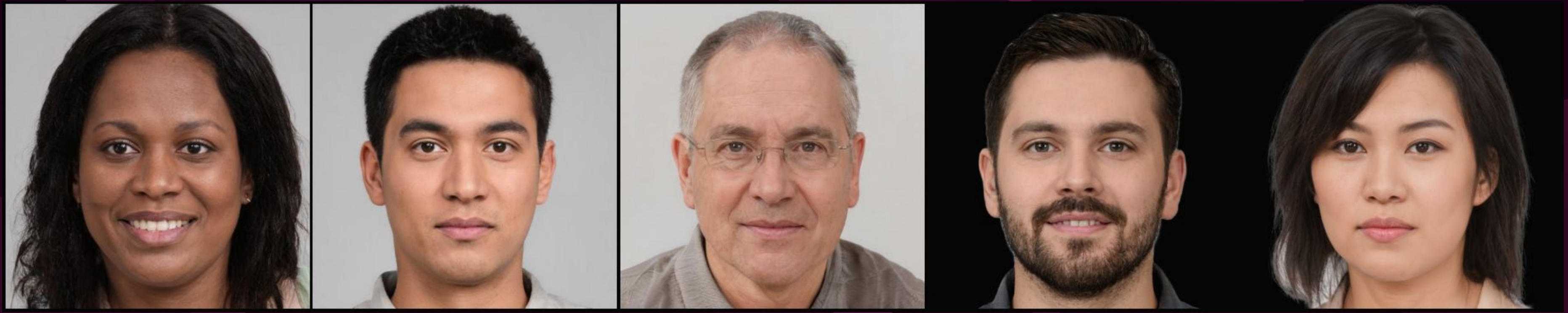
DEEPFAKE IMAGE DETECTION

BY

JEFF SPAGNOLA

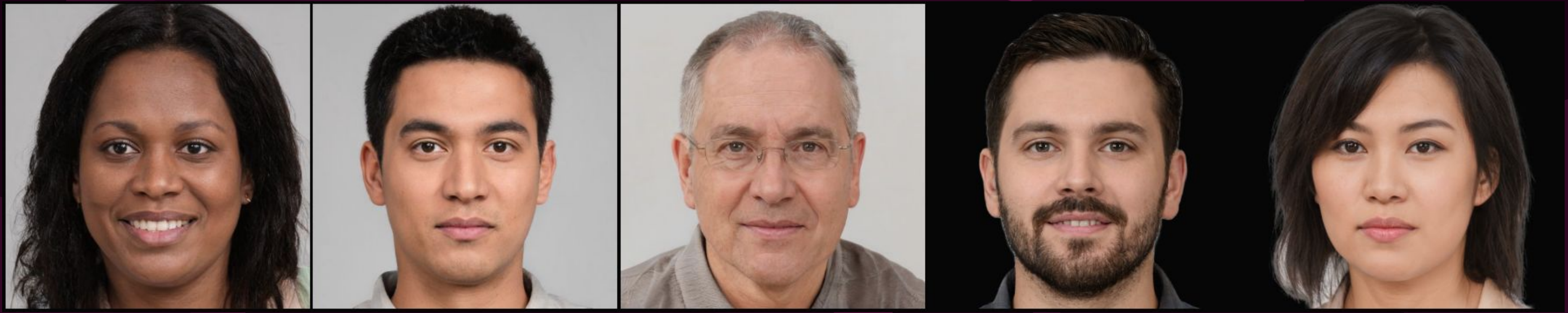
LET'S PLAY A GAME

Can you determine which of these images isn't real?



