2D CNN and noise pattern classifier, to detect manipulated media among all the mixed video. In this project, our team wanted to design models which can successfully detect manipulated media among all the mixed video. We applied feature engineering to the input video, which we should convert to image by meaningfully capturing frames in each video at first. Then, regarding processed video, we applied feature extraction and background reconstruction, such as rescaling and changing background colour to highlight emphasis. Next, we tuned and trained 2D CNN and LGB models, based on the training dataset and tested their detection accuracy on the test set.

Methodology

Data Collection and Data Preprocessing. First, we extracted the name of each video and the label (true or false) corresponding to it in the metadata (a json file). For each video, we captured 20 pictures at a fixed time interval (30 seconds), and then use the algorithm in OpenCV to crop the

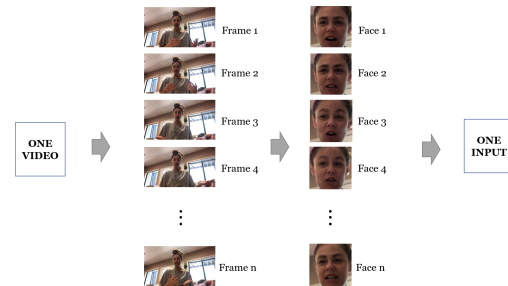


Fig. 1. The procedure of data processing

The Percent of Fake and Real Videos

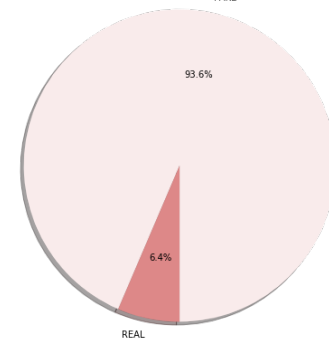


Fig. 2. The percentage of fake and real videos

face in each picture. If there are more than one face in the picture, only one face will be extracted. After that, we enlarged or shrank all the images of faces into the same size image and used them as one input in the model. So each video can generate several pictures of people’s faces at different times, and these several pictures will serve as one input data for prediction models. We handled more than 4000 videos overall, and split them into the training set and the test set.

Exploratory Data Analysis. Fake videos account for the vast majority of the entire data set (only 0.064 of videos are real). It is worth noting that these huge amounts of fake videos are modified by real videos which accounts for a small proportion in the dataset. All videos are 10 seconds long. We can find that some frames in fake videos have no details such as nose, eyes and teeth. The faces of characters can be

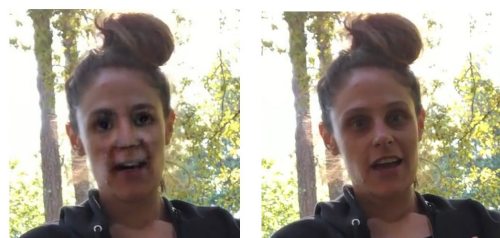


Fig. 3. The comparison of a frame in fake video and the original video

```
frames size: (20, 100, 100, 3)
```

Fig. 4. Feature Engineering: Extract Noises

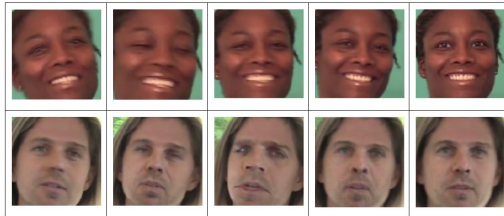


Fig. 5. consecutive capture of faces - video with label real/fake

distorted to a certain extent, which makes the frame blurry and even creepy. Moreover, some fake videos could be flickering, and you cannot even recognize and extract a face in some frames.

Model

Model1 3D-CNN. The features we worked on is the pattern noise of faces in each video. The assumption is that AI synthetic videos can have lower distinctions in each pixel which can be observed as the blurred components of faces, e.g. eyes, mouth, etc. And there are swaps of faces as well, especially in series of shots in videos. Therefore, to detect these swaps and abnormal blurs, we have the difference matrix of face,

Distinguishment = Exact Distribution (face images) - Gaussian (face images)

Overall, we capture faces in every 5 frames of the video (about 0.3 sec), and generate 20 continuous face-extractions for each video. Hence, each video corresponds to a 3-d distinguishment matrix of size (20,100,100), where 20 is the number of continuous face-capturing, (100,100) is the cropped distinguishment matrix for each face after interpolation.

Step1: Capture 20 continuous faces in each video with OpenCV packages

Step2: Generate 3-d feature set of size (20,100,100), which is the sequence of distinction matrices on a pixel level, and provides the foundation for models to capture sequential differences.

