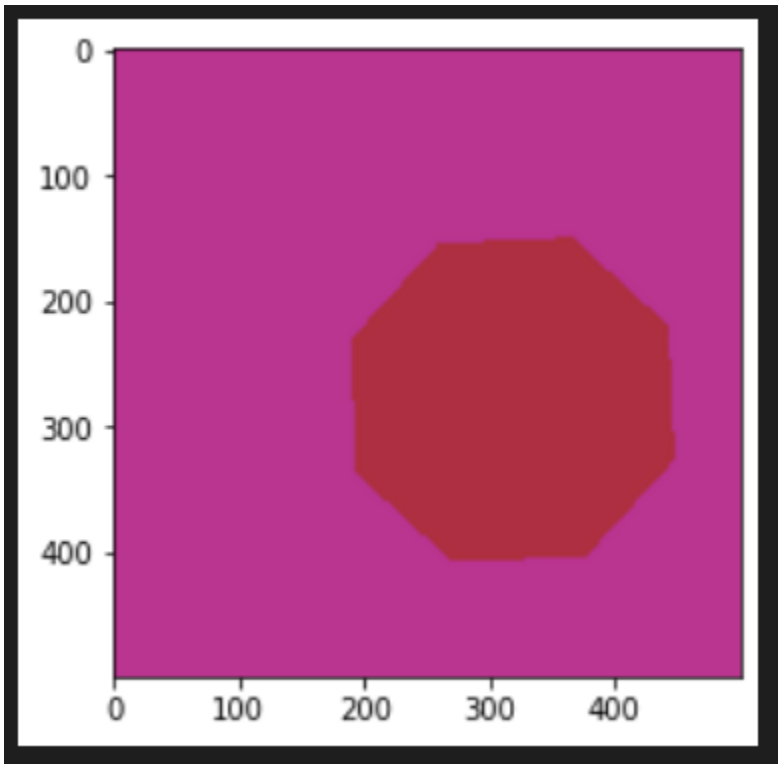
CODE REPORT
GROUP NUMBER 7

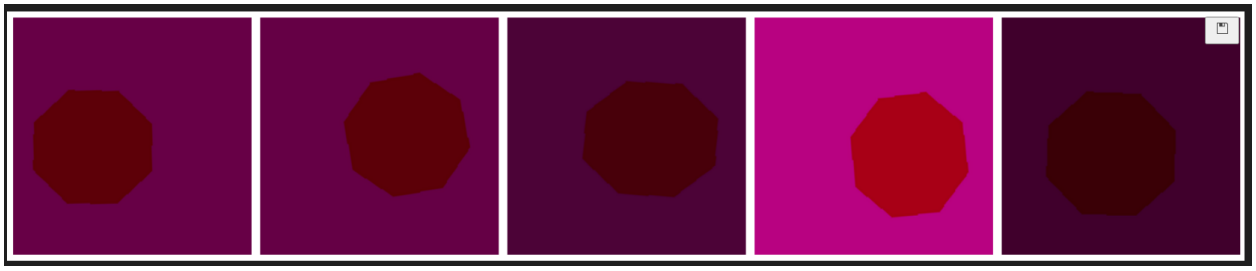
Harsh Gupta = 101916125 [LEADER]
Shubham Tiwari = 101916126
Harman Jeet Singh = 101917027

We will be using tensorflow keras CNN for this application.

First we used **ImageDataGenerator** of tensorflow to generate augmented images like [code in augumentor.ipynb]



To This :



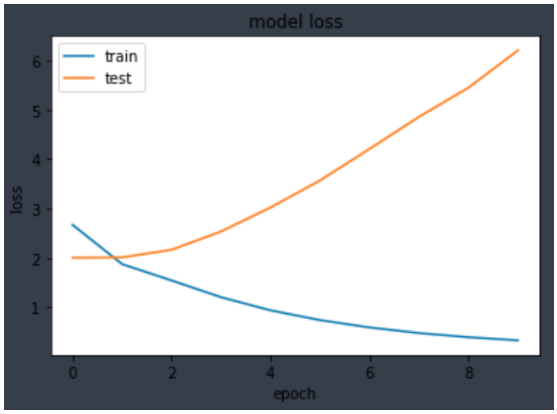
We then split the dataset into respective categories and put them into train , valid and test folders according to the 60 20 20 rule.

