Deep Fake Detection: Non-Facial Feature Cropping, and a Novel Conditional Loss Function

Joshua Denholtz
denholtz@stanford.edu
SUNet ID – 06438358

Abstract

Deep Fakes are a threat to trust in information and digital media, this paper seeks to analyze numerous binary classification tasks indicating if a video is or is not a deep fake. This is a uniquely challenging problem as deep fakes have become very difficult for humans to classify. Xception model is used with several data processing techniques such as error level analysis and methods of removing trivial image regions. A novel conditional loss function is proposed. The end results are further explored with occlusion sensitivity to interpret what is being done by the network. Despite the model complexity, learning was not found to take place. Deep Fakes have a very low Bayes Error, and the class imbalance leads to the model over predicting images as "Fake." Numerous methods for future work are proposed to resolve this challenge.

1 Introduction

Deep Fakes are fake digital images, audio and video created using deep neural networks. This technology can generate hyper realistic digital media that is rarely distinguishable to humans. The open source nature of deep fakes has democratized this powerful technology and can be harnessed as fake news, harm trust in digital media and one day may be an existential threat to society. Being able to detect forged digital media is a top priority for deep learning engineers, and this is the goal under investigation. The outcome will be an analysis of the state-of-the-art image classification methods, and novel strategies for deep fake detection.

2 Related Work

Rapid Object Detection using a Boosted Cascade of Simple Features[12] – Fast machine learning based method to detect objects in images, notably images with a high detection rate.

Realistic Image Synthesis and Classification[13] – Similar work to compare real images and computer-generated images, this work is limited by the fact the data set was comparing easily distinguished computer-generated images which came from video games, it also highlighted Error Level Analysis as a basis for detecting forged images

FaceForensics++: Learning to Detect Manipulated Facial Images[2, 3] – Makes benchmarks for numerous machine learning models. This shows that the Xception model is the highest performing algorithm on Deep Fakes

3 Dataset Description

The dataset used for this project is the deep fake data provided by AWS, Facebook, Microsoft, the Partnership on AI's Media Integrity Steering Committee for their Deep Fake Detection Challenge (DFDC) hosted on Kaggle.

The dataset is comprised of approximately 75,000 mp4 file format videos of 10 seconds with a label of 'REAL' or 'FAKE.' The dataset is very large and not hosted on Kaggle, it needs to be downloaded as 50 separate zip files of 9-10 Gb for a total of 470 Gb. The videos are not a consistent shape, although most are 1080 x 1920. Due to the size of the data set, storage and memory issues were non trivial.

The data is unstructured and requires significant processing prior to any deep learning.

Figure 1: Side by side of real and fake image

4 Challenges

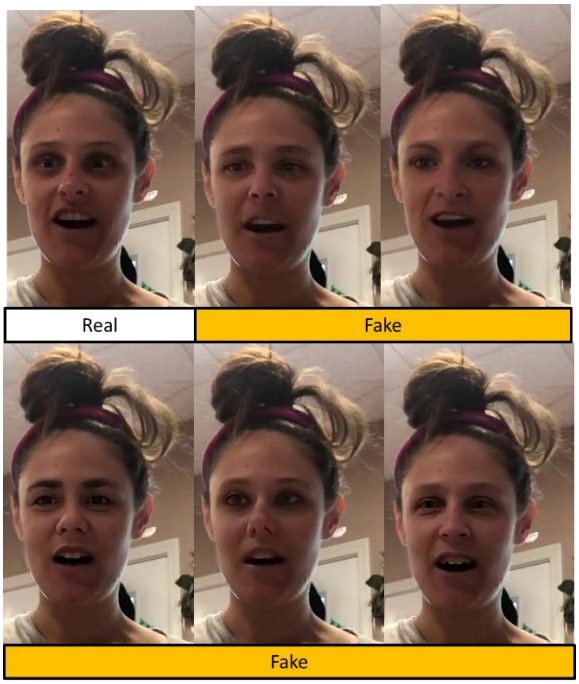


Figure 2: Real and fake images

Bayes Optimal Error

This is a significantly challenging task as human level performance is low at detecting deep fakes. From figure 4, can you recognize which image is fake? If you saw this image outside of a deep fake paper, would you ever be suspicious it isn't real? These factors lead to the assumption that Bayes Optimal Error is very high effectively capping the potential accuracy of a learning algorithm, and making learning effective parameters especially challenging. From a small trial, roughly 1 of 5 can be identified by a human.