LET’S PLAY A GAME

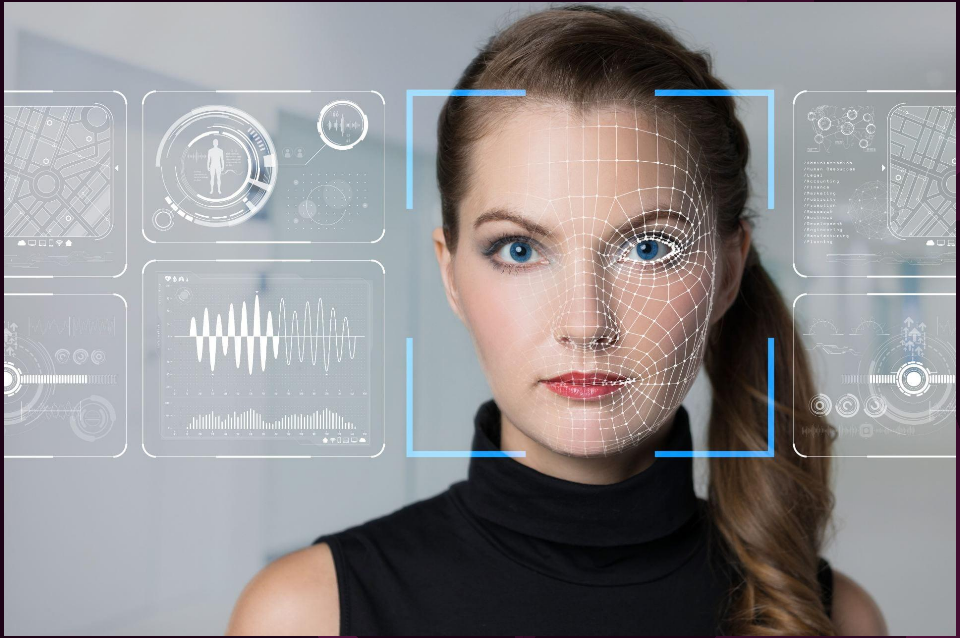
Can you determine which of these images isn’t real?



Trick question…NONE of them are real!

INTRODUCTION

- Image editing technology has improved to the point where it's nearly impossible to tell what's real.
- Deepfakes have already become a serious issue on social media, in politics, and in society at large.
- Goal of this project is to create a system that can tell the difference between a real image and a high quality deepfake.
- Deepfake Image Detection can be used in social media companies, security organizations, and news agencies.



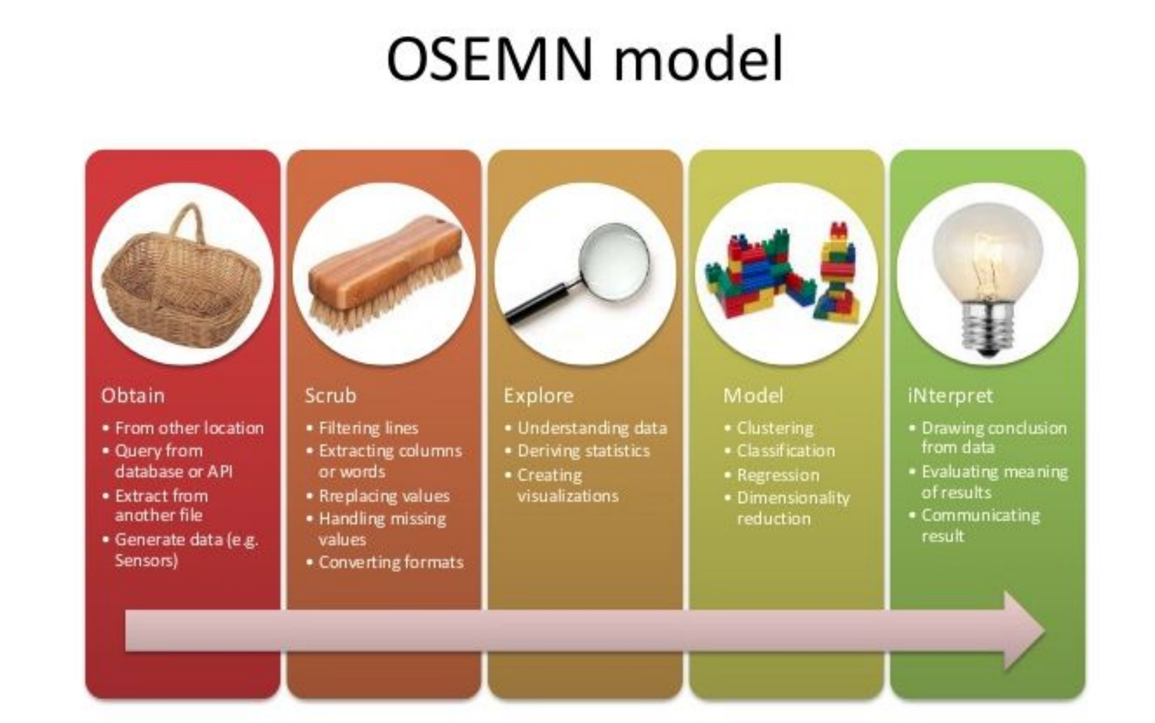
You can test out the Deepfake Detection App at the following URL:
<INSERT URL HERE>

