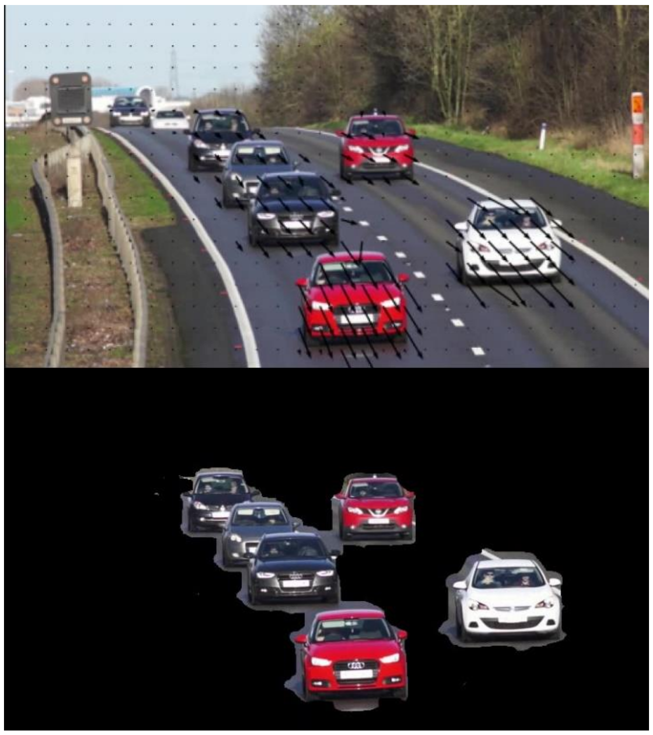
Dani Lerner
ENPM673

Project 4

Part 1: Optical Flow

For Project 4 Part 1, I used Optical Flow techniques to derive a velocity field of cars driving on a highway. Since we could use built-in functions, the process was relatively straightforward. I first converted the first frame to grayscale and saved it to a variable called previous frame. I further used this first image to create a mesh-grid with a resolution of 25 pixels both in the x and y. Then for each remaining frame, I converted the image to grayscale then immediately calculated the optical flow with the cv2.calcOpticalFlowFarneback function. This function required both the current frame and the previous frame as well as several tuning parameters.

Then, I created a matplotlib figure, docked the image onto it and further overlayed a plt.quiver vector field using the mesh-grid I had created with the first image and the results of the optical flow calculation. With the vector field image complete, I just needed to create and apply an image mask of the cars. Since the cars are moving while the surroundings are stagnant, all I needed to do was to find which pixels in the system yielded large optical flow/movement and threshold the results. To perform this process, I first used the cv2.cartToPolar function to convert the optical flow results from cartesian to polar coordinates. Then I normalized the magnitudes, converted my angles to degrees, and finally thresholded the magnitudes with a very small value. This gave me a binary image with just the pixels that displayed the moving cars. Lastly, I applied an erosion to reduce noise and remove some of the pixels surrounding the cars. Using a bitwise-and, I proceeded to mask the mask with the original frame and got the results for each case as shown below.



