Fake Video Detection

Shobhit Mehta, Prof. Nitin Sharma

Abstract

With recent advances in computer vision and graphics, it is now possible to generate videos with extremely realistic synthetic faces, even in real time. Our paper checks the realism of the videos, what are the parameters of detection fake videos, is high quality or low quality videos need different approach. This paper model for detecting fake or synthesis videos is generalize and can be trained for any type of videos.

1 Introduction

Recent advances in computer graphics, computer vision, and machine learning have made it increasingly easier to synthesize compelling fake image and video.In the static image domain, highly realistic images of people now be synthesized using generative adversarial networks (GANs). And, in the video domain, highly realistic videos can be created of anybody saying and doing anything that its creator wants. These so-called deep-fake videos can be highly entertaining but can also be easily weaponized. With recent Advances in machine learning and deep learning, in particular, the ability to learn extremely powerful image features with convolutional neural networks (CNNs). We tackle the detection problem by training a neural network.Now this can be trained in a Supervised fashion, because of the large-scale dataset of fake facial imagery and videos are available with us.Face2Face, FaceSwap, DeepFakes,Celeb-df and NeuralTextures are the well known large-scale dataset which contain million of images from thousand of videos.

The system preprcoesses the Input Video by collect data of video from Celeb-df dataset of 158 real videos and 795 synthesis videos, after collecting the videos we need to process the videos so can be used for classification of the real or fake videos. Next strp is to extract frames from the videos, i had

intentional extracted 47 frames from each videos, after this we need to extract faces from the frames which contain all the necessary features for the model to learn. Then train models we had layed off for detecting forgery. Our simple convolutional neural network (CNN) achieves 91.07% accuracy for detecting synthesis content on our dataset, 97.59% accuracy on Celeb-df. We also analyzed the effects of segment duration's, facial regions, image distortions,image shifting, rotating , re-scale, flip and dimensionality reduction techniques on mentioned datasets.

2 Dataset

A core contribution of this paper is our Celeb-df version 2 dataset extending the preliminary version 1 dataset This new large-scale dataset enables us to train a forgery detector for facial image manipulation in a supervised fashion. Because of the less processing power I had used only 40 percentage of the data of Celeb-F version 2, for training the model, which contains 785 fake or synthesis videos and 158 real or true videos , most of them are been taken from youtube.

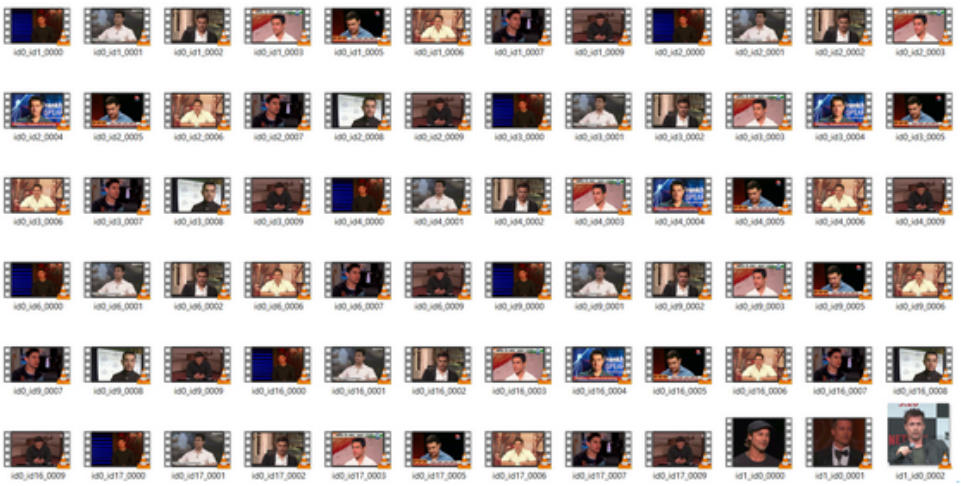

Figure 1: Dataset-fixed length videos

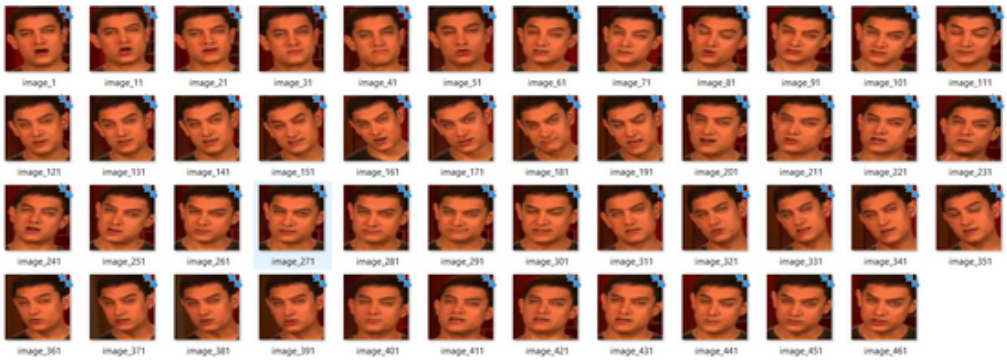

Figure 2: Final result after face-extraction using Mtcnn

2.1 Data Preprocessing

The version 2 of celeb- f contains the High Quality(HQ) videos otherwise we need to manipulate the videos. For creating the realistic setting To generate high quality videos, we use a light compression denoted by "HQ" which is visually nearly lossless. Low quality videos are produced using a quantization of 40. For the further development, we need to extract the faces from the videos , the pipeline for the end results is, first we need to separate video into frames, for extraction of frames from the video i had simply used open-cv operations,Video capture(), read(). And final part of data pre-processing is face extraction which can be done simple CNN, but its not effective for different features of faces, as it is trained only for the less amount of data.For getting efficient results i got i had used MTCNN or Multi-Task Cascaded Convolutional Neural Networks is a neural network which detects faces and facial landmarks on images.It's an algorithm consisting of 3 stages, which detects the bounding boxes of faces in an image along with their 5 Point Face Landmarks.

