

Facial Recognition of Identical Twins using Convolutional Neural Networks

Springboard Capstone #2
Anthony Johnson



Outline

- Motivation
- Model
- Data
- Training
- Interpretation
- Conclusions

My brother



Me



Motivation

- Facial recognition software has come a long way in the last few years, thanks to **convolutional neural networks (CNNs)**
 - iPhone X uses your face to unlock your phone
- How good exactly are CNNs?
- On a number of occasions I've been able to unlock my identical twin brother's iPhone with my face.
- Anything less than 100% accuracy is a major concern

Model

- **VGG-Face** model [1]
 - Open-source CNN
 - Pre-trained on hundreds of thousands of faces

Model Adaptation

- Define function

```
instantiate_model(num_dense, num_trainable)
```

- Creates same architecture as VGG-Face
- Load pre-trained weights, vgg_face_weights.h5
- Remove final layer
- Add a Dense layer with num_dense = 100 nodes
- Add a Dense Activation('softmax') layer with 2 nodes
- Set all but the last num_trainable = 5 layers as untrainable
- Input hyperparameters num_dense and num_trainable

Model Architecture

- 145M parameters
 - 11M trainable
 - 134M untrainable

	Layer (type)	Output Shape	Param #
1)	zero_padding2d_131_input	(In (None, 224, 224, 3)	0
2)	zero_padding2d_131	(ZeroPadd (None, 226, 226, 3)	0
3)	conv2d_161	(Conv2D) (None, 224, 224, 64)	1792
4)	zero_padding2d_132	(ZeroPadd (None, 226, 226, 64)	0
5)	conv2d_162	(Conv2D) (None, 224, 224, 64)	36928
6)	max_pooling2d_51	(MaxPooling (None, 112, 112, 64)	0
7)	zero_padding2d_133	(ZeroPadd (None, 114, 114, 64)	0
8)	conv2d_163	(Conv2D) (None, 112, 112, 128)	73856
9)	zero_padding2d_134	(ZeroPadd (None, 114, 114, 128)	0
10)	conv2d_164	(Conv2D) (None, 112, 112, 128)	147584
11)	max_pooling2d_52	(MaxPooling (None, 56, 56, 128)	0
12)	zero_padding2d_135	(ZeroPadd (None, 58, 58, 128)	0
13)	conv2d_165	(Conv2D) (None, 56, 56, 256)	295168
14)	zero_padding2d_136	(ZeroPadd (None, 58, 58, 256)	0
15)	conv2d_166	(Conv2D) (None, 56, 56, 256)	590080
16)	zero_padding2d_137	(ZeroPadd (None, 58, 58, 256)	0
17)	conv2d_167	(Conv2D) (None, 56, 56, 256)	590080
18)	max_pooling2d_53	(MaxPooling (None, 28, 28, 256)	0
19)	zero_padding2d_138	(ZeroPadd (None, 30, 30, 256)	0
20)	conv2d_168	(Conv2D) (None, 28, 28, 512)	1180160
21)	zero_padding2d_139	(ZeroPadd (None, 30, 30, 512)	0
22)	conv2d_169	(Conv2D) (None, 28, 28, 512)	2359808
23)	zero_padding2d_140	(ZeroPadd (None, 30, 30, 512)	0
24)	conv2d_170	(Conv2D) (None, 28, 28, 512)	2359808
25)	max_pooling2d_54	(MaxPooling (None, 14, 14, 512)	0
26)	zero_padding2d_141	(ZeroPadd (None, 16, 16, 512)	0
27)	conv2d_171	(Conv2D) (None, 14, 14, 512)	2359808
28)	zero_padding2d_142	(ZeroPadd (None, 16, 16, 512)	0
29)	conv2d_172	(Conv2D) (None, 14, 14, 512)	2359808
30)	zero_padding2d_143	(ZeroPadd (None, 16, 16, 512)	0
31)	conv2d_173	(Conv2D) (None, 14, 14, 512)	2359808
32)	max_pooling2d_55	(MaxPooling (None, 7, 7, 512)	0
33)	conv2d_174	(Conv2D) (None, 1, 1, 4096)	102764544
34)	dropout_21	(Dropout) (None, 1, 1, 4096)	0
35)	conv2d_175	(Conv2D) (None, 1, 1, 4096)	16781312
36)	dropout_22	(Dropout) (None, 1, 1, 4096)	0
37)	conv2d_176	(Conv2D) (None, 1, 1, 2622)	10742334
38)	flatten_11	(Flatten) (None, 2622)	0
39)	dense_21	(Dense) (None, 100)	262300
40)	dense_22	(Dense) (None, 2)	202