1. THE PROCESS

Steps Taken in this Project

OSEMN PROCESS

Throughout this project, we will be following the OSEMN Data Science Process



OSEMN model

Obtain

- From other location
- Query from database or API
- Extract from another file
- Generate data (e.g. Sensors)

Scrub

- Filtering lines
- Extracting columns or words
- Rreplacing values
- Handling missing values
- Converting formats

Explore

- Understanding data
- Deriving statistics
- Creating visualizations

Model

- Clustering
- Classification
- Regression
- Dimensionality reduction

iNterpret

- Drawing conclusion from data
- Evaluating meaning of results
- Communicating result

2. THE DATA

Obtaining & Cleaning the Data

THE DATA

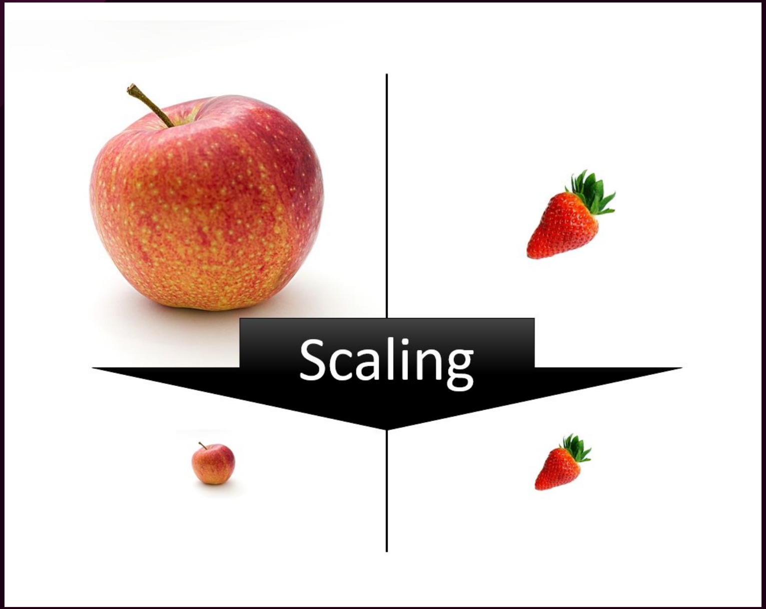
- Dataset of images was obtained by combining several collections of real and deepfake images.

- Criteria for the images is that they had to be high quality & at least 150px square.

- 142, 286 images in total.

- Nearly an equal distribution of real and deepfake images.

