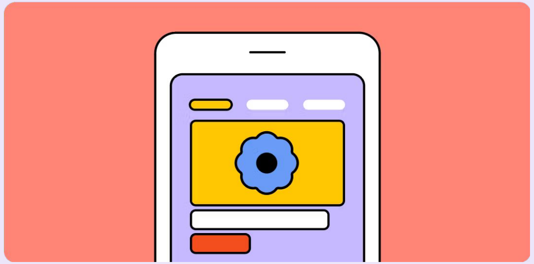
GAZE DETECTION

Achinthya Sreedhar, Aryan Sehgal,
Barrett Ratzlaff, Neha Shastri

Gaze Detection has utility in many spaces

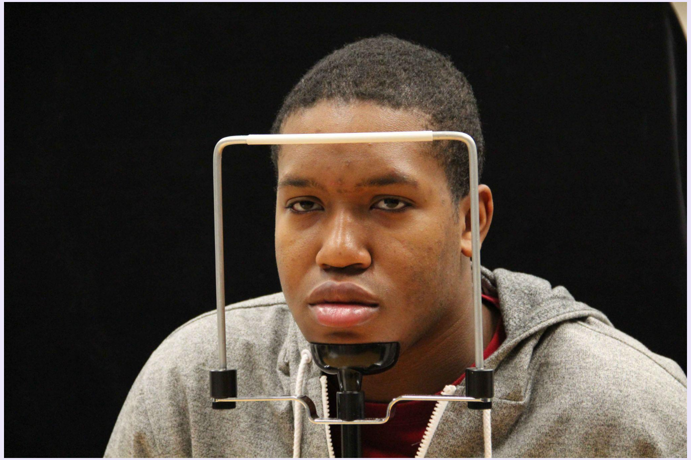


Each application has a different range of interest for a user's gaze

For the scope of our project, we chose a binary label classification: Is the user looking directly at the camera, or not?

Our dataset comes from Columbia University, in which there are 5,880 headshot photos of 56 diverse subjects taken at various angles.

Example:



Our dataset has a significant class imbalance, with ~720 photos of subjects looking at the camera, and the remaining 5,000 photos in the null class based on our initial standard.

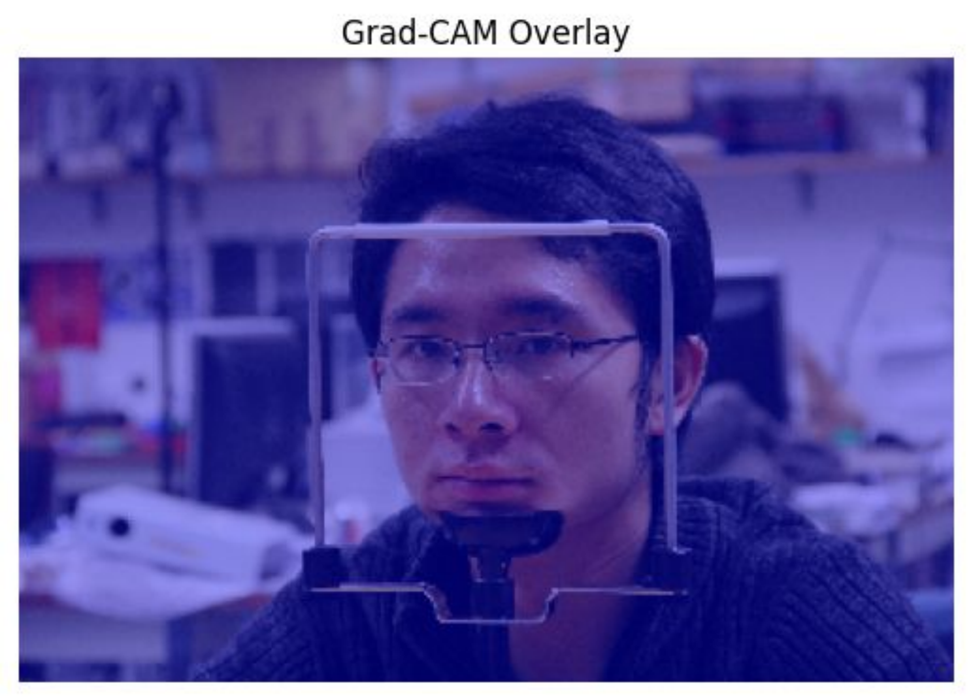
After layers to resize abnormal images to a standard size, our initial model had three convolutional layers, increasing in filters with each layer. Between each Conv2D layer was a MaxPooling layer.