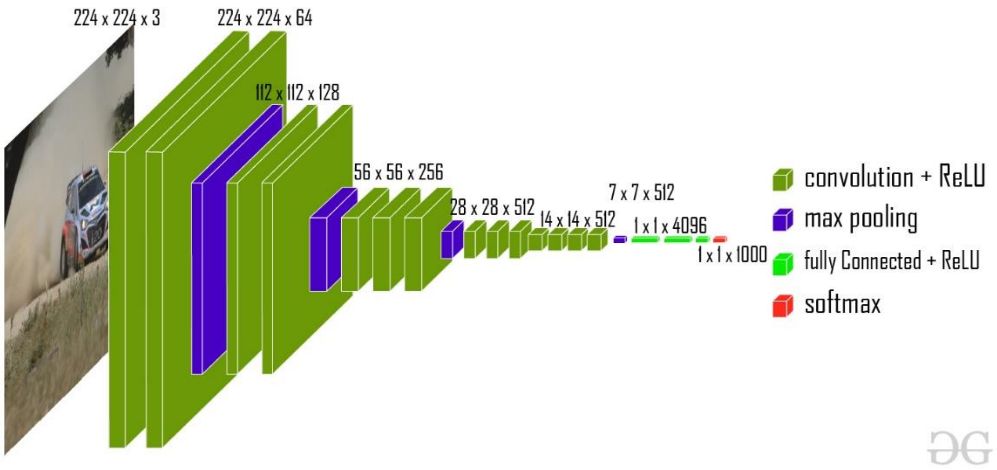
Part 2: Fish Classification

For this section, I implemented a self-made VGG-16 convolution neural network architecture using Keras and Tensorflow to train and execute fish type multi-class image classification. While this project seems daunting, the process was relatively straightforward with the high-level Keras library. At first, I tried training the model on my PC but the code only worked when I mounted the calculations on my CPU. When using my 4 GB RAM GPU, I received OOM (Out of Memory) errors and other errors that I believe were related to memory after some research. This does not make much sense since I tried working with a batch size of one but still had the same results. When I ran identical code using the CPU, however, it worked perfectly but each epoch took 30 minutes. Consequently, I gave up working on my GPU and started using Google Collab. Here, I ran the code with a TPU and High-RAM usage and got results within several minutes per epoch. It turns out, however, that the TPU runs far slower than the GPU on Google Collab for training. After some research, it turns out that TPU's are optimized for specific types of calculations. To use a TPU correctly, there is some Tensorflow setup I would have needed to implement to optimize the speed.

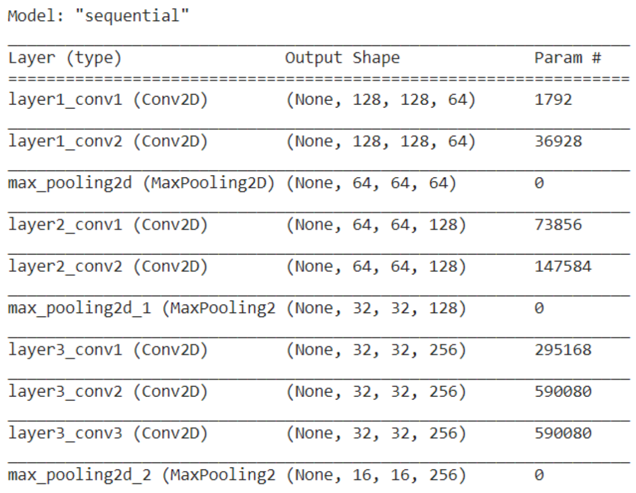
Training

For my architecture, I built a VGGNet-16 CNN architecture with the preset filter counts. Before applying the model, I had to read in all the images and their corresponding labels. Then I ran an sklearn.train_test_split on the images to separate the train data from the test data. A parameter of this function is the random_state variable which ensures that the train and testing data will be the same each time. Therefore, there will not be any randomness and variability regarding the specific images I use to train and test. I also used a label binarizer to break up my string labels into a matrix of Booleans corresponding to whether a given fish is a given type. Lastly, I implemented the actual VGG-16 CNN.