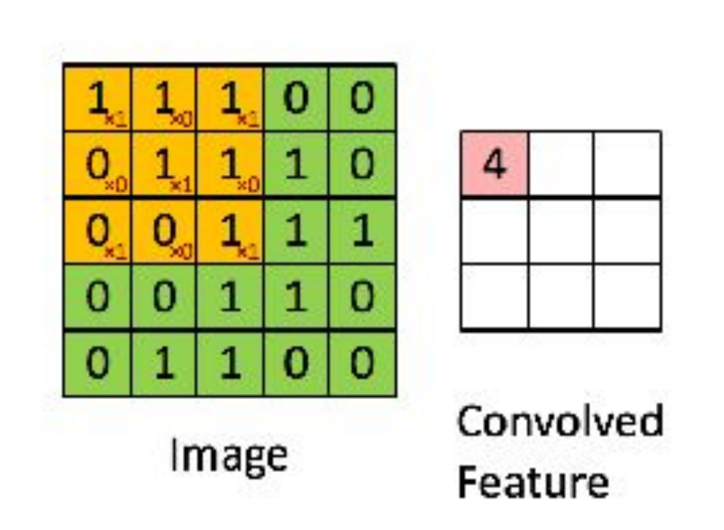
SCRUBBING



- Import the folders of images
- Rescale & Resize images
- Define the target classes:
  1 = real, 0 = deepfake
- Convert the image into an array (series of numbers)
- Create training, test, and validation sets for modeling

4. MODELING

CONVOLUTIONAL NEURAL NETWORK



$1_{\times 1}$	$1_{\times 0}$	$1_{\times 1}$	0	0
$0_{\times 0}$	$1_{\times 1}$	$1_{\times 0}$	1	0
$0_{\times 1}$	$0_{\times 0}$	$1_{\times 1}$	1	1
0	0	1	1	0
0	1	1	0	0

Image

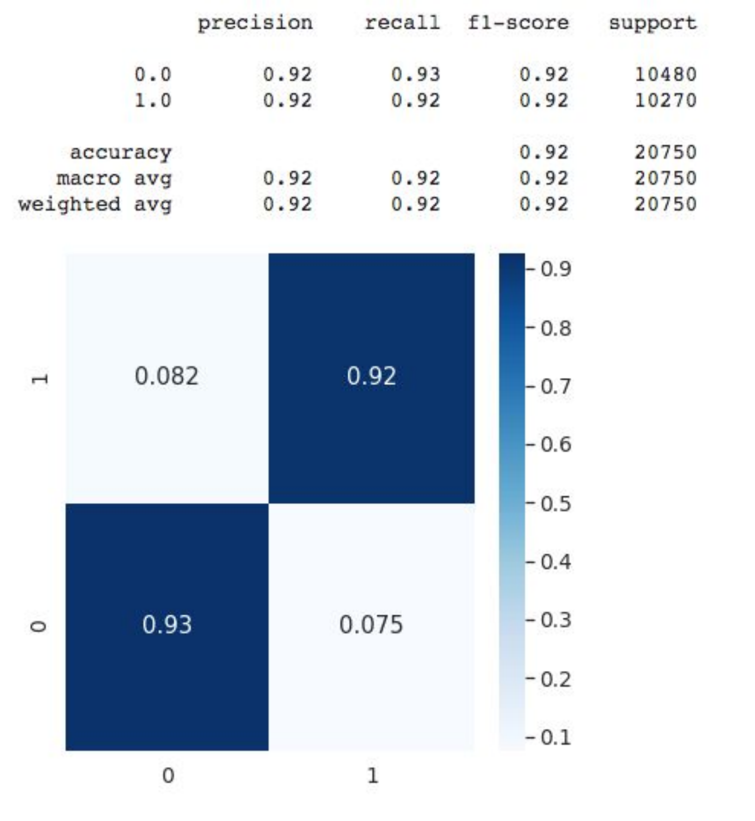
4		

Convolved Feature

- Image is scanned by a deep neural network
- Convolutional (Conv2d) layer analyzes groups of pixels in sequence
- Convolutional “weights” are fed into pooling layers, dense layers, and normalization layers.
- Experimented with a finely tuned CNN, pretrained CNN & an ensemble of both.

TUNED CNN

- Iterated through many combinations of layers and parameters.
- Able to achieve a 92% Accuracy
- Achieved 92% weighted Recall
- Faster training time than Pretrained CNN.



