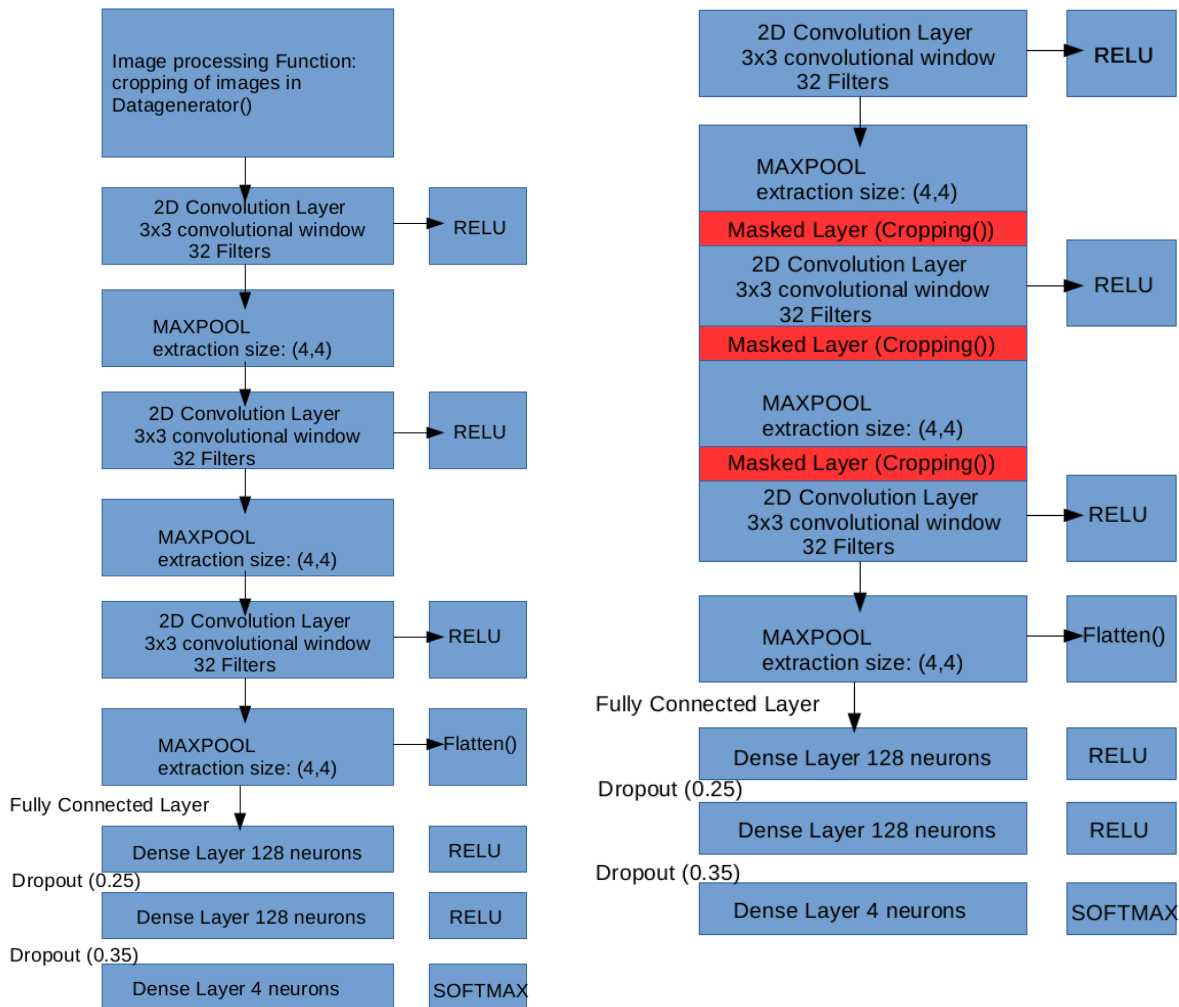


Since the neural network had a validation loss of 6.4028 with a validation accuracy of 0.58769, I changed the optimizer from rms_prop back to adam and reduced the 2D Maxpool size from (8,8) to (4,4). There was no problem loading the dataset of 1,000 images per each shape, with a total of 4,000 images. During the loading of the second half of the dataset, a callback function in keras called early stopping will stop the learning of the network to prevent overfitting. The convolutional neural network stopped at the 8/20 epoch in the loading of the second half of the dataset, which I think is too early for the neural network to stop. It might have stopped because the dataset is insufficient, I will increase the dataset by twice as much as the previous dataset after incorporating the cropping function in Keras. The validation loss is at 4.8 percent with a validation accuracy of 67.5%.