Total params: 145,265,380

Trainable params: 11,004,836

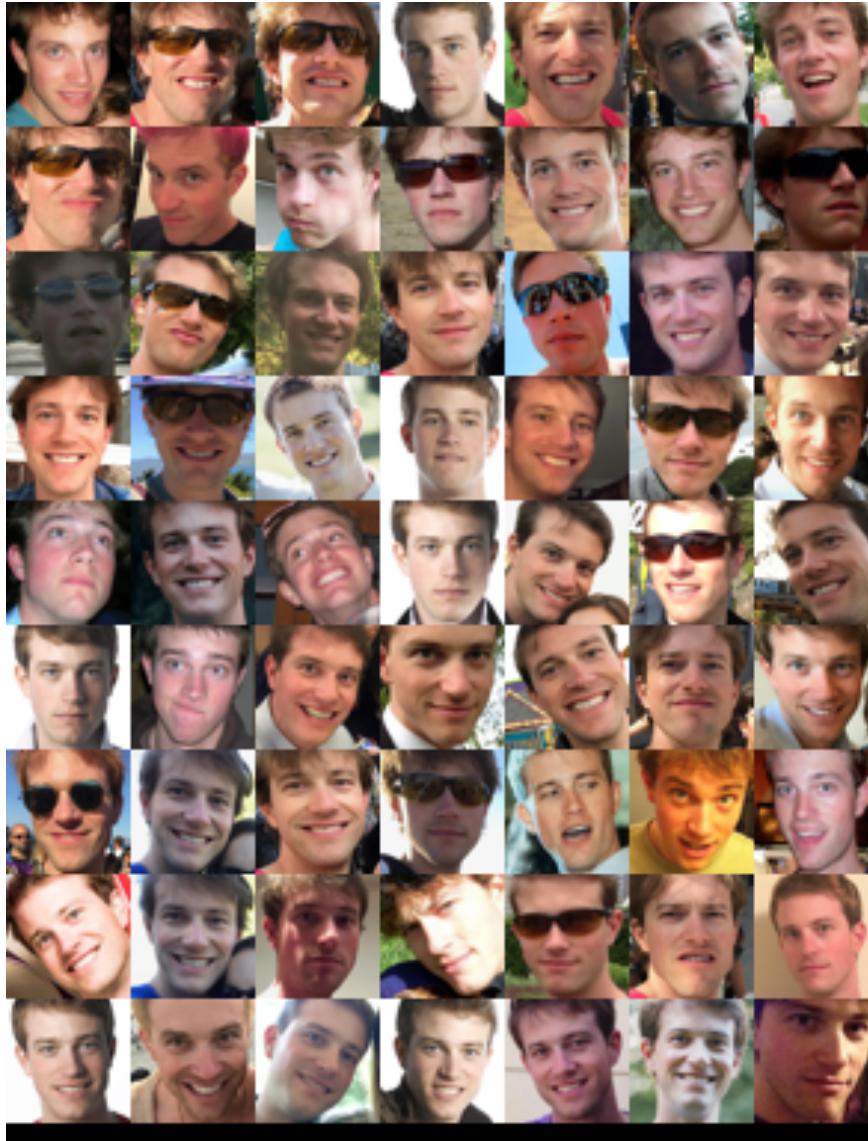
Non-trainable params: 134,260,544

Data

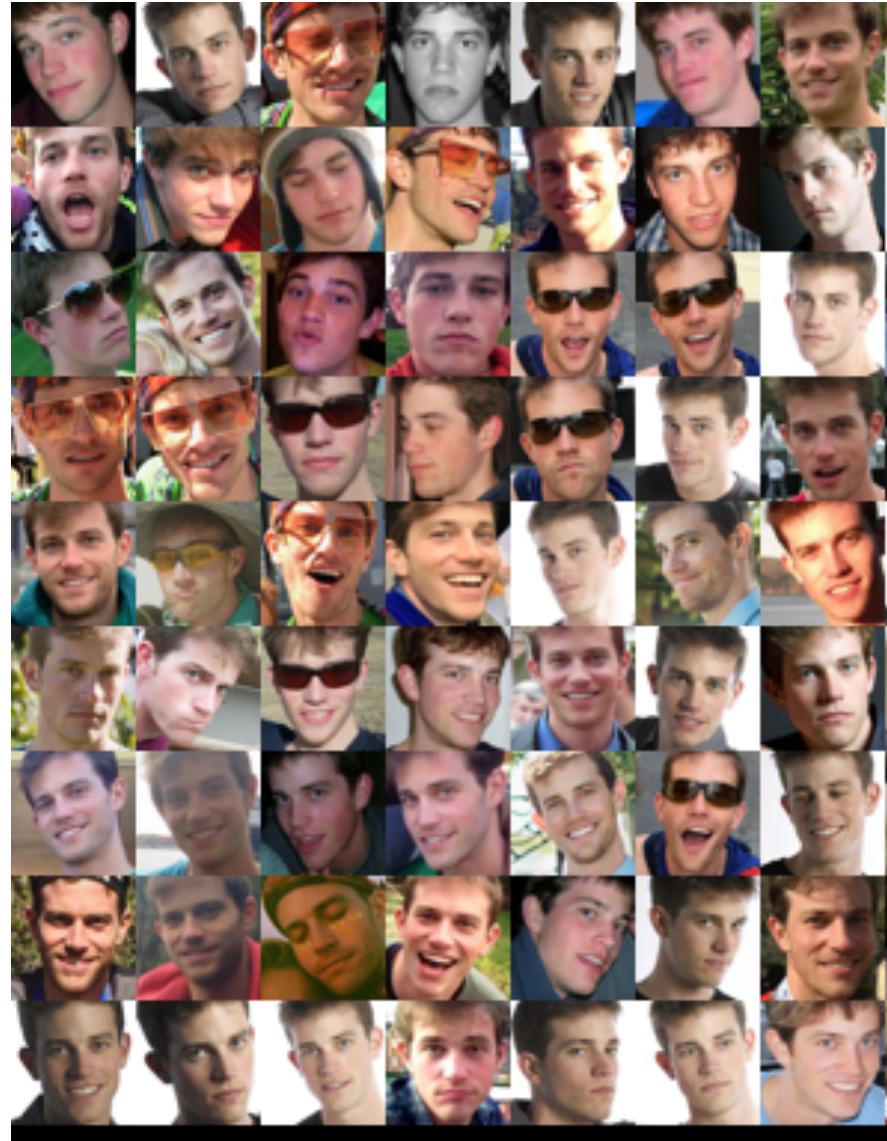
- I gathered digital photographs of me and my brother
- Square cropped each image
- Split into 3 datasets
 - Training = 118 images per twin
 - Validation = 27 images each per twin
