Exploring methods for Deepfake Detection

Rutuja Jadhav, Shine Lin, Zachary Badger-Markey
Global Innovation Exchange
Tsinghua University
Beijing, 100084 China
University of Washington
Seattle, WA 98105
rt18@tsinghua.edu.cn

Dr. Hang Su
Department of Computer Science and Technology
Tsinghua University
Beijing, 100084 China
suhangss@mail.tsinghua.edu.cn

Abstract

With the recent advancements in AI technology in-particularly General Adversarial Networks, the creation of realistic fake media has become prevalent and easily accessible. In this paper we have explored a wide range of methods to detect fake images. We have analysed machine learning classification approaches as using Decision tree, boosted trees as adaboost, gradientboost and SGDClassifier coupled with image processing techniques to distinguish fake images from real. We have also explored deep learning methods as attention-based Convolutional Neural Network and encoding latent distribution using Autoencoder for learning real and fake feature distribution. The paper discusses all the explored approaches and their performance in detail.

Keywords: Deepfakes, real, fake, classify

1 Introduction

1.1 The problem

Deepfake is composed from Deep Learning and and means taking one person from an image or video and replacing with someone else likeness using technology such as Deep Artificial Neural Networks. They are created using a Generative Neural Network (GAN). Basically, the generator creates a fake video clip and then asks the discriminator to determine whether the clip is real or fake. Each time the discriminator accurately identifies a video clip as being fake, it gives the generator a clue about what not to do when creating the next clip.

The phenomenon rapidly invades the film industry and threatens to compromise news agencies. Large digital companies, including content providers and social platforms are highly interested in fighting Deepfakes.

Deepfakes 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.

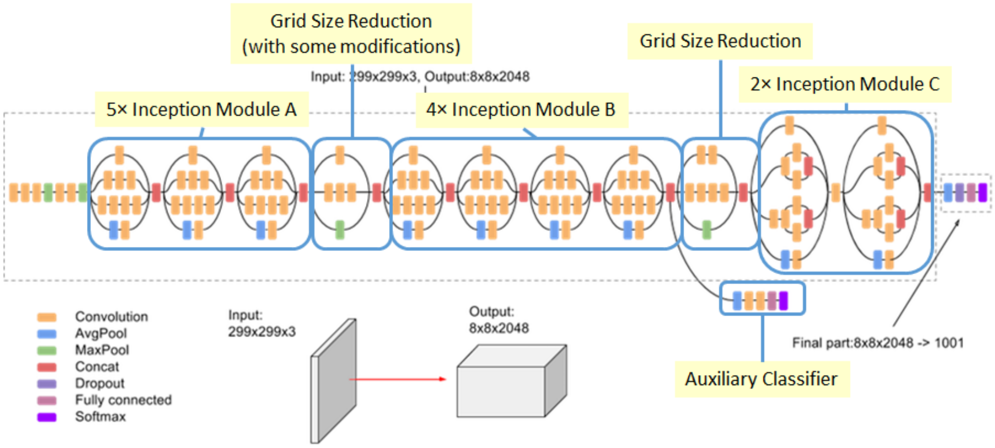


Figure 1: Inception V3 classification model

1.2 The dataset

The dataset upon which we tested our Deepfake detection algorithms is a subset of the bigger Kaggle Deepfake detection challenge. The full dataset had varying video dimensions and comprised 470 GB of data, labeled as fake or real people in the videos. The subset upon which we conducted our techniques were labeled 150x150 videos that consisted of 768 MB of data. Due the to time constraints of our project, we further edited this data, to only use sampled images from the 150x150 videos. This left us with over 80,000 images. The images themselves were comprised of primarily the person's face, with no other artifacts or objects in the images.

We split up these labeled images into two sets - one for training our algorithms and another for testing the efficacy of our model. We did a 90/10 split. It turns out there were many more fake images than real images in the data set - 64,773 fake compared to 12,130 real. To compensate for this, we simply chose a subset of the fake images to have the same number of samples. This left us with 12,130 real and 12,130 fake training samples. We similarly had 1258 real and 1258 fake testing samples.

The link to our dataset: `https://www.kaggle.com/unkownhihi/deepfake/`

1.3 The baseline

To use as a reference for our improvements and enhancing performance, we trained a baseline model without any image processing of the input images. The dataset was balanced to equalize number of real and fake samples. The unprocessed images were then passed through Inception-Net V3 convolutional neural network. The intuition behind using InceptionNet was the strength, flexibility and high reliability of the model for classification tasks.