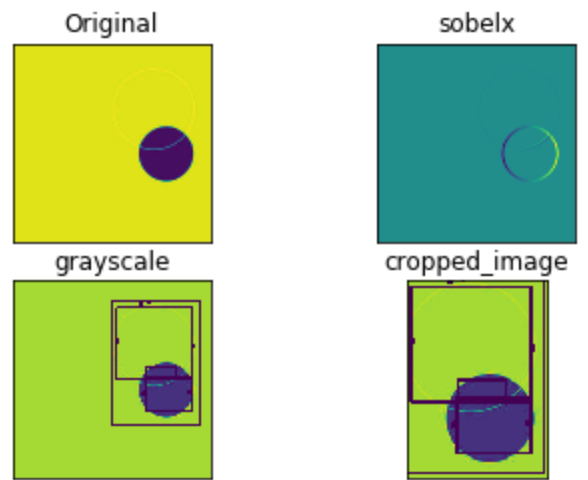
Cloud ML Job, cncnopy_train_20171110_14... All logs Any log level Jump to date

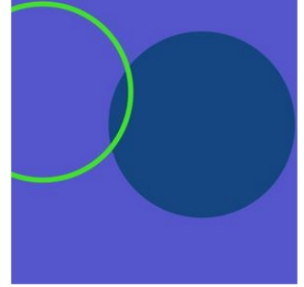
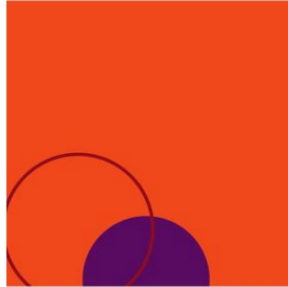
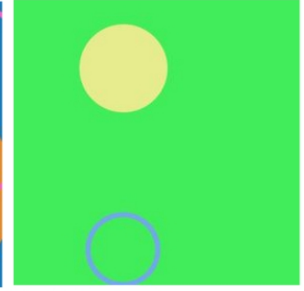
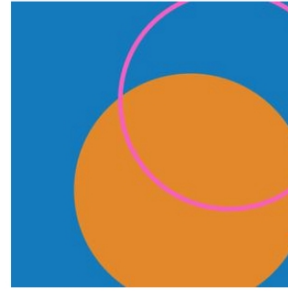
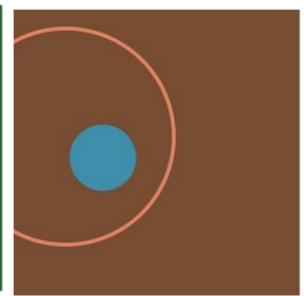
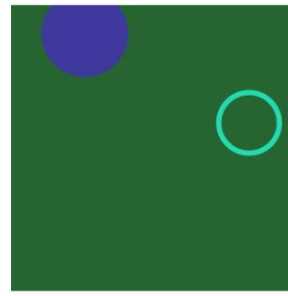
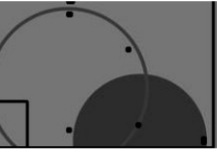
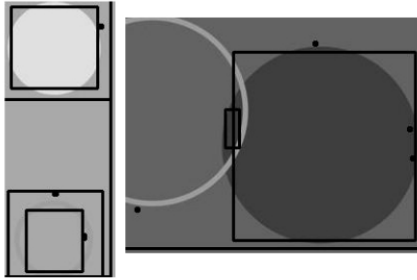
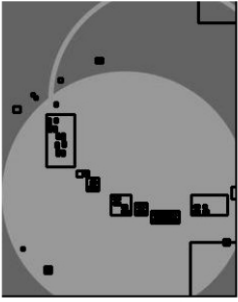
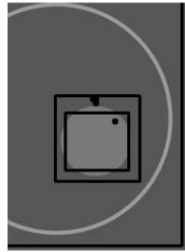
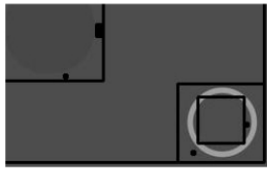
2017-11-10 EST		View Options
15:50:59.909	master-replica-0 63/62 [=====] - 169s 3s/step - loss: 0.5606 - acc: 0.6870 - val_loss: 0.6268 - val_acc: 0.6700	
15:51:00.095	master-replica-0 Epoch 8/20	
15:51:00.266	master-replica-0 1/62 [.....] - ETA: 11s - loss: 0.5262 - acc: 0.6875	
15:51:00.425	master-replica-0 2/62 [.....] - ETA: 10s - loss: 0.5432 - acc: 0.6719	
15:51:00.576	master-replica-0 3/62 [>.....] - ETA: 10s - loss: 0.5673 - acc: 0.6979	
15:51:00.717	master-replica-0 4/62 [>.....] - ETA: 9s - loss: 0.5347 - acc: 0.7031	
15:51:00.861	master-replica-0 5/62 [=>.....] - ETA: 9s - loss: 0.5171 - acc: 0.7000	
15:51:01.003	master-replica-0 6/62 [=>.....] - ETA: 8s - loss: 0.5258 - acc: 0.7135	
15:51:01.145	master-replica-0 7/62 [==>.....] - ETA: 8s - loss: 0.5091 - acc: 0.7232	
15:51:01.286	master-replica-0 8/62 [==>.....] - ETA: 8s - loss: 0.5042 - acc: 0.7305	
15:51:01.429	master-replica-0 9/62 [===>.....] - ETA: 8s - loss: 0.5036 - acc: 0.7326	
15:51:01.570	master-replica-0 10/62 [===>.....] - ETA: 7s - loss: 0.5091 - acc: 0.7219	
15:51:01.711	master-replica-0 11/62 [====>.....] - ETA: 7s - loss: 0.5057 - acc: 0.7188	
15:51:01.854	master-replica-0 12/62 [====>.....] - ETA: 7s - loss: 0.4916 - acc: 0.7266	
15:51:01.996	master-replica-0 13/62 [====>.....] - ETA: 7s - loss: 0.4943 - acc: 0.7236	
15:51:02.137	master-replica-0 14/62 [====>.....] - ETA: 7s - loss: 0.4931 - acc: 0.7277	
15:51:02.480	master-replica-0 15/62 [====>.....] - ETA: 7s - loss: 0.4980 - acc: 0.7229	
15:51:02.934	master-replica-0 16/62 [====>.....] - ETA: 7s - loss: 0.4953 - acc: 0.7266	
15:51:03.386	master-replica-0 17/62 [====>.....] - ETA: 8s - loss: 0.5006 - acc: 0.7261	
15:51:03.831	master-replica-0 18/62 [====>.....] - ETA: 8s - loss: 0.4974 - acc: 0.7257	
15:51:04.282	master-replica-0 19/62 [====>.....] - ETA: 8s - loss: 0.5003 - acc: 0.7204	
2017-11-10 EST		View Options
15:51:19.143	master-replica-0 52/62 [=====>.....] - ETA: 3s - loss: 0.4991 - acc: 0.7188	
15:51:19.600	master-replica-0 53/62 [=====>.....] - ETA: 3s - loss: 0.4992 - acc: 0.7193	
15:51:20.049	master-replica-0 54/62 [=====>.....] - ETA: 3s - loss: 0.4977 - acc: 0.7211	
15:51:20.503	master-replica-0 55/62 [=====>.....] - ETA: 2s - loss: 0.4997 - acc: 0.7193	
15:51:20.958	master-replica-0 56/62 [=====>.....] - ETA: 2s - loss: 0.5013 - acc: 0.7193	