For technical implementation challenges, see Appendix A.

5 Methods

5.1 Baseline Model Architecture – Xception Transfer Learning

Xception is the baseline model for the experiments[7], it stands for extreme inception based on the well-known Inception model pretrained on the ImageNet database. Transfer learning was used with three fully connected layers added to the end of the model.

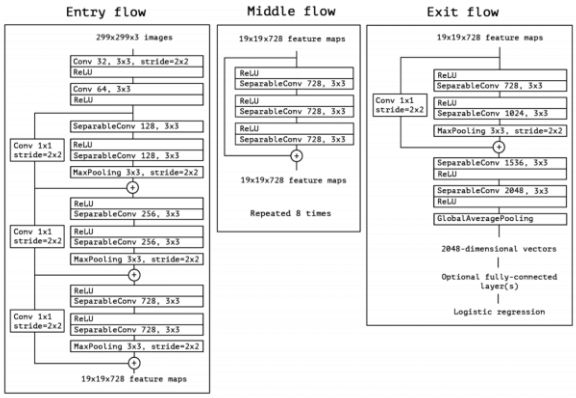
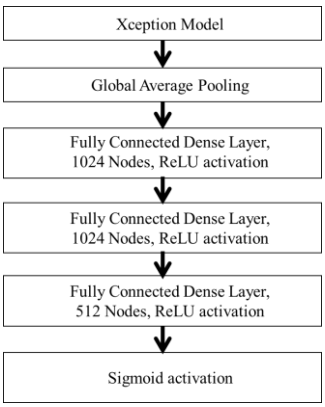


Figure 3 Xception Model Architecture [7] and Figure 4: Layers added to Xception Model for transfer learning baseline



The loss function is binary cross entropy, this is both a sensible choice as it is a binary classification task, but it is also dictated by the competition as the loss function to be used.

From the perspective of the DFDC the only metric is loss. From an intuitive perspective however, recall is likely the most important metric given a minimum level of precision. To align this project's work with the competition, loss will be the primary evaluation metric to determine the optimal model despite it not being the optimal real-world evaluation metric. Recall and precision will also be analyzed, but not from the perspective of selecting the best method. Solely for the method in which the loss function is adapted,

validation accuracy will be considered as the evaluation metric.

In the baseline, this model will be used with the normal images resized to fit more images into memory with a width and height of 200 pixels.

5.2 Model using Error Level Analysis

In the second experiment, the same baseline model will be used, but it will be fed in images that are made using Error Level Analysis [9]. Error Level Analysis is a process of resaving images and comparing their differences. This is a common method of detecting digital forgeries as a forged region of an image may have been saved differently than the original image.

Figure 5: Showing Image and Image in Error Level Analysis

5.3 Model using Non-Facial Feature Cropping

The third experiment will utilize images in which only the detected face from Haar Cascading [12] is shown and all other contents cropped out. The non-face cropping method proposed in this paper is a novel method. It is included as an experiment not because faces are more likely to reveal fake image data, but the most threatening deep fakes are manipulating human faces and a model focusing on those deep fakes will be more impactful than models detecting deep fakes from non-facial features.

Figure 6: Showing Image and Image with removal of contents not detected as a face

5.4 A Novel Conditional Loss Function

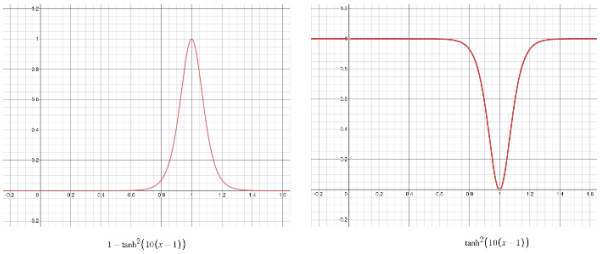


Figure 7: The first image showing a function that represents an "if" statement by which if x-1 = 0, the function is triggered. The second image shows an equation representing an "if not" function.