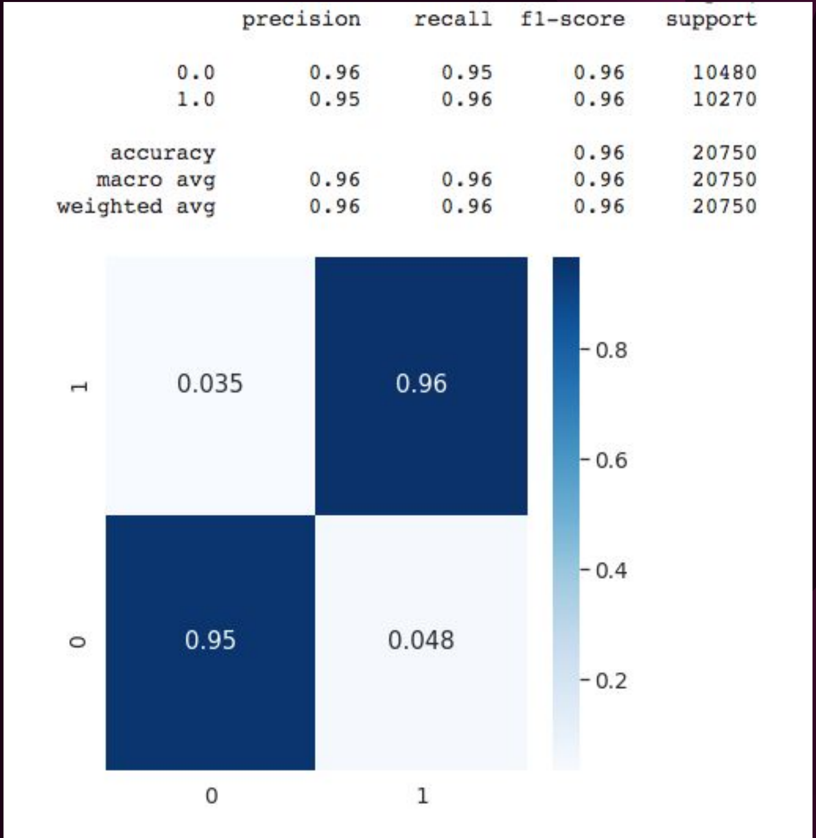
```
              precision    recall  f1-score   support

         0.0       0.92      0.93      0.92     10480
         1.0       0.92      0.92      0.92     10270

    accuracy                           0.92     20750
   macro avg       0.92      0.92      0.92     20750
weighted avg       0.92      0.92      0.92     20750
```

PRETRAINED CNN

- Used a pretrained CNN (Xception) as a convolutional base
- Able to achieve a 96% Accuracy
- Achieved 96% weighted Recall
- Slowest training time among the models we used.