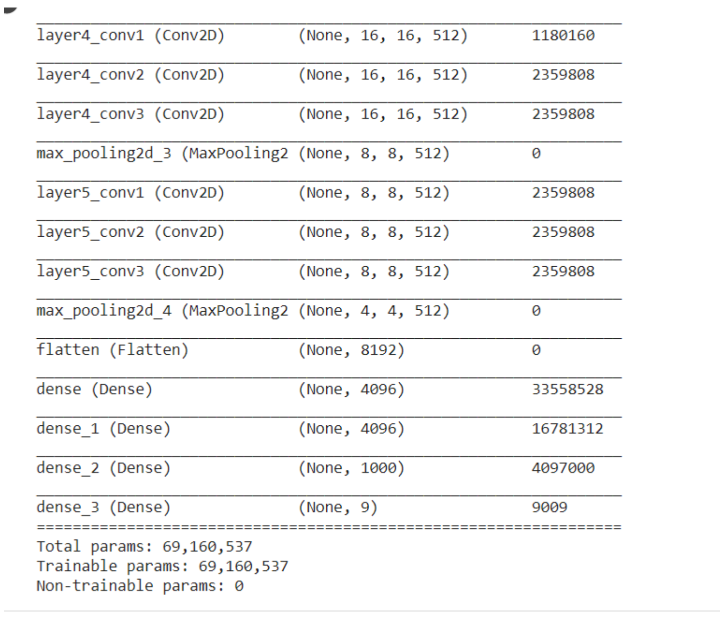
VGG-16 is a 16-layer deep neural networks with just a few types of steps. The most used step is a simple series of convolutions. Here, I convolve a designated number of 3x3 kernel filters with the image stack. This process runs filters across the data to detect features in the image. Then, I applied the ReLU (Rectified Linear Unit) activation function to the result of each convolution layer. The next most used function was the MaxPooling2D with a size and stride of 2. This function reduces the size of our images by searching through each 2x2x1 window of the image stack and only selects the largest value in the window. This results in another image with shape of "original height/2", "original width/2." The MaxPooling function does not count as a layer in the CNN. Lastly, I used the Dense layer which is the most used layer in deep neural networks. This layer is comparable to the convolution layer but has far more parameters and only works with 1x1xn data. By the time we have reached the dense layers, we have broken down areas of the image into features (through convolution). Based on the features that our convolution layers return, our dense layers map the features to the corresponding type of fish. On all but one dense layer, I used ReLU for my activation function. On the last iteration, however, I used SoftMax to train and predict the type of fish for a given image. I ran the model with multiple configurations but my most simple and control model summary is shown below:



[1]



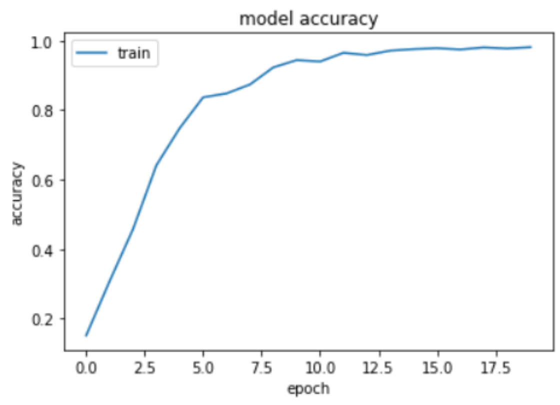
```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
```

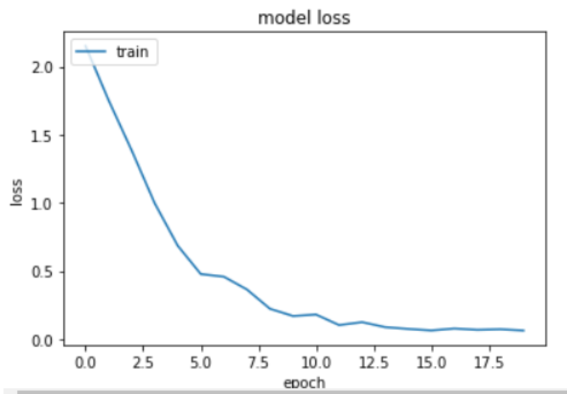


```
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              33558528
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 9009
=================================================================
Total params: 69,160,537
Trainable params: 69,160,537
Non-trainable params: 0
```

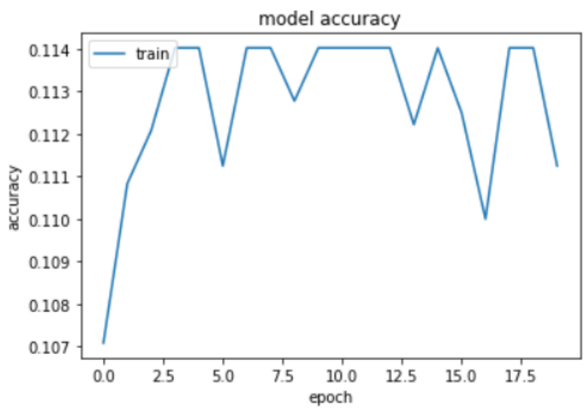
[1] https://www.geeksforgeeks.org/vgg-16-cnn-model/