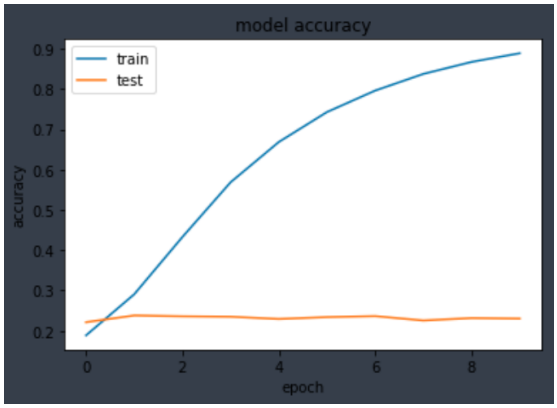
Then We uploaded This dataset into google colab and trained our model there.

Model Details [Model trained on google cloud code in train_model.ipynb]:

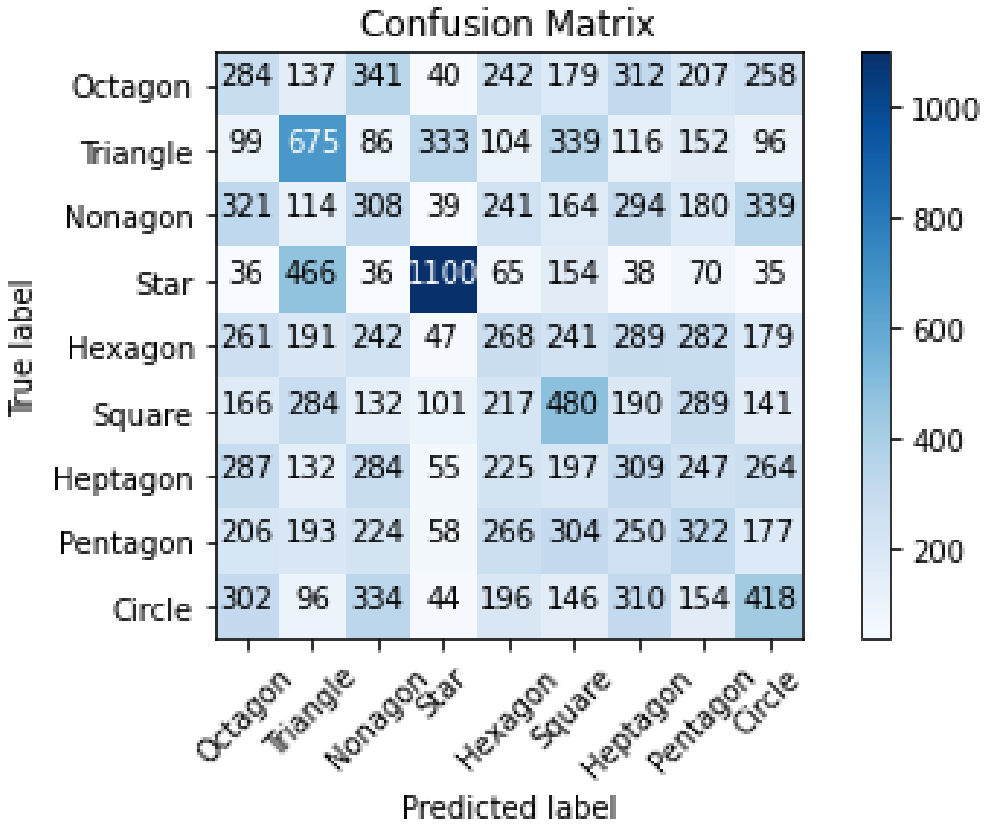
1. Our model had 5 hidden layers 2 of them were convolutional layers.
2. First hidden layer was a convolutional layer having 32 filter size and (3,3) kernel size with relu as activation function.
3. Then We had a Max pooling layer having pool size of 2,2 with strides = 2.
4. Then we again had a convolutional layer having 64 filter size and (3,3) kernel size with relu as activation function.
5. Then We had a Max pooling layer having pool size of 2,2 with strides = 2
6. Followed by a flatten layer.
7. Our Output layer was Dense type with softmax as activation function.

Loss and Accuracy graphs of our model history are shown below [code in model.ipynb]





We then run our model to predict results on our test data as this was the confusion matrix at the end



We also calculated our models accuracy using model.evaluate which comes out to be close to 0.3 which is very less.