2 Our machine learning approaches

2.1 Data preprocessing

Before we apply our machine learning approaches, we did several preprocessing to our dataset: grayify hogify, scaling, dimension reduction. We transfer the images with RGB three channel down to gray and extract the hog (Histogram of Oriented Gradients) features. The reason that we are doing this is because edge detection is found useful in deepfake detectioin. Then we do the normalization on data mean and standard deviation, we assume the mean is 0.5 and standard deviation is 0.5, which value can be tuned for performance improvement. Last method we apply is dimension reduction by using PCA (principal component analysis), there are two main reasons we use PCA: first, reduce memory usage and computation cost. After data preprocessing, each image is transformed with around 20 thousand feature dimensions and computer often runs out of memory when fit and train the model. Second, PCA also prevent model overfitting, where reducing feature dimensions lead to a

Figure 2: Original data sample

Figure 3: Grayified data sample

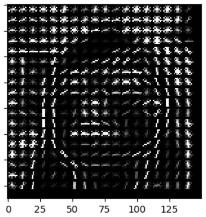


Figure 4: Hogified data sample

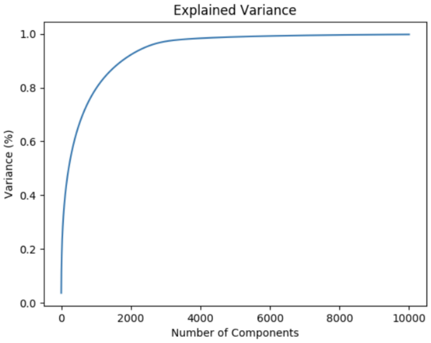


Figure 5: Information contained with numbers of dimension

smaller model and reduce the chance of overfitting. The desired dimensions we found out is around 2000 where can represent around 90% of information.

2.2 AdaBoosting and GradientBoosting

Adaboost is original designed for binary classification which is suitable for our deepfake challege, separating fake human images from real images. The principle of adaboost is combine multiple weak classifications into a strong classification. We then update the weights of different weak models, in our case, we choose 50 weak models and get 70% accuracy. Gradient Boosting is similar with Adaboost where both two are based from decision tree and update its weights. The main difference between Gradient Boosting and Adaboost is that Gradient boost updates the residual by its gradient while Adaboost reweights its weak models.

2.3 Stochastic Gradient Descent Classifier

Stochastic Gradient Descent is to minimize the loss to achieve global or local optimal by learning the gradient of loss in each iteration and update its weights. We apply hinge loss and view the training as a SVM(Support vector machine) problem to find the maximum margin for our binary classification problem. There are many variables for us to tune in future such as learning rate, loss function, etc,.

3 Deep learning approaches

After exploring ML based approaches, we worked with 2 deep learning methods. Both are discussed in this section.

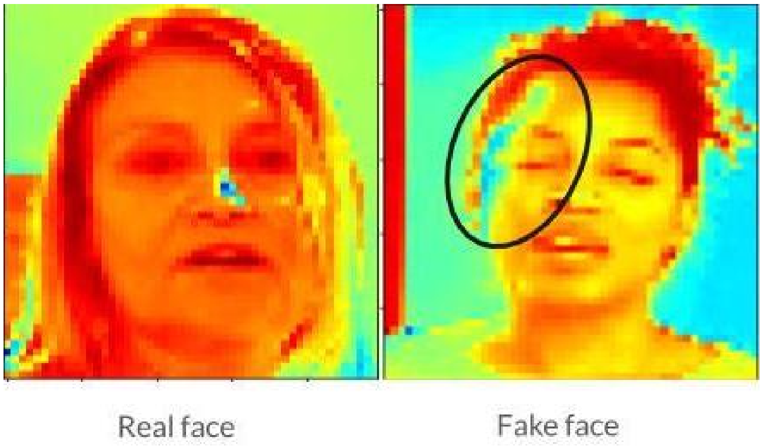
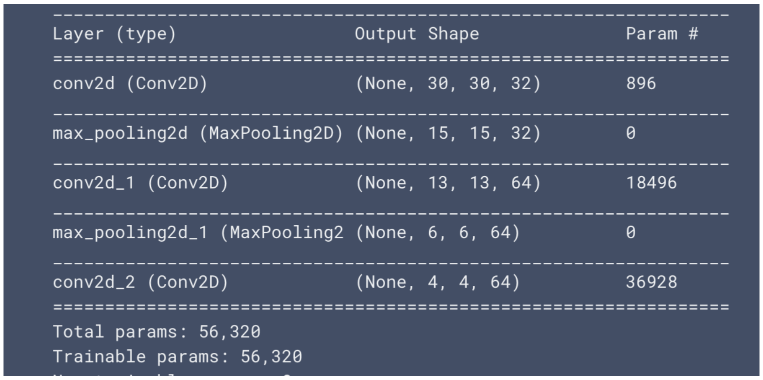