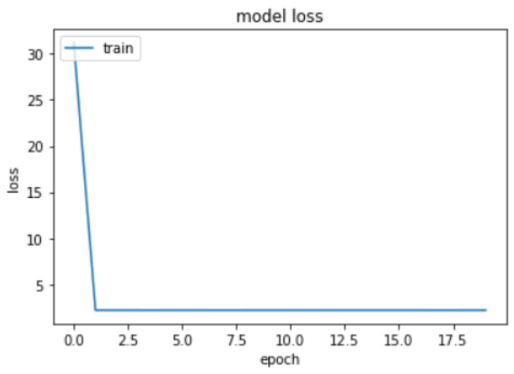
For my control run, I further used the Adam algorithm with a learning rate of 1e-3, 20 epochs, and a batch size of 128. The results were excellent, and I ended up with a training accuracy of 97.7%. I also realized that the use of 20 epochs was overkill since I had already received 97% accuracy by the 12th epoch. Consequently, I changed the number of epochs to 10 for all future iterations. The training accuracy and loss VS. epoch graphs are shown below.





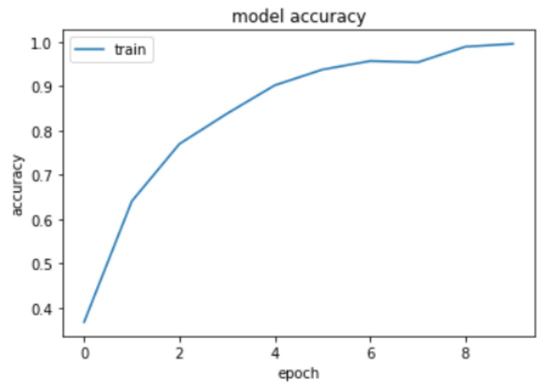
Once I had a valid model, I attempted to change up individual parameters to see how they would affect my results. For my first trial, I simply lowered batch size to 32. This yielded horrible results and my model would not converge. See below:

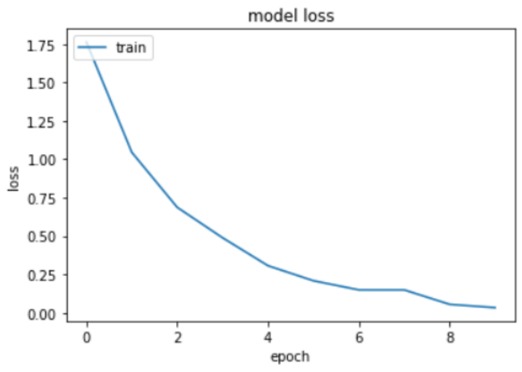




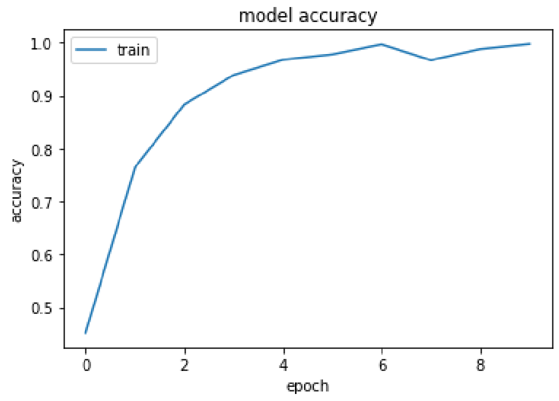
Later, I repeated this process with a smaller learning rate and the results were far better. It turned out that my learning rate had been okay but was a little too large for this application. When I changed the learning rate to 1e-5, the results were clearly better. Somehow, the training accuracy was a whopping 99.5% with a batch size of 128 and the accuracy increased per batch far faster than before. The top 2 plots shown below have a batch size of 128 (original batch size) and the lower two have a batch size of 32. Notice the sharp improvement shown by the 32 sized batch shown below.