A method of creating a conditional loss function is proposed. This was done because the imbalanced dataset led the model to repeatedly always predict the image is fake rather than learning key features. A method is proposed to adapt the loss function if all predictions are "FAKE." Although there are other potentially methods, the loss function was chosen so it will be differentiable, represent an if statement, and punish the model with large weight updates if the model always predicts "FAKE." To do this, I use the $\tanh^2$ function added to traditional binary cross entropy as shown below.

$$\text{Loss} = \text{Loss}_{\text{Binary Cross Entropy}} + \text{Loss}_{\text{Conditional}}$$

$$\text{Loss}_{\text{Conditional}} = -[1 - \tanh^2(I(y\ \text{pred}, y\ \text{true}) * \tau)] * \Phi$$

- **I(y pred, y true), Iota – The condition to be triggered when Iota equals zero**
- **Φ, phi – punishment factor**
- **τ, tau – tightness factor**

Equation 1 for the conditional loss function I is a function of the predicted and true y values where if I is zero the condition will be met, τ is a non-trainable hyper parameter that tightens the "if" band, typically a larger value will be desired to tighten when the if statement is met, and Φ is the punishment factor that determines how strong the weights will update similar to a learning rate and is also a non-trainable hyper parameter.

For Equation 1, in this experiment Iota is set to being the difference between the sum of the predictions minus the length of the values, this means that if the length of the test values equals the number of predictions (therefore all predictions are "FAKE") then the conditional loss will be included in the loss function.

Several hyper parameters were tuned for this including the punishment factor and tightness factor.

5.5 Occlusion Sensitivity

Occlusion Sensitivity is also used as a subjective means to evaluate how learning is taking place.

```
def occlusion_sensitivity(model, X, patch_side_length):
    # function returns map of output based on occlusion
    # model - trained keras model
    # X - input square image data as a numpy array
    # patch_side_length - int, square side length of the patch to occlude from the image

    l = X.shape[0] # image length

    # define a mapped image of shape X and set to zero
    mapped = np.copy(X)
    mapped[:,:] = 0

    # iterate through each occlusion position
    for i in range(int(l/patch)):
        h_pos = i*patch
        for j in range(int(l/patch)):
            v_pos = j*patch

            # make a new image of shape X with the occluded region
            occluded = np.copy(X) # np.expand_dims(np.copy(x_train[0]), axis=-1)
            occluded[h_pos:h_pos+patch, v_pos:v_pos+patch] = 0
            occluded = np.expand_dims(occluded, axis=-1)

            # get a score from evaluating the model with the occluded image
            score = model.predict(occluded)

            # add the score to the mapped image
            mapped[h_pos:h_pos+patch, v_pos:v_pos+patch] = 255*score
    return mapped
```

Figure 8 Occlusion Sensitivity Code

6 Results

6.1 Comparing Data Processing Models – Validation Loss

The following graphs show the Xception model using the full image, the error level analysis image, and the face cropped image.

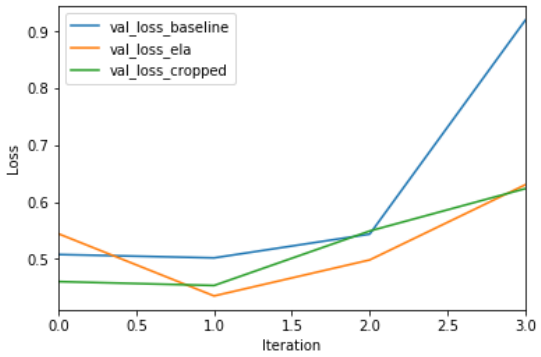


Figure 9 Validation Loss for different types of image processing

The validation loss performs worst with a typical full image, it performs noticeably better with both ELA and the non-facial features cropped. The novel cropping method performs slightly better than ELA.