15:51:21.423	master-replica-0 57/62 [=====>.....] - ETA: 2s - loss: 0.5000 - acc: 0.7198	
15:51:21.884	master-replica-0 58/62 [=====>.....] - ETA: 1s - loss: 0.4990 - acc: 0.7220	
15:51:22.340	master-replica-0 59/62 [=====>.....] - ETA: 1s - loss: 0.5015 - acc: 0.7203	
15:51:22.829	master-replica-0 60/62 [=====>.....] - ETA: 0s - loss: 0.5001 - acc: 0.7214	
15:51:23.287	master-replica-0 61/62 [=====>.....] - ETA: 0s - loss: 0.5001 - acc: 0.7203	
15:53:48.819	master-replica-0 62/62 [=====>.....] - ETA: 0s - loss: 0.5017 - acc: 0.7203	
15:53:48.820	master-replica-0 63/62 [=====>.....] - 169s 3s/step - loss: 0.5016 - acc: 0.7208 - val_loss: 0.5592 - val_acc: 0.6801	
15:53:52.904	master-replica-0 Test loss: 4.8990672636	
15:53:52.904	master-replica-0 Test accuracy 0.675	
15:53:53.001	master-replica-0 Module completed; cleaning up.	
15:53:53.001	master-replica-0 Clean up finished.	
15:53:53.002	master-replica-0 Task completed successfully.	
15:54:05.236	Tearing down TensorFlow.	
15:55:23.513	Finished tearing down TensorFlow.	
15:56:07.727	Job completed successfully.	

I made changes to the architecture of the network where I won't incorporate dropout during the beginning of the convolutional extractions. This is because it is important to retain as much important information about the shapes since I will be incorporating a cropping function either as a unified data generator as a callback parameter or ask a layer mask between the convolution layer. Dropout will occur after the convolutional layers are flattened and in between the 2-layer network. The network will still use stochastic learning since it's convenient and generates good learning rates. Below are the diagrams that explains the two modified designs of the neural network:



In order to make the convolutional neural network learn better, I created a cropping function that finds the boundaries of the shapes and crops accordingly. This cropping function will be applied as a parameter in the data_generator in the convolutional neural network loader file. This function is needed because the unnecessary background data that does not contain the pixels of the shapes will create more noise in the neural network during the convolutional 2D process of image extraction. The cropping function will make the neural network focus on the shapes in background instead of extracting colors of backgrounds with no important features in them besides colors. The function uses the cv2 module which is having problem with dependencies from python 2 to python 3 in the anaconda installations. The module works in python 3 but not in python 2 so adjustment to fix this issue will be needed. The procedure of the program is to use the sobel algorithm to detect the contours of the edges. The images have to be in grayscale beforehand, but the convolutional neural network extract grayscale images because the function will return the coordinates necessary for cropping. The function will create a boundary rectangle around the found contours and crop as necessary. Here are the black and white outputs of the cropped images:





```
cnncopy.py x pickletestcats.py x sobel_copy.py x draw.py x
1 """
2 Created on Sun Nov 12 18:41:03 2017
3 Program finds contours and creates a rectangular boundary around it
4 This program will be implemented in the cnn loader file
5 @author: maggie
6 """
7 import cv2
8 from matplotlib import pyplot as plt
9 import scipy.misc
10
11 """code to get boundaries of contour shapes and crops the images based on the location of the edges"""
12 def get_edges(image_path):
13     img = cv2.imread(image_path)
14     #requires the image to be in grayscale
15     grayscale = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
16
17     # Gaussian to remove noise from
18     img = cv2.GaussianBlur(grayscale,(3,3),0)
19     # Edge Detection Filter: use sobel algorithm to detect edges
20     sobelx = cv2.Sobel(img,cv2.CV_64F,1,0,ksize=5)
21     sobely = cv2.Sobel(img,cv2.CV_64F,0,1,ksize=5)
22     gradient_x = cv2.convertScaleAbs(sobelx)
23     gradient_y = cv2.convertScaleAbs(sobely)
24     #combine two sobel gradients
25     gradient_t = cv2.addWeighted(gradient_x, 0.5, gradient_y, 0.5, 0)
26
27     #Threshold filter
28     #create threshold to separate edges from background
29     #retval, thresh_gray = cv2.threshold(grayscale, thresh=127, maxval=255, type=cv2.THRESH_BINARY)
30     im2, contours, hierarchy = cv2.findContours(gradient_t, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
31     cv2.drawContours(gradient_t, contours, 0, (0,0,255), 2)
32
33     #with each contour, find 2 sets of coordinates to draw bounding rectangle
34     edge_list_x = []
35     edge_list_y = []
36     for c in contours:
37         x, y, width, height = cv2.boundingRect(c)
38         cv2.rectangle(grayscale, (x,y), (x+width, y+height), (0,255,0), 2) #draw green rectangle
39         edge_list_x.append(x)
40         edge_list_y.append(y)
41         edge_list_x.append(x+width)
42         edge_list_y.append(y+height)
```

```

43
44 smallest_x = min(edge_list_x)
45 smallest_y = min (edge_list_y)
46 largest_x = max(edge_list_x)
47 largest_y = max(edge_list_y)
48
49 offset = 5
50 #cv2.rectangle(grayscale, (smallest_x, smallest_y), (largest_x, largest_y), (0,255,0), 2)
51
52 # use this function to crop in keras 2Dcropping (extra layer between CNN layers)
53 crop_image = grayscale [smallest_y + offset:largest_y + offset,
54                         smallest_x + offset:largest_x + offset]
55
56 plt.subplot(2,2,1),plt.imshow(img)
57 plt.title('Original'), plt.xticks([], plt.yticks([]))
58 plt.subplot(2,2,2),plt.imshow(sobelx)
59 plt.title('sobelx'), plt.xticks([], plt.yticks([]))
60 plt.subplot(2,2,3),plt.imshow(grayscale)
61 plt.title('grayscale'), plt.xticks([], plt.yticks([]))
62 plt.subplot(2,2,4),plt.imshow(crop_image)