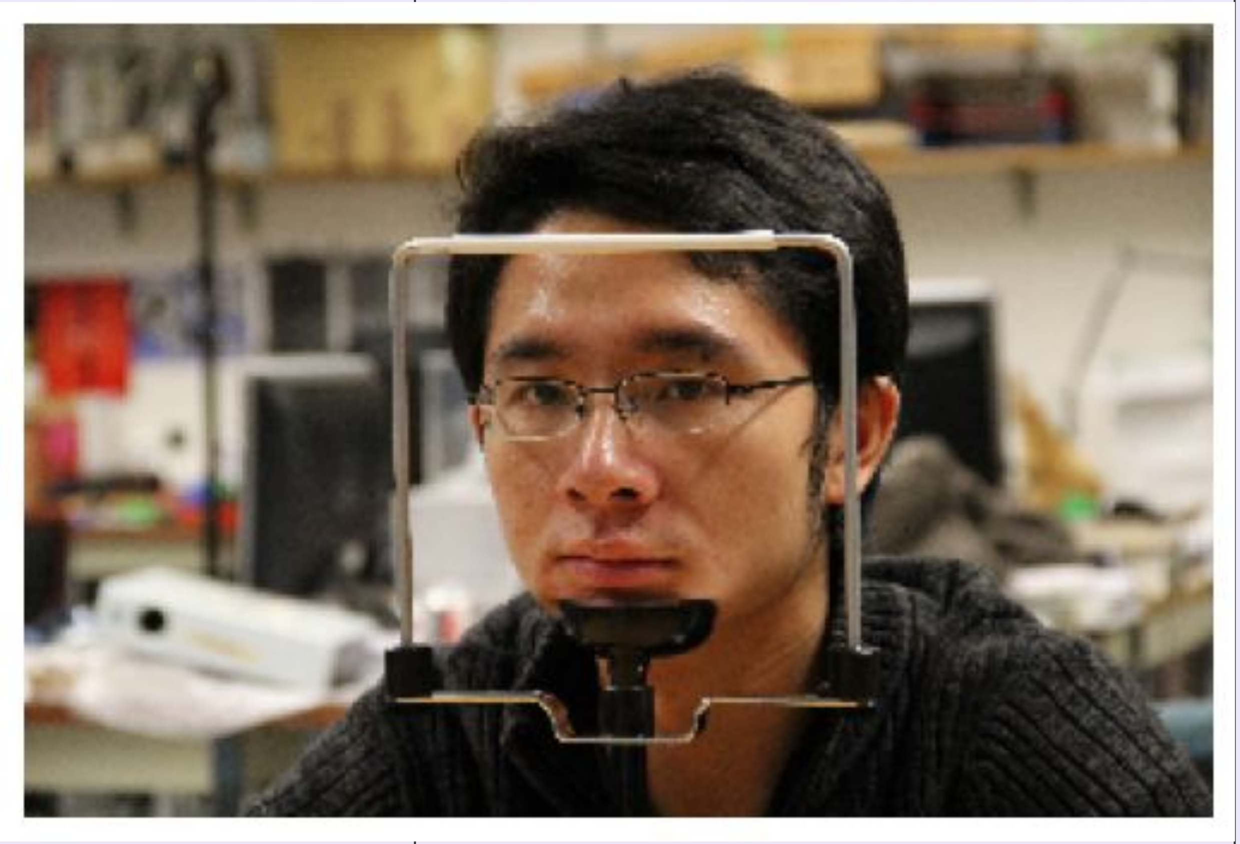
```
inputs = Input(shape=(None, None, 3))
resized = Resizing(200, 300)(inputs)  # making sure model is applicable beyond init
rescaled = Rescaling(1./255)(resized)  # scaling pixel values from 0 to 1
first_layer = Conv2D(filters=16, kernel_size=(2,2), activation='relu')(rescaled)  #
first_pool = MaxPooling2D(pool_size=2)(first_layer)  # 2x2 pixel pool. This could be
third_layer = Conv2D(filters=32, kernel_size=(2,2), activation='relu')(first_pool)
third_pool = MaxPooling2D(pool_size=2)(third_layer)
fifth_layer = Conv2D(filters=128, kernel_size=(2,2), activation='relu')(third_pool)
fifth_pool = MaxPooling2D(pool_size=2)(fifth_layer)
flatten_out = Flatten()(fifth_pool)

hidden_layer = Dense(128, activation='relu')(flatten_out)
outputs = Dense(1, activation = 'sigmoid')(hidden_layer)
model = Model(inputs=inputs, outputs=outputs)
```