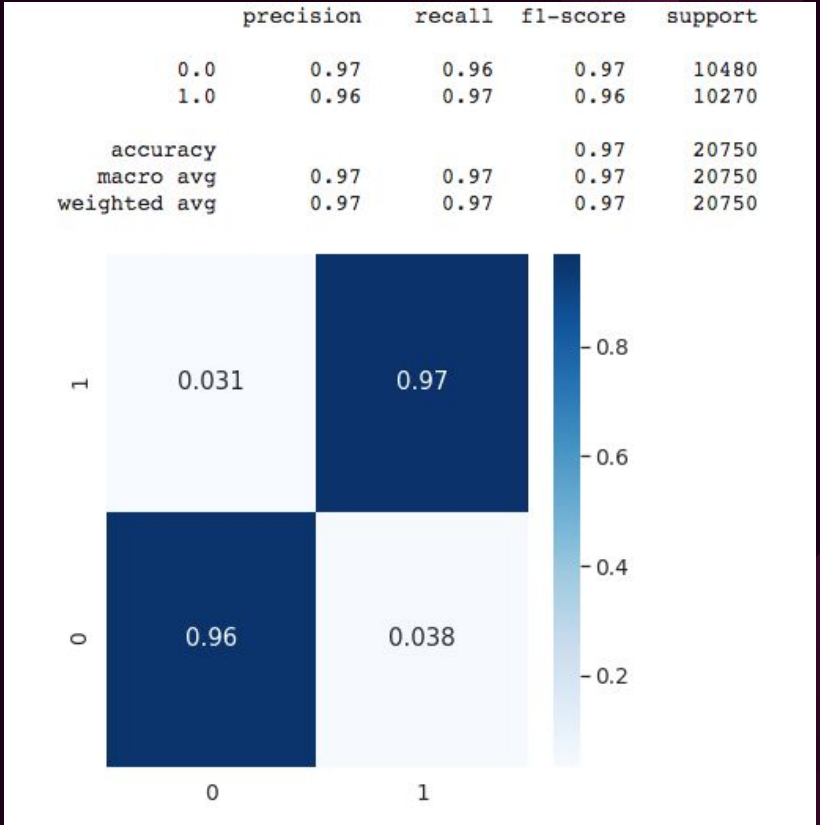
```
              precision    recall  f1-score   support

         0.0       0.96      0.95      0.96     10480
         1.0       0.95      0.96      0.96     10270

    accuracy                           0.96     20750
   macro avg       0.96      0.96      0.96     20750
weighted avg       0.96      0.96      0.96     20750
```

ENSEMBLE CNN

- Combined the layers of the Tuned CNN & Pretrained CNN into a new model.
- Able to achieve a 97% Accuracy
- Achieved 97% weighted Recall
- Slow training time plus this model required added processing of the data.



```
              precision    recall  f1-score   support

         0.0       0.97      0.96      0.97     10480
         1.0       0.96      0.97      0.96     10270

    accuracy                           0.97     20750
   macro avg       0.97      0.97      0.97     20750
weighted avg       0.97      0.97      0.97     20750
```

MODELS BY THE NUMBERS

	Accuracy	Weighted Recall	Training Time
Tuned CNN	92%	92%	1:16:48
Pretrained CNN	96%	96%	2:39:08
Ensemble CNN	97%	97%	0:27:28

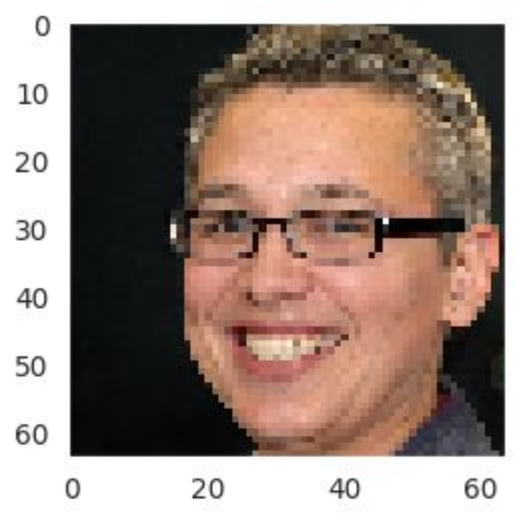
The **pretrained & ensemble CNNs both had higher scores** but the training & **loading time of the pretrained CNN makes it a difficult choice** for deployment for the Deepfake Detection App. The **ensemble model requires an extra step in preprocessing** and does add a bit of loading time in the final app. This is something we're **still experimenting** with.