 - Testing = 33 images each per twin
- Define **getXY()** function
 - Resizes images to 224x224
 - Stacks them together into trainX, valX, testX
 - Categorical dependent variable trainY, valY, testY

Sample Training Images

Anthony



Paul

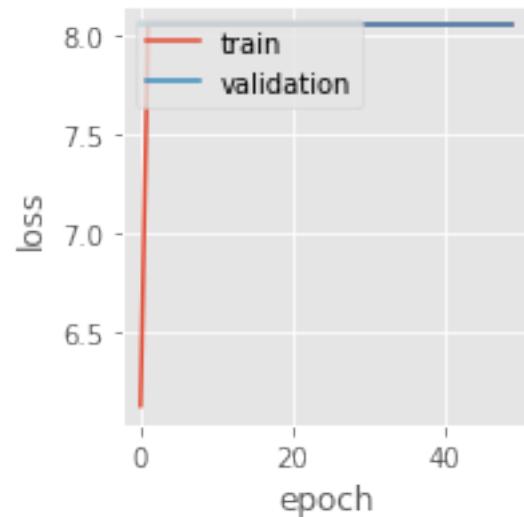
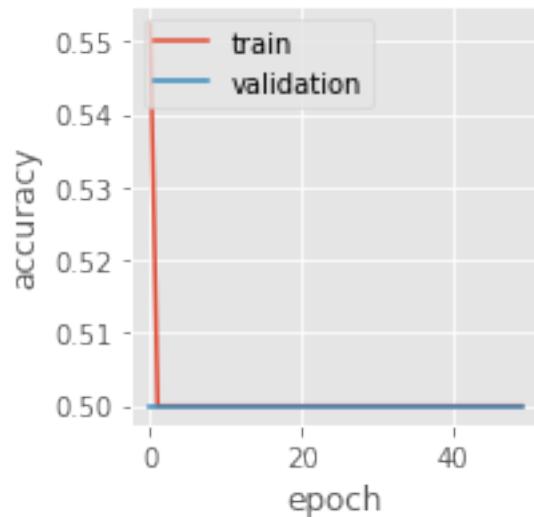