Real videos always have high distinctions, because they have nice and clear displays of every face component and no swaps, as shown in Fig5. In contrast, fake videos are with disruptively low distinctions, because there are always blurs in details of faces, e.g eyes and mouth, as shown in Fig5, and sometimes there are swaps of faces and therefore cameras cannot capture the faces, which all lead to low distinguishments in deep fake videos.

For example, the distinguishment metric on the left video with label fake is (50.12, 50.4, 49.7, 48.9, 49.2), and the

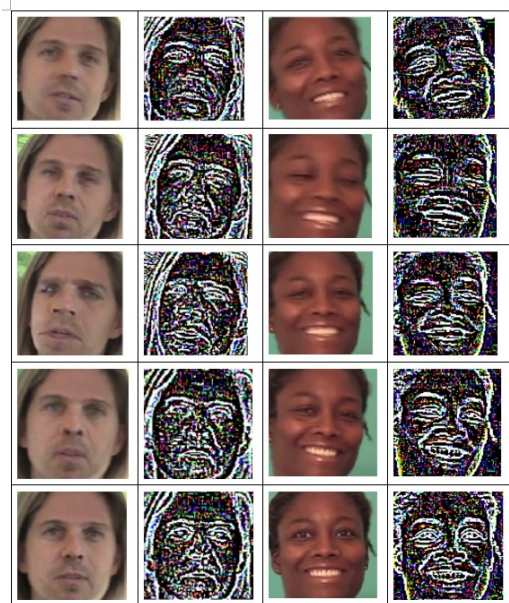


Fig. 6. Gaussian Blurred plots

```
Epoch 47/50
21/21 [=====] - 1s 39ms/step - loss: 0.4937 - acc: 0.8037
Epoch 48/50
21/21 [=====] - 1s 39ms/step - loss: 0.4988 - acc: 0.8037
Epoch 49/50
21/21 [=====] - 1s 38ms/step - loss: 0.4913 - acc: 0.8037
Epoch 50/50
21/21 [=====] - 1s 43ms/step - loss: 0.4985 - acc: 0.8037
```

distinguishment metric on the right video with label real is (54.34, 54.56, 52.8, 52.1, 52.78). That's why we extract distinctions as a feature set for videos. There's one thing to note that the colors of skins does not affect the distinction values, though we plot it in RGB form and colored faces seem more distinctive. In fact, the distinction value is based on the average of three primary colors, and therefore they will be cancelled out.

Now we have extracted out feature sets for each video, and the overall array is of size (400, 20,100,100), and over 400 MB.

Classify Videos with GRU and 3-d CNN on trained feature sets

Step1: Upsample the images with label real, because the labels are unbalanced as shown in Figure 2 in 2.2 Exploratory Analysis.

Step2: Split training set and test set on upsampled videos/images.

Step3: Train 3-d CNN to capture pixel noises and sequential swaps to predict labels from perspectives of both time and space.

Step4: Train 3-d CNN to capture pixel noises and sequential swaps to predict labels from perspectives of both time and space.

Model2 Meso-CNN. In the deep learning area, Convolutional Neural Network(CNN) is a special case of deep learn-

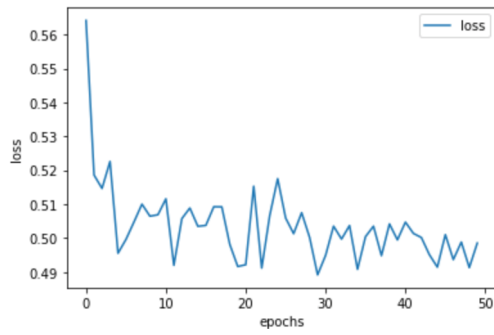


Fig. 7. Loss Plot of whole dataset

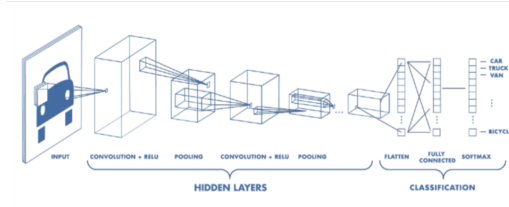


Fig. 8. The Typical CNN Architecture

ing networks. CNN is a pertificialy useful tool for imagery visualization. CNN represents a huge breakthrough in image processing and image classification. Image classification is the process of taking an input like a picture or photo and the output is normally a class or the confidence that the input is classified into a particular class. The basic layer includes Convolutional Layers, ReLU Layers, Pooling Layers and a Fully connected layer. The basic CNN architecture looks like Input -> Convolution -> ReLU -> Convolution -> ReLU -> Pooling -> ReLU -> Convolution -> ReLU -> Pooling -> Fully Connected.