After training, we found that our model had ~87% accuracy on the validation set! However…

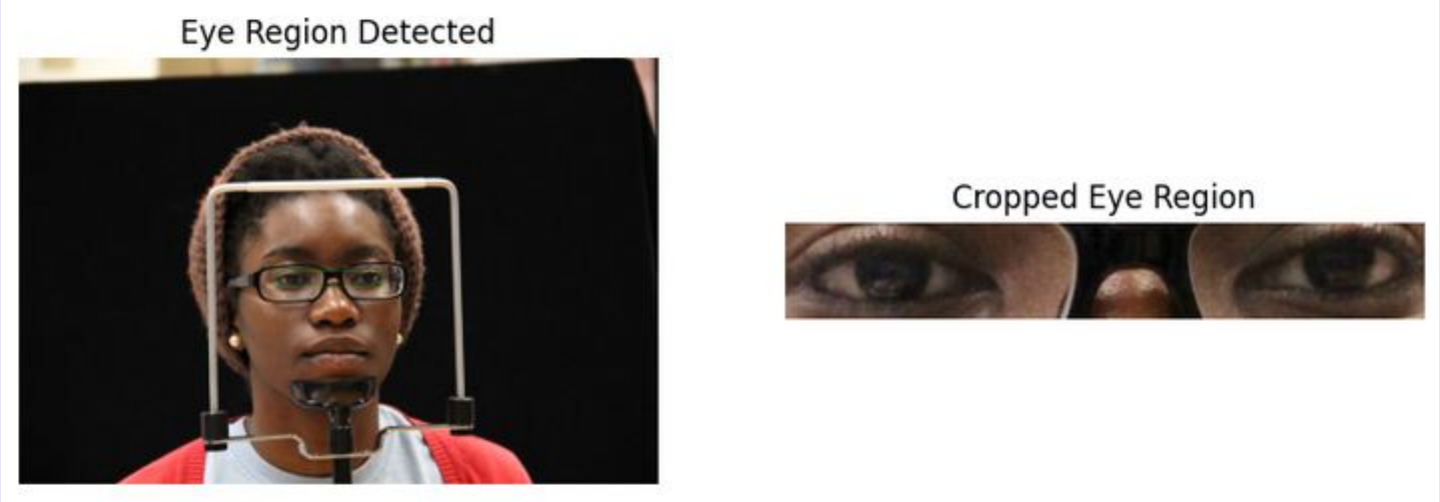
While our model was learning on the training set, it was invariably predicting the null class on any validation/test set.



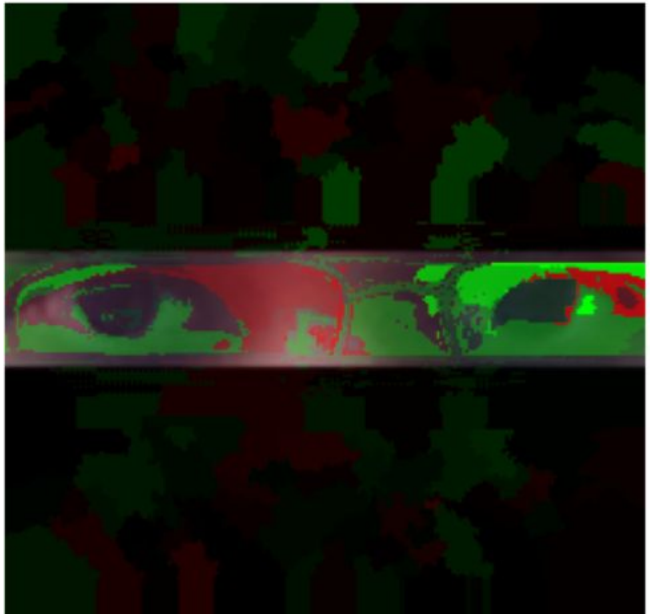


- 1 of 728 positive class photos
- Part of the trouble is the extra details in the photo.

