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

Joshua Denholtz
Stanford University
https://youtu.be/6GjHOjkG8Y4

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. Being able to detect forged digital media is the goal under investigation.

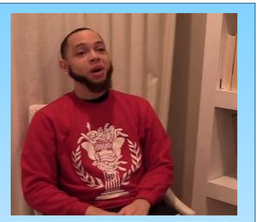
Related Work

- Rapid Object Detection using a Boosted Cascade of Simple Features[12]
- Realistic Image Synthesis and Classification[13]
- FaceForensics++: Learning to Detect Manipulated Facial Images[2, 3]

Dataset

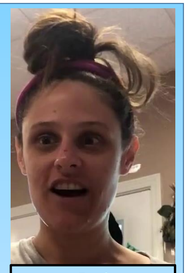
The dataset used for this project is the deep fake data provided by AWS, Facebook, Microsoft, the Partnership on AI's Media Integrity Steering Committee for their Deep Fake Detection Challenge (DFDC) hosted on Kaggle.

- Approximately 75,000(10s) mp4 files
- Labeled 'REAL' or 'FAKE.'
- The dataset is needs to be downloaded as 50 zip files of a total of 470 Gb.

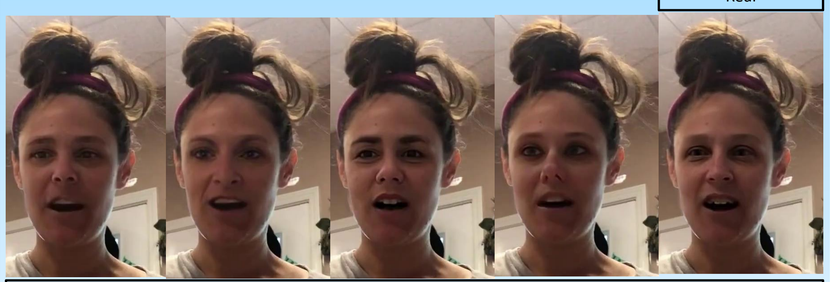


Challenges - Bayes Optimal Error

- Human level performance is low at detecting deep fakes, ~1 out of 5
- Can you recognize which are fake?
- Bayes Optimal Error is high making learning parameters challenging

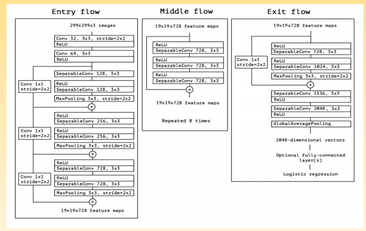


Real



Fake

Model Architecture



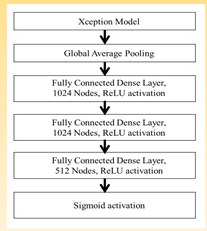


Image Processing Types

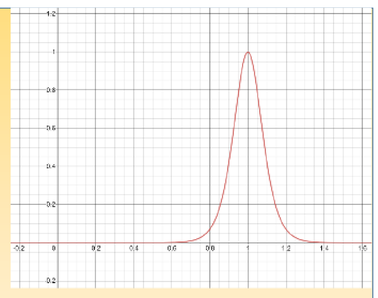
The models utilize the full image, error level analysis and an image with all non facial features cropped

Error Level Analysis

Non-Facial Feature Cropping

Conditional Loss

- A novel **conditional** loss is used with binary cross entropy.
- Here I is equal to the sum of preds minus the length of preds.
- If the model always predicts 1, a higher loss function triggered.

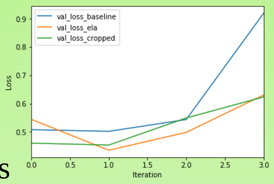


Loss = Loss Binary Cross Entropy + Loss Conditional

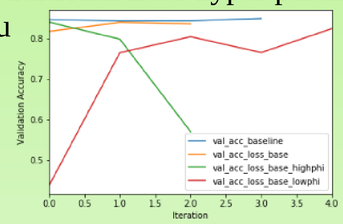
Loss Conditional = -[1 - tanh² (I(y pred, y true)*τ)] * Φ

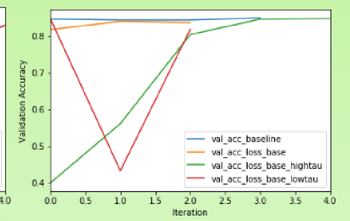
Results

Image Processing Methods comparing validation loss using binary cross entropy

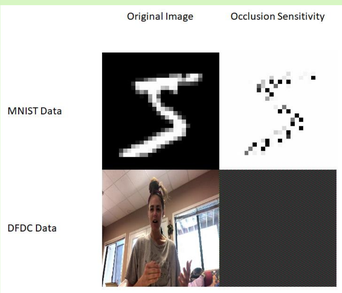


Tuning Conditional Loss hyper parameters phi and tau





Occlusion Sensitivity confirms evaluation metrics that learning was unsuccessful




```
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
```

Future Work

- **Padding the cropped non-facial features**
- **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**
- **Exploring the audio data**
- **Combining vision models with sequential models**
- **Concatenate full images, with ELA images and cropped faces**
- **Analysis of statistical distribution of image pixels**