Training

- Hyperparameters/settings to tune:
 - The number of layers that are trainable, **num_trainable**
 - The learning rate, **lr**
 - The number of nodes, **num_dense**, in the penultimate layer
 - The decay rate, **decay**, of the learning rate
 - The optimizer: SGD vs. Adam

Training – Iteration #1

- Varied **num_trainable** = [3, 4, 5, 6, 7, 8, 9]
- 120 nodes in the penultimate layer
- SGD optimizer with
 - Learning rate = 0.1 (Decay = 0)
 - Loss = ‘categorical_crossentropy’
 - Metrics = ‘accuracy’
- These runs suffered from overfitting and thus did not converge.



Training – Iteration #2

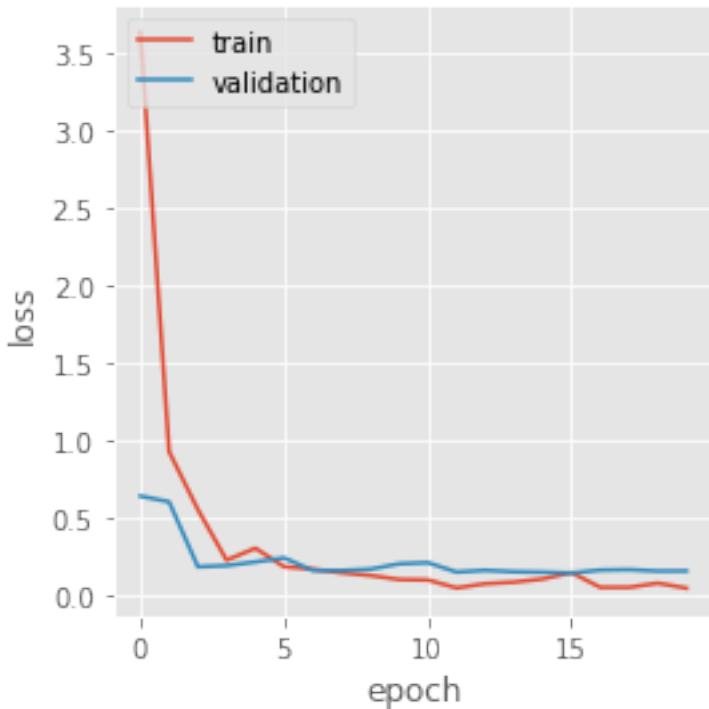
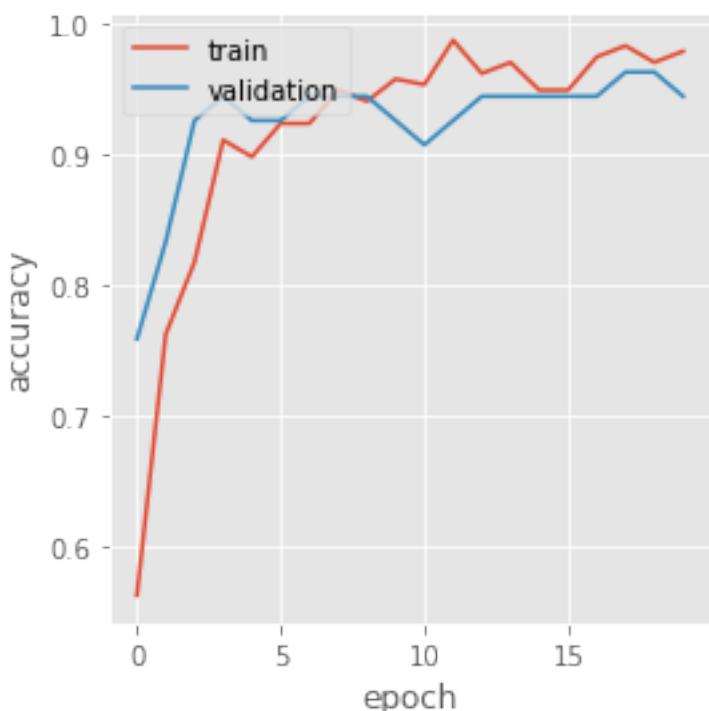
- Varied:
 - **num_trainable** = [3, 4, 5, 6, 7, 8, 9]
 - Learning rate **lr** = [0.001, .005, .01, .05, .1, 1]
 - **num_dense** = [10, 50, 100, 200]
- EarlyStopping condition
 - Patience of 4
 - Monitoring **val_loss**
- These runs converged!
 - Training accuracies 90%-100%
 - Validation accuracies 85%-95%