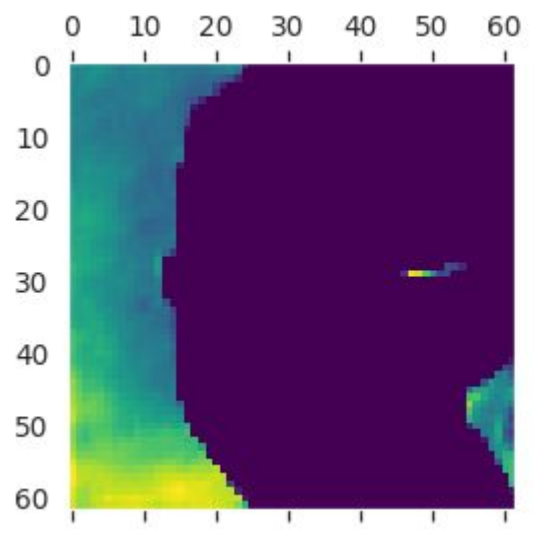
5. INTERPRET

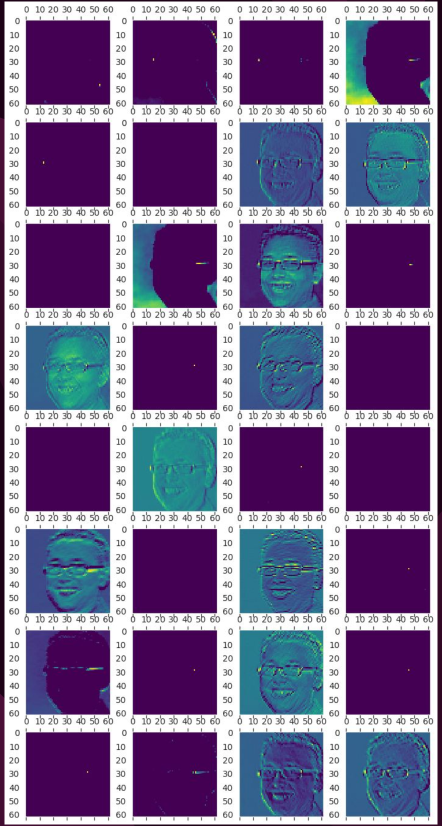
What have we learned?

HOW DOES THIS WORK?

Earlier, we mentioned the CNN works by scanning layers. Below is an original image, a single layer of a CNN and multiple layers of a CNN.