We figured our model was overfitting due to Extra Large Sample Size and Not Tuning the model properly and using poor Edge detectors and not enough layers.

But we could not re-run the model as it took a large amount of time to run in the first place.

Table of the parameters in our own model -

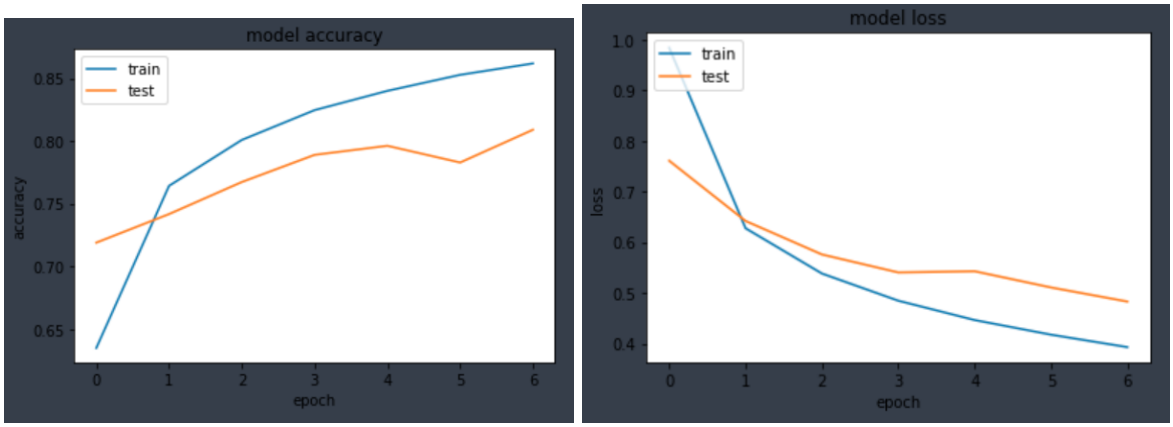
train valid test ratio	60 20 20
Learning rate in optimization algorithms	0.0001
Choice of optimization algorithm	Adam optimizer
Choice of activation function	relu for convolution layer and softmax for Dense layer
The choice of cost or loss function	categorical_crossentropy
Number of hidden layers in ANN	5
Number of iterations (epochs) in training ANN	10
Kernel size in convolutional layers	3,3
filter size in convolutional layers	32 in one and 64 in other

So we decided to use a pre-built model like **vgg16_model from tensorflow** and use a transfer learning technique. Here is the final model summary after changing the last layer [code in premodel.ipynb]