63 plt.title('cropped_image'), plt.xticks([], plt.yticks([]))
64
65 return crop_image
66
67 if __name__ == '__main__':
68     crop = get_edges("test_images/circle101.jpg")
69     scipy.misc.imsave("circle7.jpg", crop)

```

```

cnn_copy_sobel.py X sobel_copy.py X
55 # a decrease in accuracy @ adeshpande3.github.io''
56 classifier.add(Conv2D(32, (3, 3), input_shape = (300, 300, 3), activation = 'relu'))
57
58 # Step 2:
59 # Max Pooling downsamples the number pixels per neuron and create a max
60 # number that describes those features in a pool_size of 2 by 2
61 # change pool size from (2,2) to (8,8) to (4,4)
62 classifier.add(MaxPooling2D(pool_size = (4, 4)))
63
64 # Adding a second convolutional layer, which is the same as the first one
65 classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
66 # change pool size from (2,2) to (8,8)
67 classifier.add(MaxPooling2D(pool_size = (4, 4)))
68 # Dropout layers at the second convolutional layer before flattening
69 #classifier.add(Dropout(0.25)) #don't add dropout here to fix it? 11/11 Saturday
70
71 # Step 3: Flattening the convolutional layers for input into a fully
72 # connected layer
73 classifier.add(Flatten())
74
75 # Step 4:
76 # Fully connected: Dense function is used to add a fully connected
77 # 3 layer perceptron at the end
78 classifier.add(Dense(units = 128, activation = 'relu'))
79 # dropout at the first layer perceptron
80 classifier.add(Dropout(0.25))
81 # adding second hidden layer - remove if accuracy decreases or loss increases
82 classifier.add(Dense(units = 128, activation = 'relu'))
83 classifier.add(Dropout(0.35))
84 # softmax classifier as an activation from the last layer perceptron
85 # units represent number of output classes
86 # the output classes are triangle, rectangle, square, circle
87 classifier.add(Dense(units = 4, activation = 'softmax'))
88
89 # Compiling the CNN:
90 # check if optimizer adam is good, categorical_crossentropy is for
91 # multi-class network, multilabel with intersection needs binary_crossentropy
92 # and sigmoid activations
93 # change from adam to rmsprop
94 classifier.compile(optimizer = 'adam',
95                  loss = 'categorical_crossentropy',
96                  metrics = ['accuracy'])

```

```

98 # Part 2:
99 # Feeding CNN the input images and fitting the CNN
100 # CNN uses data augmentation configuration to prevent overfitting
101 # datagen augmentation is for training data input
102 datagen = ImageDataGenerator(featurewise_center = True,
103                             featurewise_std_normalization = True,
104                             rescale = 1./255,
105                             shear_range = 0.2,
106                             zoom_range = 0.2,
107                             horizontal_flip = True)
108 datagen.preprocessing_function = get_edges(train_shape_dataset))
109
110 # augmentation configuration for rescaling images used for validation