CNN learns features from the input data using 2D/3D convolutional layers which makes it possible to be an ideal method for processing 2D images compared with other algorithms. They can learn the filters that have to be manually produced by other algorithms. CNN can be used in multifaceted applications including image/video recognition, image classification and natural language processing. CNN is inspired by biological areas regarding the vision in cats and monkeys. In these animals' eyes, they capture a picture of vision by the receptive fields which are restricted. These fields of regions partially overlap therefore the entire vision is able to be covered which is the same way as how CNNs work. [5]

Hyper-Parameter Tuning - MesoNet. To deal with either Deepfake or Face2Face problems, a unique network is not an effective approach.[3] In our project, we implemented our method by building a mesoscopic level of analysis. We cropeed the video and images only to focus on the face image in our project. The following architectures achieved the best classification scores among all our tests. They are implemented on well-performing neural networks architectures for image classification with convolutional layers , features extracted pools and dense classification. Specifically, we built a sequential Meso neural network model. Six layers are used in the neural network mode. Two of them are dense layers. The input image are in the shape of $150 * 150$ with depth

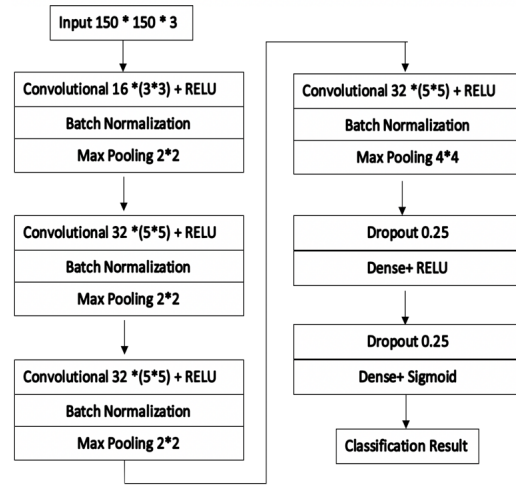


Fig. 9. MesoNet Model Architecture

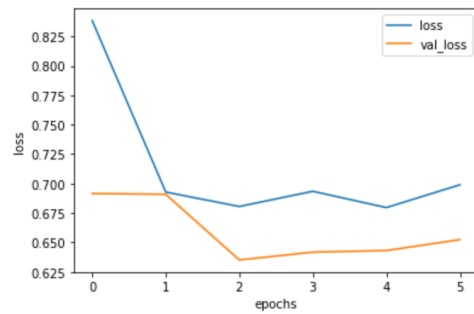


Fig. 10. 3-d CNN

of three. We built four layers of consecutive convolutional and pooling layers like Convolution -> ReLU -> Pool module. After that, there is one hidden layer and a dense network. We chose ReLU as our activation function for all the convolutional layers which would create non-linearities and used Batch normalization to take care of input and output to avoid the gradient effect. We add dropout 0.25 to improve the model robustness. In the output dense layer, the activation function is set to be sigmoid which helps model classification of two possible classes.

Results

Model1. In 3-d CNN, we upsampled videos with label real, and got an accuracy of 0.69 with a fast convergence. The input size is (400, 20,100,100), and the output is 0/1 binary value. The architecture is Input -> Conv(filters=20, activation=tanh) -> Max Pooling -> Conv(filters=16, activation=relu) -> Conv(filters=8, activation=relu) -> Global Pooling / Flatten -> Dense (activation=sigmoid) -> Fully Connected.

In the GRU model, we transformed input size to (400,20,1) where the last dimension is the average value of the distinction matrix, and the accuracy is 0.58 if we split training set and test set.

But accuracy reached 0.80 if we combine them as a whole training set. The reason is probably the lack of training samples, which eliminated the stability of model training.

```

Epoch 47/50 [=====] - 1s 39ms/step - loss: 0.6830 - acc: 0.5825
Epoch 48/50 [=====] - 1s 40ms/step - loss: 0.6808 - acc: 0.5858
Epoch 49/50 [=====] - 1s 39ms/step - loss: 0.6780 - acc: 0.5858
Epoch 50/50 [=====] - 1s 39ms/step - loss: 0.6790 - acc: 0.5858