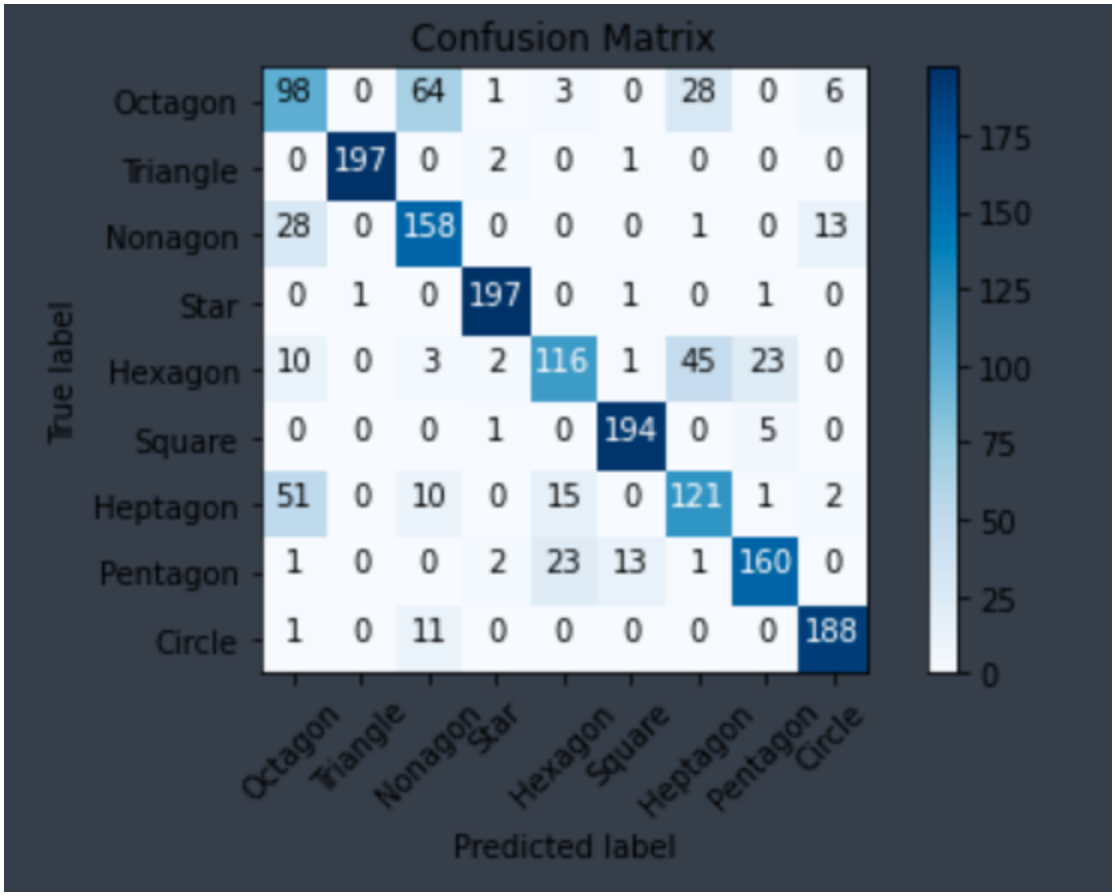
```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                 Output Shape              Param #
=================================================================
 block1_conv1 (Conv2D)        (None, 224, 224, 64)      1792

 block1_conv2 (Conv2D)        (None, 224, 224, 64)      36928

 block1_pool (MaxPooling2D)   (None, 112, 112, 64)      0

 block2_conv1 (Conv2D)        (None, 112, 112, 128)     73856

 block2_conv2 (Conv2D)        (None, 112, 112, 128)     147584

 block2_pool (MaxPooling2D)   (None, 56, 56, 128)       0

 block3_conv1 (Conv2D)        (None, 56, 56, 256)       295168

 block3_conv2 (Conv2D)        (None, 56, 56, 256)       590080
```

```
block3_conv3 (Conv2D)       (None, 56, 56, 256)       590080

block3_pool (MaxPooling2D)  (None, 28, 28, 256)       0

block4_conv1 (Conv2D)       (None, 28, 28, 512)       1180160

block4_conv2 (Conv2D)       (None, 28, 28, 512)       2359808

block4_conv3 (Conv2D)       (None, 28, 28, 512)       2359808

block4_pool (MaxPooling2D)  (None, 14, 14, 512)       0

block5_conv1 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv2 (Conv2D)       (None, 14, 14, 512)       2359808

block5_conv3 (Conv2D)       (None, 14, 14, 512)       2359808

block5_pool (MaxPooling2D)  (None, 7, 7, 512)         0

flatten (Flatten)           (None, 25088)             0

fc1 (Dense)                 (None, 4096)              102764544

fc2 (Dense)                 (None, 4096)              16781312

dense_2 (Dense)             (None, 9)                 36873

=================================================================
Total params: 134,297,417
Trainable params: 36,873
Non-trainable params: 134,260,544
_________________________________________________________________
```

Now we trained and evaluated the performance over epochs. Here are the graphs



We then run our model to predict results on our test data as this was the confusion matrix at the end