6 .2 Comparing Conditional Loss Models Tuning Phi and Tau – Validation Accuracy

To evaluate the difference between conditional loss function hyper parameters, validation accuracy is the best evaluation metric because other evaluation metrics will not be effective in this situation. The loss function cannot be used as these are *different* equations, for example a punishment factor of 10000 will have a much higher loss than a punishment factor of 1 regardless of model performance, it is also very noisy. Precision and recall cannot be used as well as the model is always predicting 'FAKE' so all precision and recall values are the same and therefore cannot help draw any comparison between hyper parameters. The absence of better evaluation metrics makes validation accuracy the best choice.

By comparing validation accuracy, all values converge to the same results except for high phi.

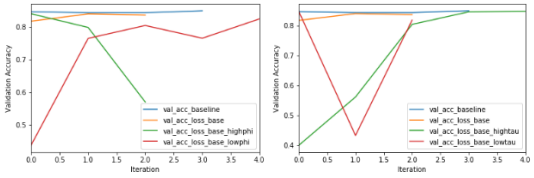


Figure 10 and Figure 11 Tuning Phi and Tau

6.3 Evaluation Metrics

Table 1 and 2 indicate learning is not taking place as they each should perfect recall and a precision equal to the percentage of Fake images in the dataset (84.86%).

	precision	recall
baseline full image	0.8486	1.0000
ELA	0.8486	1.0000
Non-facial feature cropped	0.8486	1.0000

	precision	recall
baseline loss function	0.8486	1.0000
high phi	0.8486	1.0000
low phi	0.8486	1.0000
high tau	0.8486	1.0000
low tau	0.8486	1.0000

Table 1 & 2

6.4 Interpreting the Model with Occlusion Sensitivity

Presently, the model is not learning and is only predicting each image as fake, we can see this from tables 1 and 2 as well as visually with occlusion sensitivity. The results on the MNIST data set are shown to highlight the occlusion sensitivity algorithm works.

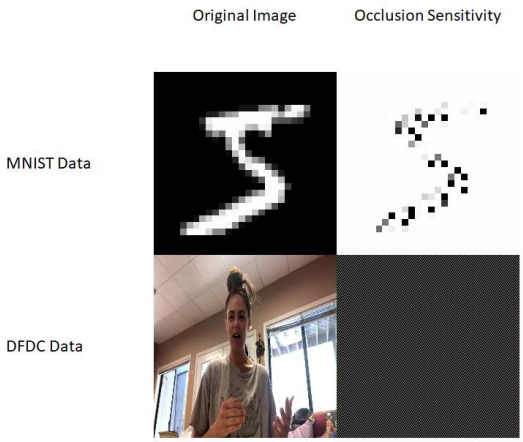


Figure 12 Showing Occlusion Sensitivity Comparing MNIST data and DFDC data

```
def occlusion_sensitivity(model, X, patch_side_length):
    # function returns map of output based on occlusion
    # model - trained keras model
    # X - input square image data as a numpy array
    # patch_side_length - int, square side length of the patch to occlude from the image

    l = X.shape[0] # image length

    # define a mapped image of shape X and set to zero
    mapped = np.copy(X)
    mapped[:,:] = 0

    # iterate through each occlusion position
    for i in range(int(1/patch)):
        h_pos = i*patch
        for j in range(int(1/patch)):
            v_pos = j*patch

            # make a new image of shape X with the occluded region
            occluded = np.copy(X) # np.expand_dims(np.copy(x_train[0]), axis=-1)
            occluded[h_pos:h_pos+patch, v_pos:v_pos+patch] = 0
            occluded = np.expand_dims(occluded, axis=-1)

            # get a score from evaluating the model with the occluded image
            score = model.predict(occluded)

            # add the score to the mapped image
            mapped[h_pos:h_pos+patch, v_pos:v_pos+patch] = 255*score
    return mapped
```

Figure 13 Occlusion Sensitivity Code Snippet

7 Future Work

Given more time, there would be many other research avenues to explore. Some areas to explore would be:

- **Padding the cropped non-facial features.** This may be helpful because the current model only crops the face, but by padding the non-cropped area, more context may be used while focusing on the face area.
- **Taking random patches of image data from the detected face, and a random patch of image data taken from the cropped features and comparing their distributions**. This may be helpful to compare pixel distribution in an area susceptible to deep fakes such as a face with an area less susceptible to deep fakes.
- **Exploring the audio data.** Perhaps in certain circumstances the sound may communicate information that may not match the visual information as an indicator of a forged image.
- **Combining vision models with sequential models.** This may work because deep fake pixel distributions may be less consistent when they are dynamic and in motion compared to when they are static.
- **Concatenate full images, with ELA images and cropped faces.** The combination of the image processing techniques to highlight certain features without removing other features such as the cropped features may prove an additional way to improve model performance.
- **Analysis of statistical distribution of image pixels.** This may help indicate a deep fake as forged pixels distributions may not match the distribution of pixels throughout the image.

These are some of many areas of future work for deep fake detection. Additional methods that can be suggested from the results are combining ELA and cropped images, or higher phi values and lower tau values.

Appendix A – Successful Implementation Tips

1. The dataset must be downloaded online, downloading directly to the cloud is not practical as most AWS instances won't be optimized for machine learning computing and be able to store the 470 Gb. I had to download all the data to my local machine, unzip it, and load it to an AWS S3 bucket, to be accessed by the AWS EC2 Instance.
2. Downloading one video at a time so as to not keep extra information in memory is important to do video processing as needed
3. To conserve instance memory, save small parts of the image that will be used in machine learning as directly as possible. An example might be save the numpy array of an image rather than the entire video or a pickle file of numerous arrays.
4. For my model, I used the first frame of each video for 10,000 videos.
5. For the labels, you must combine all the meta json files into a single file. This is over a Gb to look up the label for each video.
6. Training data can take a significant amount of time when you are downloading it mid training, and you will likely need to make manual early stopping functions.
7. To save time, I experimented with running epochs over each mini batch to avoid repeatedly downloading the same data as an effort to save time, but the loss functions were too noisy to make sense from.
8. You will likely have to iterate to find a stable batch size that can be held in memory.
9. Due to the 84%/16% break down of FAKE/REAL videos, for Keras model fit class_weight={1:.84,0:.16}
10. Keras Early Stopping was used as a call back for model fit with a patience of 2

References

1. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," pp. 1-9.
2. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J.Thies, and M. Niebner, "FaceForensics++: Learning to Detect Manipulated Facial Images," 2019, pp. 1-14.
3. Rossler, D. Cozzolino, L. Verdoliva, C. Riess, J. Thies, and M. Niebner, "FaceForensics: A Large-scale Video Dataset for Forgery Detection in Human Faces," 2018, pp. 1-21.
4. Bayar and M. Stamm, "A Deep Learning Approach To Universal Image Manipulation Detection Using A New Convolutional Layer," 2016, pp. 1-6.
5. Afchar, V. Nozick, J. Yamagishi, and I. Echizen, "MesoNet: a Compact Facial Video Forgery Detection Network," 2018.
6. Cozzolino, D. Gragnaniello, and L. Verdoliva, "Image forgery detection based on the fusion of machine learning and block-matching methods," 2013.
7. Chollet, "Xception: Deep Learning with Depthwise Separable Convolutions," 2017, pp. 1-8.
8. J. Fridrich and J. Kodovsky, "Rich Models for Steganalysis of Digital Images," vol. 7, 2012, pp. 868-882.
9. J. Muindi, R. Kosaraju, and Y. Lundia, "Deep Fake Detector: Using ML techniques to Distinguish Real Images from Fake," 2018, pp. 1-6.
10. K. He, X. Zhang, S. Ren, and J. Sun, "Deep Residual Learning for Image Recognition," 2015, pp. 1-12.
11. N. Rahmouni, V. Nozick, J. Yamagishi, and I. Echizen, "Distinguishing computer graphics from natural images using convolution neural networks," 2017, pp. 1-6.
12. P. Viola and M. Jones. “Rapid Object Detection Using a Boosted Cascade of Simple Features.” Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001, Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on, Computer Vision and Pattern Recognition 1, 2001.
13. C. Fontas, W. Li & E. Mendiola, “Realistic Image Synthesis and Classification,” 2018, pp. 1-6.