To reduce the noise any one photo introduces to our model, we implemented Mediapipe to crop the photos before prediction:



To improve performance, we added MobileNet on top of our base architecture.



Overfitting!

After testing 8 different adjustments individually to see how the model addressed **overfitting** , we found that:

Improving Data

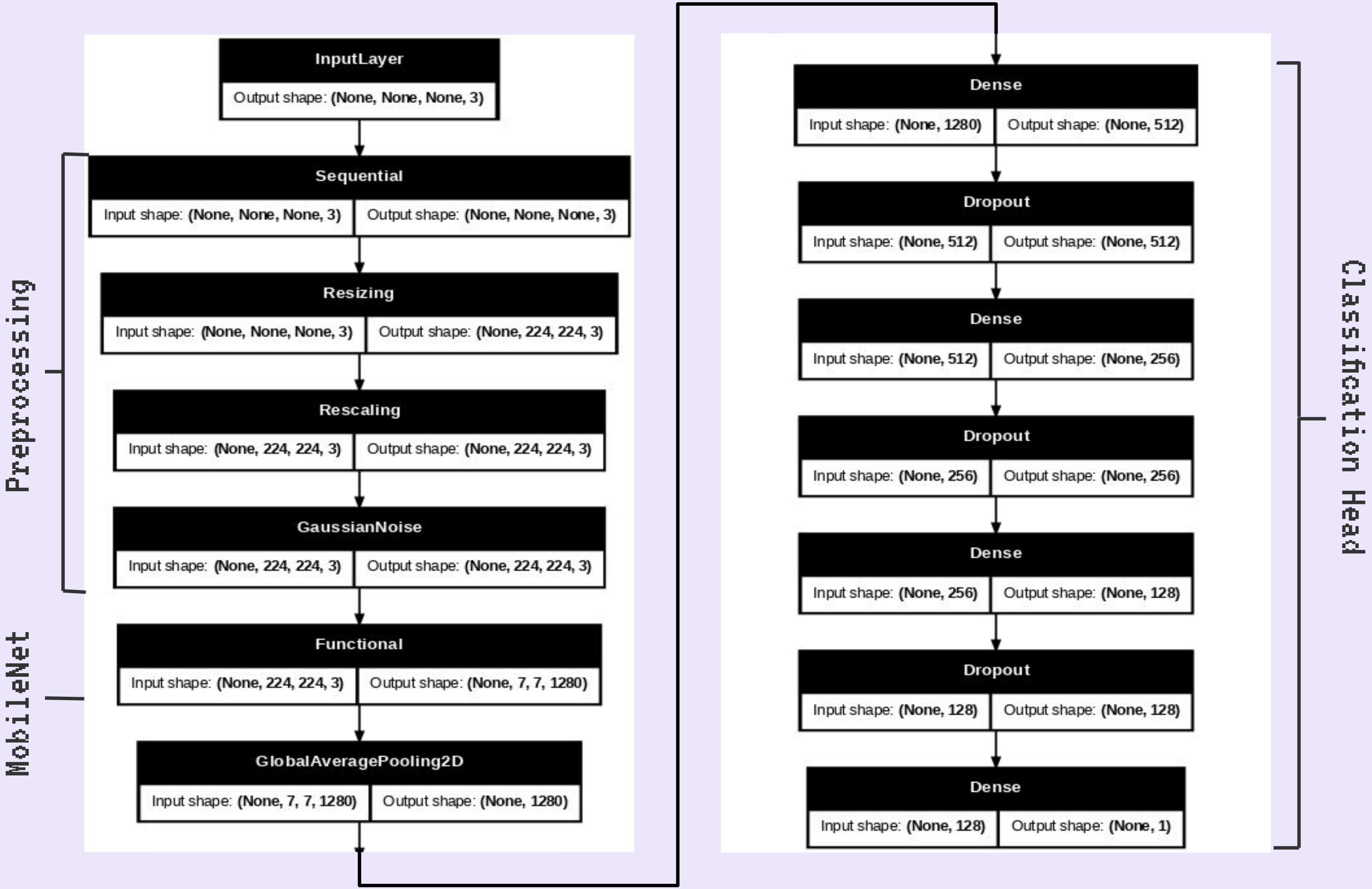
- Data Augmentation
- Gaussian Noise
- Class Weights