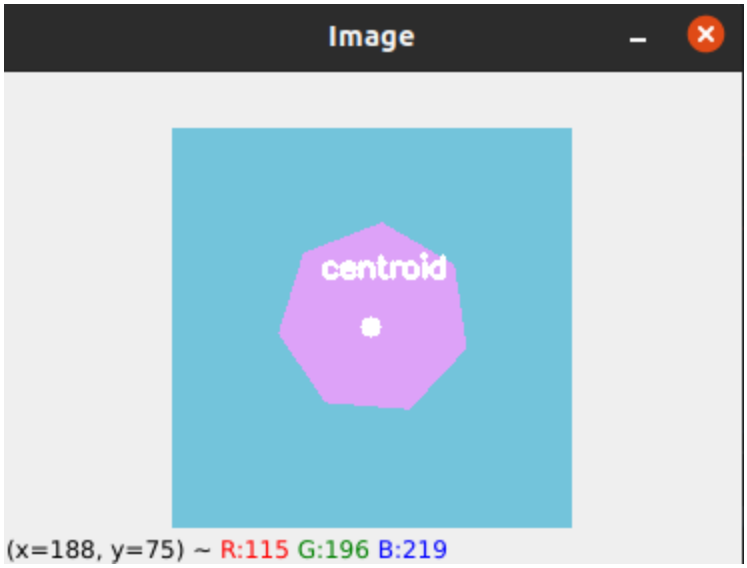


Also our model loss was `0.48529407382011414` **`on test data and accuracy was`** `0.7938888669013977.`

Which is not bad considering the low number of epochs we were able to run and fit our model with.

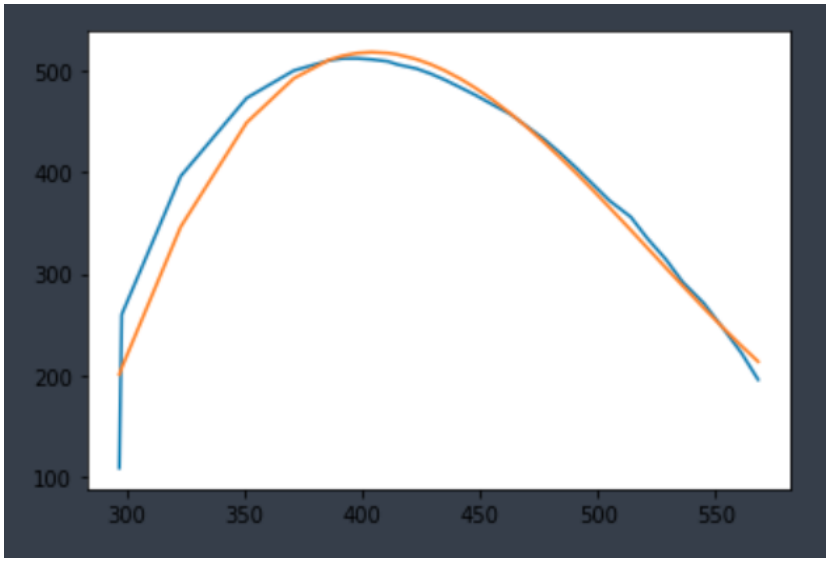
Object Tracking using open CV -

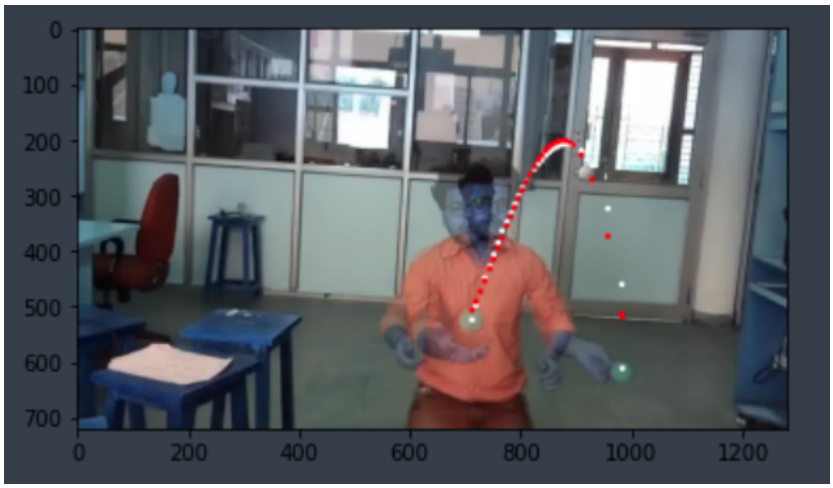
We also tried object tracking on the dataset 5 using open cv [code in object_tracking2.ipynb]



Also we tried to track object paths in Dataset 1,2,3,4 using open cv [code in objecttracking1.ipynb] but was only able to successfully do so in dataset 1 because ball was of distinct color from rest on the background and so easier to detect but could not do it for others because -
Dataset 3 as blue object and person shirt is also blue so we could not mask out the cube.
Similarly for 4 , black cube could not be masked out due to many black objects in the background.

Output for dataset 1 -





Output for rest [code in objecttracking2.ipynb]-