```

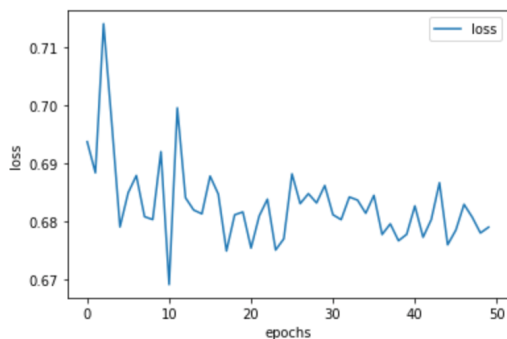


Fig. 11. Loss Plot of splitted dataset

Model2. We trained our model on GPU with 100 epoch and batch size 20 and we found that the accuracy result converged around 60 epoch and achieved 0.95 accuracy for training dataset with loss 0.1. Therefore, we stopped our training and saved the result at around 60 epoch to avoid overfitting. Please see the loss trajectory and accuracy trajectory before the epoch 60. From the graph below we can see the accuracy and loss result converged around 60 epoch.

Conclusions

In our first model, we focus more on the identification of noises and distinctions specifically on face components and

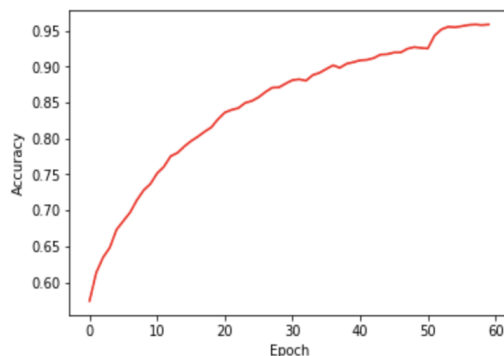


Fig. 12. Accuracy Plot for Meso-CNN (Accuracy vs Epoch)

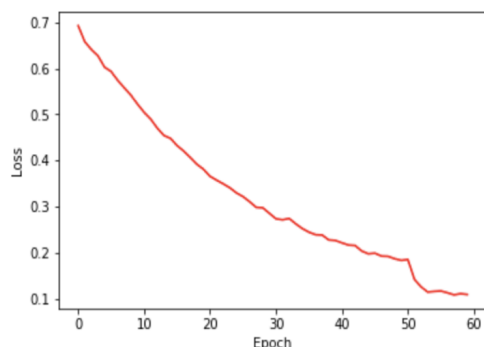


Fig. 13. Loss Plot for Meso-CNN (Loss vs Epoch)

head poses, which may contribute to better predictions of synthetical videos. [8] In our second CNN model, we built a MesoNet neural network model which can detect videos or images that are deep fake or real using deep learning techniques and feature engineering. Different from the first CNN model, we use the image-level input as our training data and took the 150th frame from our test video as our test dataset. After tuning our model using hyper-parameters, we achieved over 0.9 accuracy for image classification. With the limited time contribution, our top priority is tuning our existing model with higher accuracy. However, there are definitely aspects in our current model which can be improved. For future work, we would put more efforts on feature engineering process align more advanced algorithms to discover more key features for deep fake detection, such as using LSTM-CNN to construct frame-level sequence descriptor which are useful for classification, [6] constructing an XceptionNet CNN is used for facial feature extraction while audio embeddings are obtained by stacking multiple convolution modules. [7]

References

1. Nguyen, T.T., Nguyen, C.M., Nguyen, D.T., Nguyen, D.T. and Nahavandi, S., 2019. Deep learning for deepfakes creation and detection. arXiv preprint arXiv:1909.11573, 1.
2. <https://www.kaggle.com/c/deepfake-detection-challenge>
3. Darius Afchar, Vincent Nozick, Junichi Yamagishi, Isao Echizen. MesoNet: a Compact Facial Video Forgery Detection Network
4. <https://www.kaggle.com/unkownhihi/starter-kernel-with-cnn-model-ii-lb-0-69235>
5. <https://towardsdatascience.com/wtf-is-image-classification-8e78a8235acb>
6. Guera, D., and 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.
7. Mittal, T., Bhattacharya, U., Chandra, R., Bera, A., and Manocha, D. (2020). Emotions don't lie: A deepfake detection method using audio-visual affective cues. arXiv preprint arXiv:2003.06711.
8. Xin Yang, Yuezun Li and Siwei Lyu. Exposing deep fakes using inconsistent head poses
9. Ivan Perov, Daiheng Gao, et.al. DeepFaceLab: A simple, flexible and extensible face swapping framework. <https://github.com/iperov/DeepFaceLab>
10. [https://www.kaggle.com/timesler/facial-recognition-model-in-pytorch](https://www.kaggle.com/pathofdata/dpn50>Noise Pattern Classification
11. <a href=) Facial Recognition