Figure 6: Sample real and fake pre-processed images using HEATMAPJET



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 30, 30, 32)        896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 15, 15, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 13, 13, 64)        18496
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 4, 4, 64)          36928
=================================================================
Total params: 56,320
Trainable params: 56,320
```

Figure 7: CNN model details

3.1 Facial gradients based Convolutional Neural Network

3.1.1 Pre-processing

Observing the inconsistency in facial features such as pixel value ranges, connectivity and uniformity. We were willing to explore a way to capture differences in the gradients of facial attributes. We used heatmap technique to exactly get the preprocessing of desired format. Heatmaps can be used on a wide range of colors. We specifically used HEATMAPJET that can be helped to capture gradients through VIBGYOR spectrum. Figure 2 helps show the difference in outcomes for a real vs fake image. The real image doesn't have a sharp variance across the gradient however we can detect a huge patch for a fake image which has been highlighted.

3.1.2 The model

The pre-processed images were there passed through a attention-based CNN which focused on capturing gradient variances. The CNN has a total of 56,000 trainable parameters. We chose to train on a lesser deep architecture taking our relatively small data-size into account. The model is inspired from traditional LeNet-5 architecture. Figure 3 showcases details of the model parameters.

3.2 Performance

The model gave a huge boost in our accuracy results in comparison with our machine learning performance. The validation accuracy rose to 0.95 and F1 score to 0.87.

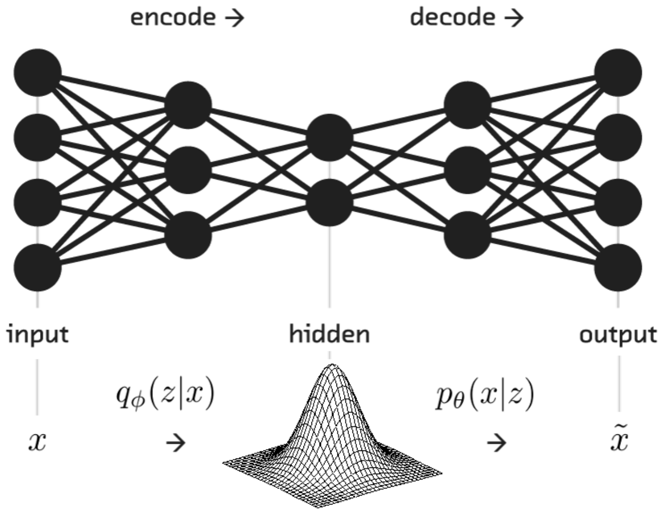


Figure 8: Autoencoder to learn latent space distribution

3.3 Learning feature distribution through autoencoder

3.3.1 The model

The motivation for trying an autoencoder approach was to use the latent space for learning distribution of features for real and fake data samples. 2 autoencoders were trained to learn the latent space distribution for real and fake data subsets. The z-score was calculated between the test data samples and the two distributions to predict the class. Through this bi-encoder approach we used utilized the latent space compression feature of a traditional autoencoder for classification.

3.3.2 Performance

The validation accuracy obtained through this approach is 0.94 and F1 score is 0.85 The model performs better than our machine learning techniques and is in par with facial gradient CNN performance. The Facial gradient CNN remains our best classification model with a slight margin.

4 Discussion

Deepfake detection is a challenging problem to solve. Basic machine learning techniques are not enough to detect this rapidly-evolving field. Starting small with basic machine learning gave us insight into the features and scale up into deeper techniques. This approach is something that we found very useful and will use it in future projects.

Sampling is important. Having the same number of real and fake images influenced the accuracy, especially of the more basic machine learning models like stochiastic gradient and boosting. Similarly, pre-processing is influential to the efficacy of our models, as it influenced the accuracy and F1 scores a lot.

In our experience with this project, deep learning is better than regular machine learning for deepfakes. The accuracy difference was over 30 percent. However, when we were more mindful about the type of deep learning technique and the pre-processing, we had better results than simply using a deep learning technique with many layers. In this case we are referring to the InceptionNet vs Facial Gradient.

While the tools available online have improved rapidly over the past few years, we found that processing large amounts of data can lead to memory issues. We were only using 150x150 images -

Table 1: Explored models and their performance comparison

Part		
Approach	Test accuracy	F1 (μm)
Baseline Inception	0.63	0.6
Stochastic gradient	0.57	0.57
AdaBoost	0.57	0.56
GradientBoosting	0.59	0.61
FacialGradient CNN	0.95	0.87
BiModal encoder learning	0.93	0.85

whole videos would have lead to even more issues than we experienced and we were thankful we didn’t have to go through that.

If we had more time, we would ensemble these different techniques that we tried in order to improve the accuracy even further.

References

[1] https://www.kaggle.com/unkownhihi/deepfake/

[2] https://www.kaggle.com/c/deepfake-detection-challenge