111 validate_datagen = ImageDataGenerator(rescale = 1./255)
112
113 # the test set data augmentation only rescales the images
114 # is this enough to test the network correctly? if you want a more manual
115 # representation of fitting the input data use for loop
116 validate_datagen.fit(validate_shape_dataset)
117 validate_generator = datagen.flow(validate_shape_dataset,
118                                 validate_y_dataset,
119                                 batch_size = 32)
120
121 # the code below fits the training data that is loaded by pickle file
122 # to prevent memory error, 1/2 of the number of data inputs are feed first
123 # an epoch define the input being run once from
124 # the architecture of the cnn is:
125 # 2DConv -> ReLU -> MaxPool -> 2DConv -> ReLU -> MaxPool -> Flatten() ->
126 # Fully connected 2-layer neural network
127 # 128 neurons for the first layer -> ReLU -> 128 for hidden layer -> ReLU
128 # -> 3 neurons for output layer -> softmax
129
130 # compute quantities required for featurewise normalization
131 datagen.fit(train_shape_dataset)
132 #early_stopping = EarlyStopping(monitor = 'val_loss', patience = 2)
133 # fits the model on batches with real-time data augmentation
134 train_generator = datagen.flow(train_shape_dataset,
135                               train_y_dataset,
136                               batch_size = 32)
137 classifier.fit_generator(train_generator, #train generator
138                        steps_per_epoch = len(train_shape_dataset) / 32,
139                        epochs = 20,
140                        validation_data = validate_generator,

```

The ImageDataGenerator will be normalized via featurewise_center and featurewise_std_normalization in order to make the convolutional neural network gather features that have equal significance. Sometimes the shapes blend with the background environment, so normalizing the data will make the unfilled and filled shapes the same. In this code, the cropping function of data shapes will be incorporated as a function in data generator of keras. This means that the images will be cropped before being fed into the network. To maintain the original images of the network, I can use the cropping as a layer mask in between the 2D convolution architecture by using Kera's 2D cropping:

```

classifier.add(Conv2D(32, (3, 3), input_shape = (300, 300, 3), activation = 'relu'))
classifier.add(MaxPooling2D(pool_size = (4, 4)))
# Adding a second convolutional layer, which is the same as the first one
classifier.add(Conv2D(32, (3, 3), activation = 'relu'))
# change pool size from (2,2) to (8,8)
classifier.add(MaxPooling2D(pool_size = (4, 4)))
# Dropout layers at the second convolutional layer before flattening
keras.layers.Cropping2D(cropping=get_edges(train_shape_dataset), data_format=None)
# Step 3: Flattening the convolutional layers for input into a fully
# connected layer
classifier.add(Flatten())

```