Training – Iteration #3

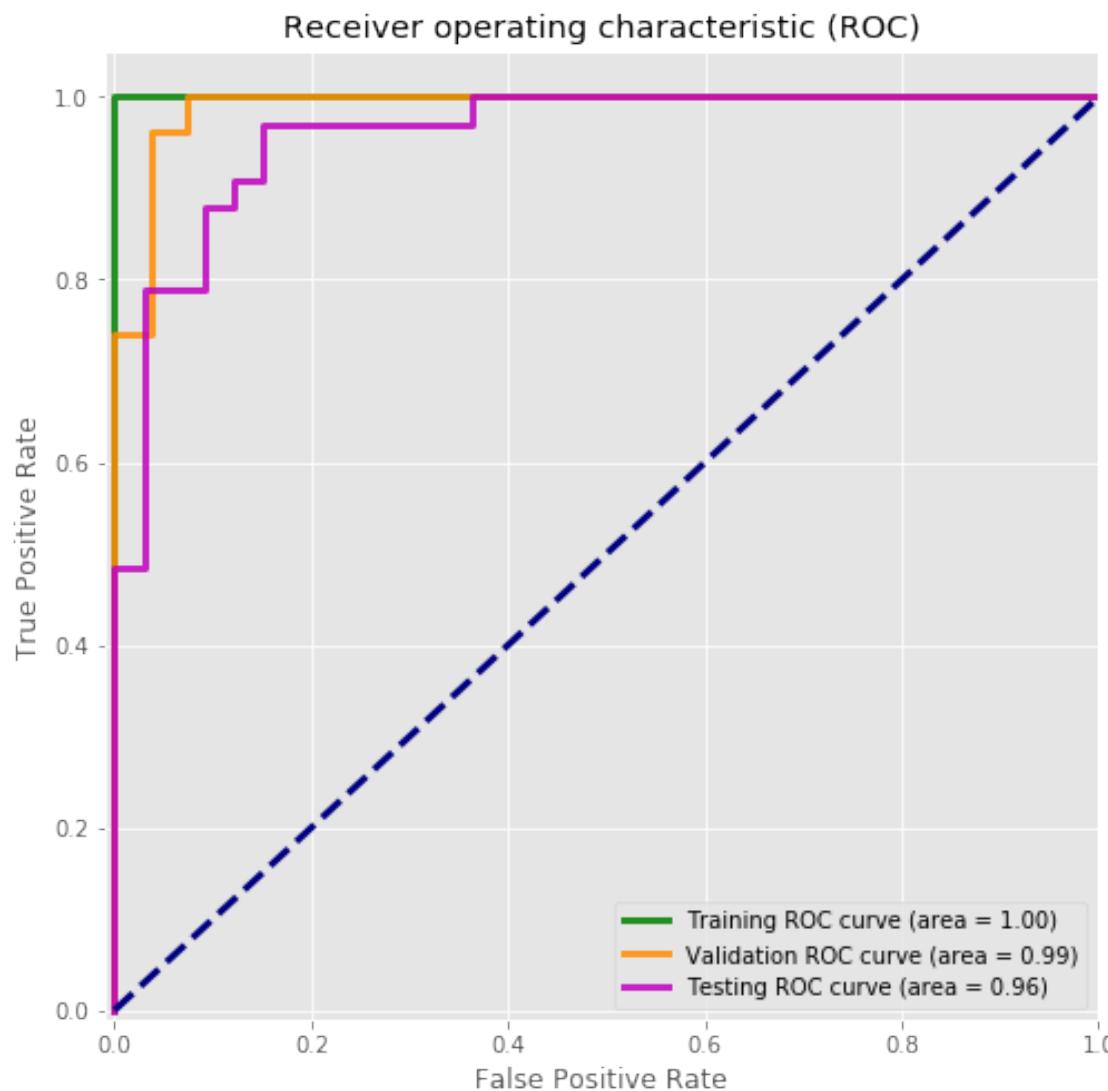
- Varied: Same as previous but varied decay
 - Decay = [.000005,.00001,.00002,.00004]
- EarlyStopping condition
 - Increased patience to 6
 - Monitoring **val_loss**
- These runs also converged!
 - Training accuracies 90%-100%
 - Validation accuracies 85%-95%