To make the model more efficient and also to save up on the overhead memory I had used Keras.preprocessing.imageImageDataGenerator class Make it easy to load images from disk and augment them in various ways.

3 Method

We cast the false video detection as per-frame binary classification problem of the videos contain both synthetic as well as real videos. We split the dataset into a training, validation and test set. For detecting from simple CNN image classifier, I add a convolutional 2D layer with 16 filters, a kernel of 3x3, the input size as our image dimensions, 200x200x3, and the activation as ReLU. After that we will add a max pool layer the halves the image dimension, stack 5 of these layers together. Finally, we'll flatten the output of the CNN layers, feed it into a fully-connected layer, and then to a sigmoid layer for binary classification. The model provide 73.92Xception by Google, stands for Extreme version of Inception, is reviewed. With a modified depthwise separable convolution, it is even better than Inception-v3 a depthwise separable convolution can be understood as an Inception module with a maximally large number of towers. This observation leads us to propose a novel deep convolutional neural network architecture inspired by Inception, where Inception modules have been replaced with depthwise separable convolutions. We show that this architecture, dubbed Xception, slightly outperforms Inception V3 on the ImageNet dataset (which Inception V3 was designed for), and significantly outperforms Inception V3 on a larger image classification dataset comprising 350 million images and 17,000 classes. We transfer it to our task by replacing the final fully connected layer with two outputs. The other layers are initialized with the ImageNet weights and assigning the parameter "trainable" equals true for the weights.

4 Architecture

Xception architecture is a linear stack of depthwise separable convolution layers with residual connections

- Model has 36 convolutional layers forming the feature extraction base of the network.
- We exclude the top of the network by setting include top is false.
- This excludes the global average pooling layer and the dense output layer

Figure 3: Our Model pipeline: face-extracted is fed into a learned classification network that outputs the prediction

- We then add our own global average pooling layer based on the output of the base model.
- Next for thinning of the weights, for reducing overfitting in the architecture by preventing complex co-adaptions on training data, We had used Dropout.
- Followed by dense output layer with one unit per class that is 2(false or true), using the softmax activation function.
- This is the final model for fitting the training data
- The each layers of the model architecture is open or non- freeze from top to buttom, accessible to be trained for the whole epochs.Optimizer used is a well known optimizer Nadam which is a combination of Adam and NAG.

5 Results

After training for 13 epochs there are no improvement in the "Val loss" that i had extracted from the model history for each epochs. Show in Figure 4
After running for 20 epochs i got an accuracy of 97.59 percent during training. For the testing, I had only tested on the 20 videos, and getting accuracy of about 96.2 percentage.

Figure 4: Training loss vs epochs

6 Work in Progress

the another Model i am working was based on LSTM(Long short-term memory) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. I had trained the model but getting some error with how the input drives in the model architecture. The LSTM model goes like,

- for the training of the model I need the different features of face, for which I am using Fab-net, but due to some error, at last i am using pre-trained models in CNN to get the features, other training data needed is labels which is provided in the dataset.
- The network has a visible layer with trainData,shape input, a hidden

layer with 32 LSTM blocks or neurons.The default activation function is used for the LSTM blocks. And an output layer that makes a single value prediction with softmax activation function. The network is trained for 25 epochs and a batch size of 32 is used.

- Once the model is fit, we can estimate the performance of the model on the train and test datasets.
- I had also added an temporal pooling layer, this layer accepts the temporal sequence output by a recurrent layer and performs temporal pooling, looking at only the non-masked portion of the sequence.this is shown to be capable of producing stable declarative representations of sequences that activate consistently for each item in that sequence.
- Just masking an LSTM layer with return sequence equals to true, and the dimensions are infered based on the output layer shape, and returning the mask layer.

7 Reference

- https://towardsdatascience.com/4-pre-trained-cnn-models-to-use-for-computer-vision-with -transfer-learning-885cb1b2dfc
- https://www.programcreek.com/python/example/93730/keras.backend.expand$_dims$
- https://towardsdatascience.com/cnn-based-face-detector-from-dlib-c3696195e01c
- https://qiita.com/li-li-qiita/items/1e7827ced044c3277c70
- https://github.com/ipazc/mtcnn
- https://openaccess.thecvf.com/content$_cvpr_2$017/$papers/Chollet_Xception_Deep_Learning_CVPR_2$017
- https://colah.github.io/posts/2015-08-Understanding-LSTMs/
- https://arxiv.org/pdf/1808.07272.pdf
- https://stats.stackexchange.com/questions/226593/how-can-calculate-number-of-weights-in-lstm/273595: :text=Each%20cell%20in%20the%20LSTM,from%20the%20previous%2

- https://medium.com/analytics-vidhya/demystifying-lstm-weights-and-biases-dimensions-c47dbd39b30a