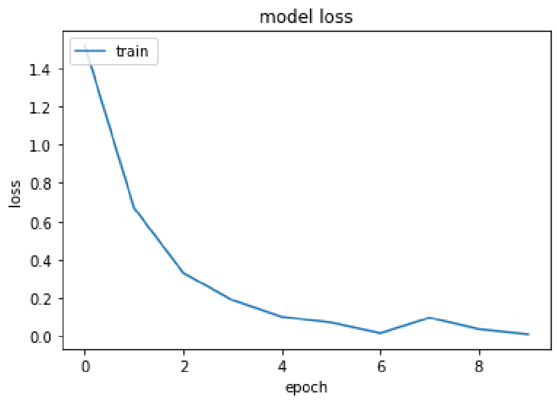
Batch size: 128





Batch size: 32





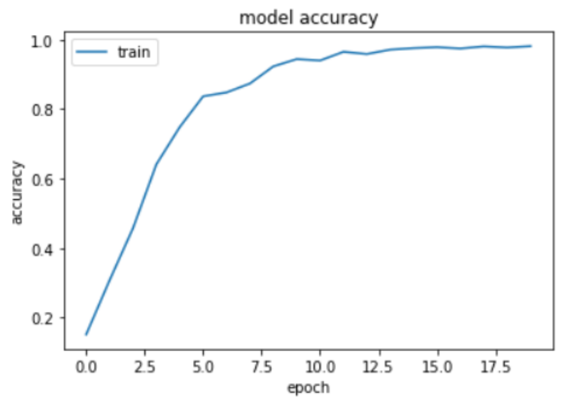
Furthermore, I expected the algorithm to train significantly slower than previously but this learning rate change from 1e-3 to 1e-5 only slowed the training down by 50s per batch or by 11%. Because of all this information, it seems that the gradient descent was missing the mark and never truly found a proper minimum with the higher learning rate. Unfortunately, I only tested the learning rate after my other iterations and changes, but the better learning rate would have certainly affected the outcomes. For the remaining changes, I altered the actual model. I changed the number of filters in each convolution and the number of parameters in the dense layers. I also added batch normalization between each convolution layer and its activation function in one iteration. Without multiple runs, it is hard to give a detailed analysis regarding every metric and its effect. However, I will display each iteration and its corresponding train/test data output. Furthermore, all model summaries can be found in the Model Summaries section.

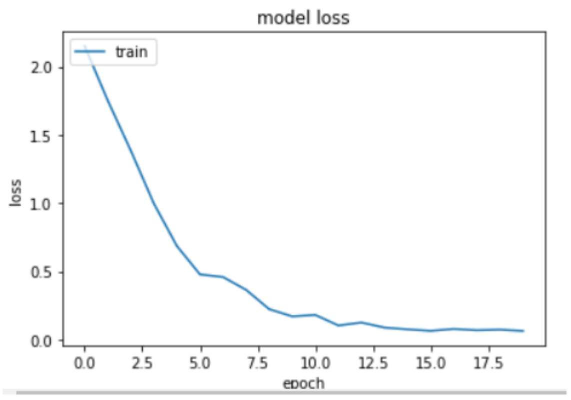
Case 1:

- Model summary shown above.
- Learning rate: 1e-3
- Batch size: 128
- Epochs: 20
- Num Params: 69.2e6

Results:

- ~400 seconds per batch
- 98.7% train accuracy
- 95.2% test accuracy
- .0726 train loss
- 0.1502 test loss



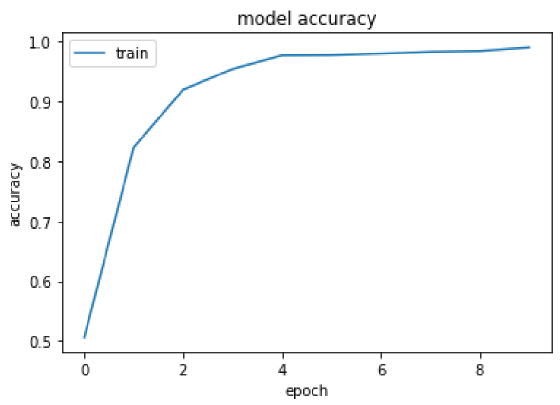


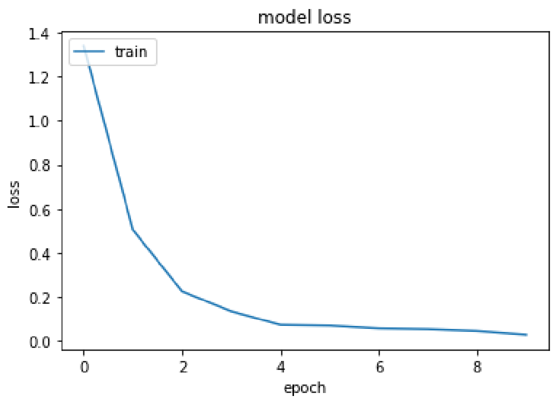
Case 2:

- Added batch normalization after every convolution layer.
- Learning rate: 1e-5
- Batch size: 32
- Epochs: 10
- Num Params: 31.7e6 including 500 non-trainable params

Results:

- ~23 seconds per batch *TRAINED ON THE GPU.* This cannot be compared with the others.
- 99.8% train accuracy
- 97.94% test accuracy
- .0087 train loss
- 0.2023 test loss





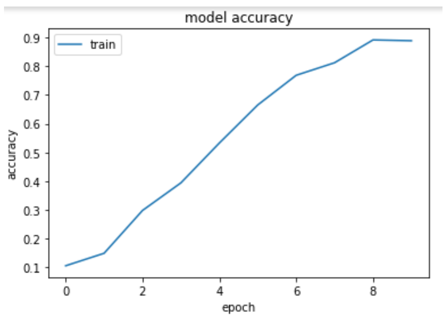
Case 3:

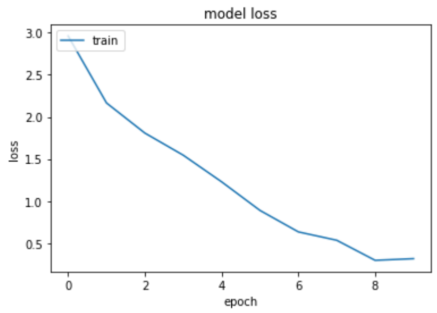
- Halved all filter sizes in convolution and dense layers.
- Learning rate: 1e-3

- Batch size: 128
- Epochs: 10
- Num Params: 17.3e6

Results:

- ~60 seconds per batch
- 87.7% train accuracy
- 88.5% test accuracy
- .3574 train loss
- 0.3369 test loss



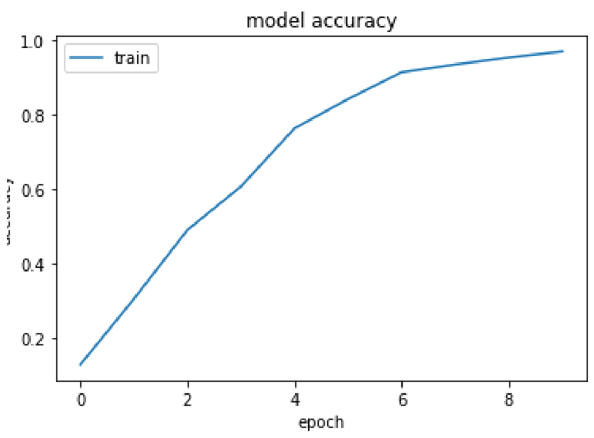


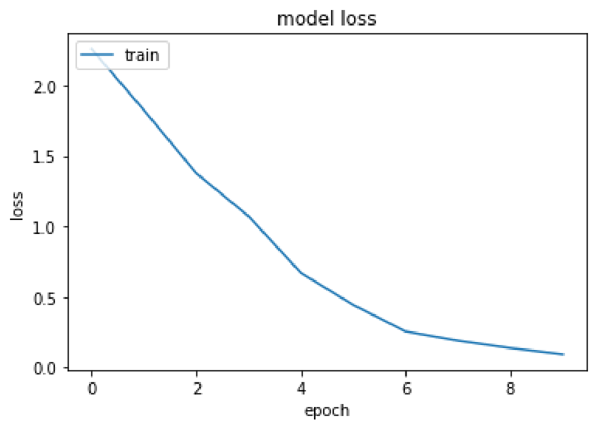
Case 4:

- Divided all filter sizes in convolution and dense layers by 4.
- Learning rate: 1e-3
- Batch size: 128
- Epochs: 10
- Num Params: 4.33e6

Results:

- ~150 seconds per batch
- 96.4% train accuracy
- 93.4% test accuracy
- .1052 train loss
- 0.2032 test loss