Best Model

- Parameters:
 - Num_dense = 100
 - Num_trainable = 5
 - Learning rate = 0.001
 - Decay = 0
 - Optimizer = SGD
 - EarlyStopping patience of 6 epochs
- Model accuracy:
 - Training: 100%
 - Validation: 94%
 - Testing: 88%



Best Model – ROC Curve



Best Model – Testing Results

TESTING SET	PREDICTED - AJ	PREDICTED - PJ
ACTUAL - AJ	29	4
ACTUAL - PJ	4	29

Actual = AJ, Predicted = PJ



Potential causes of these misclassifications:

- Bad Lighting
- Sunglasses
- Unusual facial expressions

Actual = PJ, Predicted = AJ



Interpretation

- **LIME** = Local Interpretable Model-Agnostic Explanations [2]
 - *LimeImageExplainer()* identifies features in an image that increase the probability (green) or decrease the probability (red) of a certain class
 - Red patches indicate a higher probability of Paul and green patches indicate a higher probability of Anthony

Paul

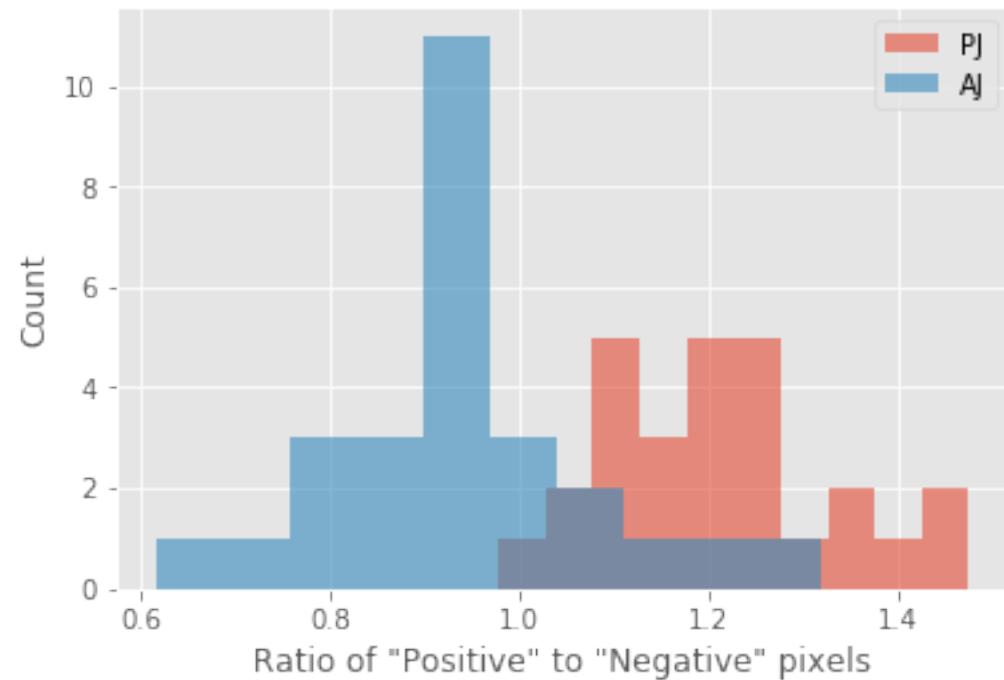


Anthony



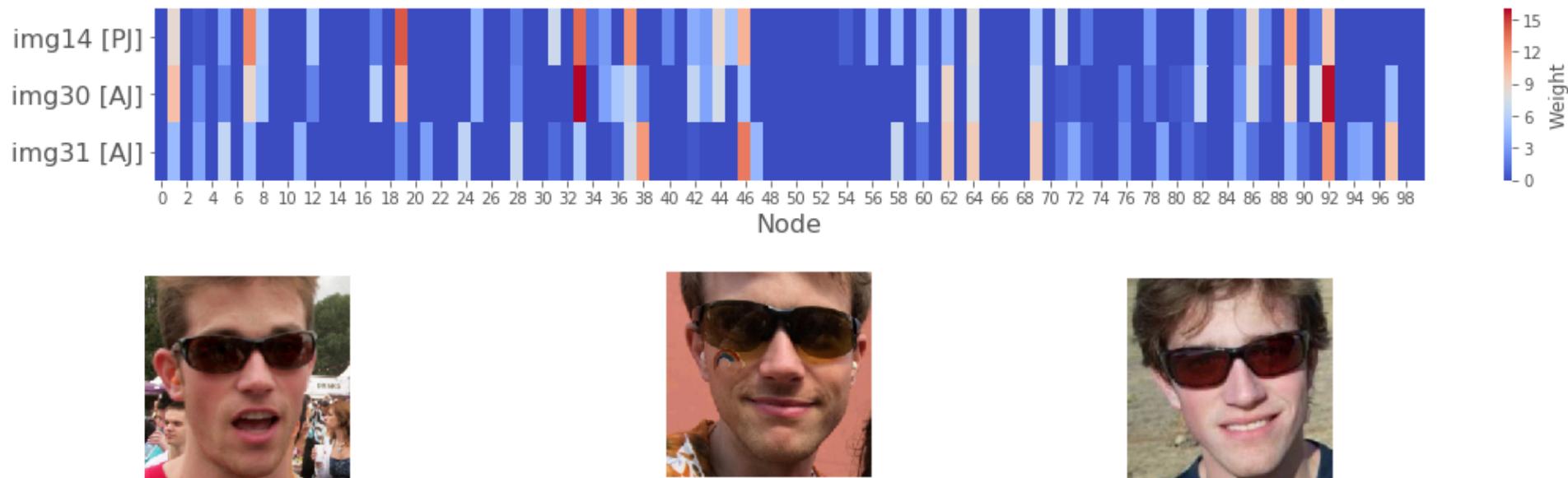
Interpretation

- Analysis:
 - Sum the total number of green and red pixels for both AJ and PJ
 - Divide total “positive” pixels by total “negative” pixels
 - Create histogram (right)
- In PJ images, the ratio of red to green pixels was between 1.0 and 1.5, averaged to 1.21.
- In AJ images, the equivalent ratio varied much more widely, between 0.6 and 1.3.



Direct Face Comparison

- Penultimate layer node weights
 - Middle row is misclassified



Img14 [PJ]



Img30 [AJ]
[thinks it is PJ]



Img31 [AJ]

Conclusions

- The final convolutional neural network:
 - Optimized the learning rate (.001), number of trainable layers (5), and number of nodes in the final layers (100)
 - Converged in <5 epochs
 - Final testing accuracy = 88%
 - Final testing ROC AUC = 0.96
- Overall, the model failed in cases with bad or inconsistent lighting or sometimes when we were wearing sunglasses
- Model seems to have slightly easier time recognizing PJ than AJ
 - Maybe PJ images are "better" by chance

Future Work

- While decent for identical twins, this model would be unacceptable in a security application
- To improve the model:
 - Train more images (the more the better)
 - Further optimize the hyperparameters
 - Optimize the optimizer
- I would like to dig deeper into interpreting and visualizing CNNs
 - With millions of parameters, CNNs are difficult to interpret
 - But this is still a new field, and anything is possible