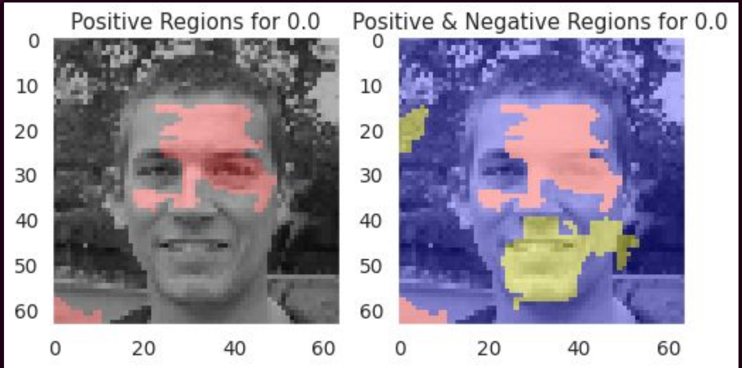


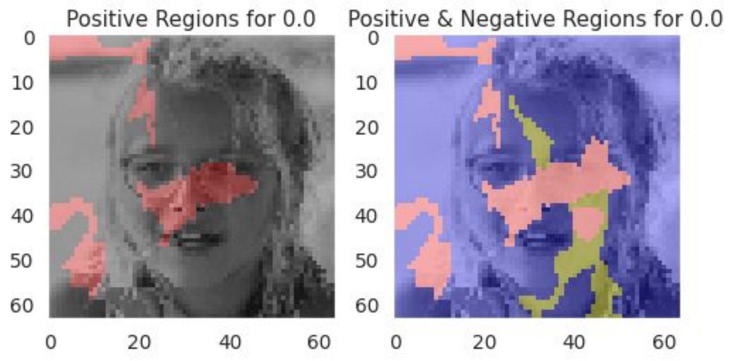


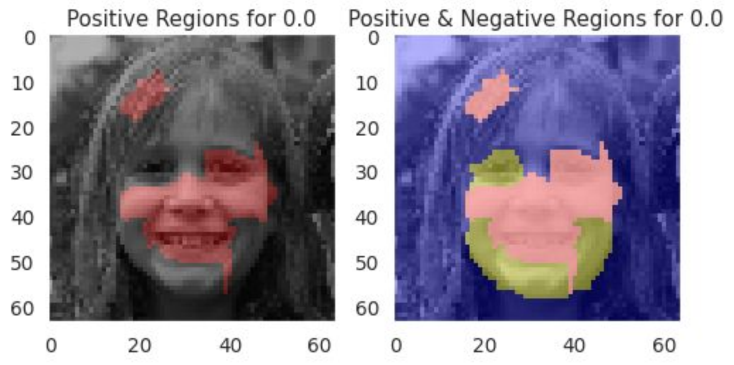


LIME EXPLAINER

The Lime Package gives us additional insight into how the model is making predictions. We can see that the model seems to "notice" the area around the eyes.







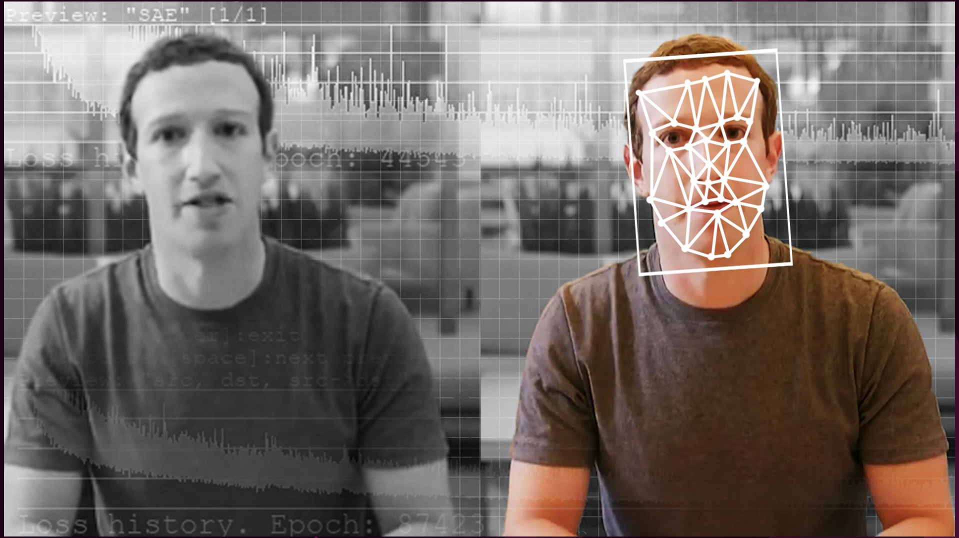
RESULTS

- Tuned CNN - 92% accuracy
  Pretrained - 96% accuracy
  Ensemble - 97% accuracy

- Pretrained networks are a valuable tool, but only when retraining parameters.

- Ensemble model is fastest & most accurate, but requires the training time of previous models.

- Lime Explainer shows that eyes are a focal point for making a prediction.



THE APP

Check out the functionality of the app here <insert URL>

6. RECOMMENDATIONS

How to Proceed

RECOMMENDATIONS

- For pretrained networks, retrain the parameters
- Use an ensemble of tuned CNN & Pre-trained CNN for for highest accuracy
- For model deployment, use a finely tuned CNN for speed & solid accuracy
- The Deepfake Image Detection App is recommended for Social Media Companies to weed out bots.

FUTURE WORK

More Data

Add additional images to the dataset. More data = higher accuracy.

More User-Friendly

Update the associated app to be more user-friendly and have more image classification features.

Expand the Scope

Modify the app to be able to scan for video files as well as images..

Adapt for Poor Quality

Expand the capability of the model to account for poor quality fake images.

THANKS!

Any questions?

@spags093
jeff.spags@gmail.com