Improving Model

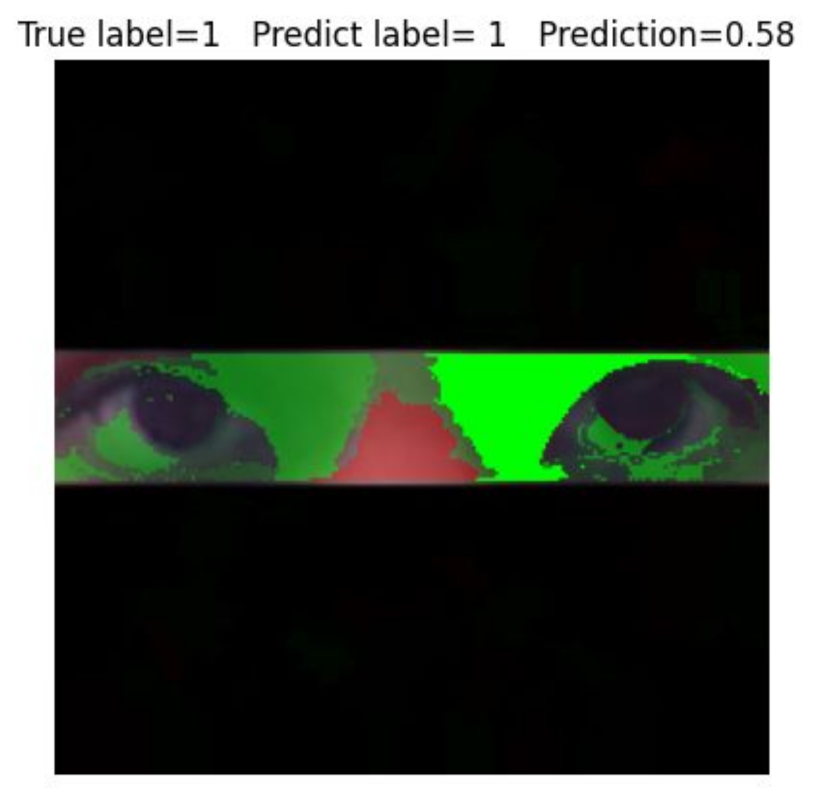
- Drop-out Layers
- L2 Regularization
- Threshold Adjustment

All resulted in marginal improvements in the model's precision. We were most interested in the model's ability to predict the positive class.

FINAL MODEL ARCHITECTURE



The highlighted area includes both eyes and the nose bridge, suggesting the model may have learned that eye alignment and facial symmetry are important cues for gaze detection.



Progression Made

Base	Predicted -	Predicted +
Actual -	773	00
Actual +	110	00

- Mediapipe
- MobileNet

Intermediate	Predicted -	Predicted +
Actual -	430	343
Actual +	36	74

- Deeper Model
- Fine Tuning

Final	Predicted -	Predicted +
Actual -	268	496
Actual +	28	152

Failures & Learnings

Face/ Eye detection Models Tested

- Haar Cascades
- Mediapipe (Iris Detection)
- Face Alignment
- Dlib
- Insightface

Pre-Trained Models Tested

- DenseNet
- EfficientNet
- ResNet

Additional Techniques Tested

- Random blur
- Random blackout
- Label smoothening
- Other loss functions
- Oversampling Minority Class

Say Cheese!

Results

Compared several cropping methods, backbone architectures and regularization strategies to arrive at a model balancing speed and accuracy

Next steps

Real-time face detection works reliably but gaze prediction lags in the live demo. By allocating more computational power and increased development time, we can fine tune the model to perform even better on webcam quality data.

Thank You!

Questions?