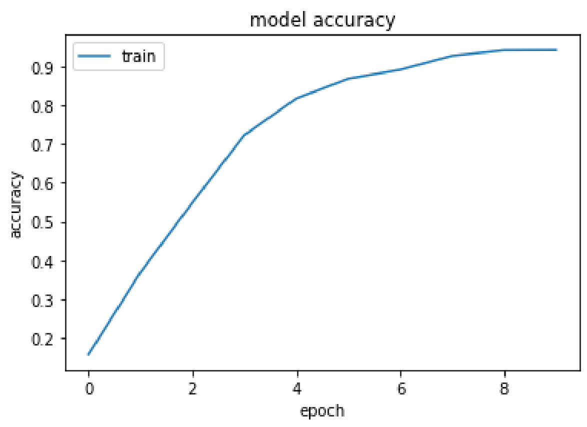


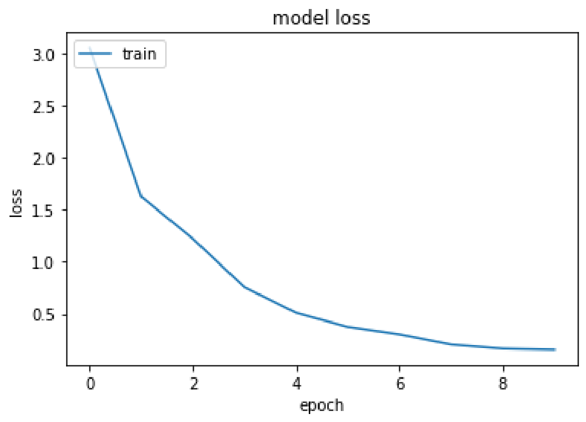
Case 5:

- Drastically changed the filter sizes. See model summary in index.
- Learning rate: 1e-3
- Batch size: 128
- Epochs: 10
- Num Params: 21.2e6

Results:

- ~490 seconds per batch
- 93.1% train accuracy
- 92.4% test accuracy
- .1875 train loss
- 0.2891 test loss



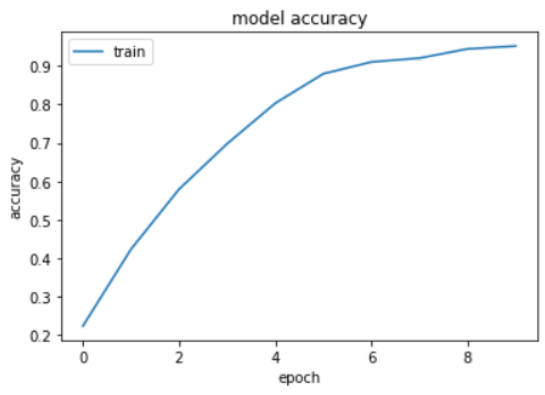


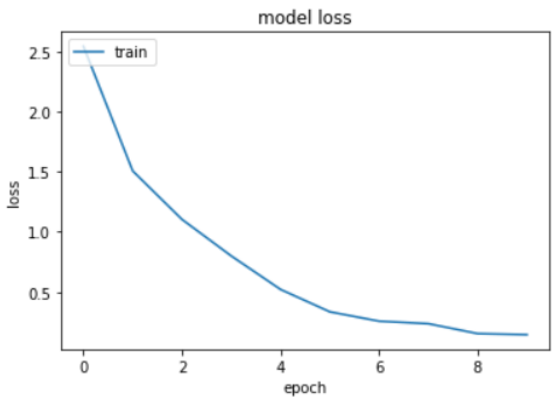
Case 6:

- Same summary as 1. Used batch normalization.
- Learning rate: 1e-3
- Batch size: 128
- Epochs: 10
- Num Params: 31.7e6 including 5000 non-trainable params

Results:

- ~600 seconds per batch
- 95.2% train accuracy
- 38.3% test accuracy
- .1458 train loss
- 12.92 test loss



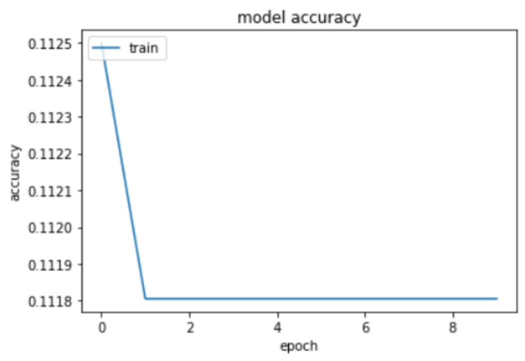


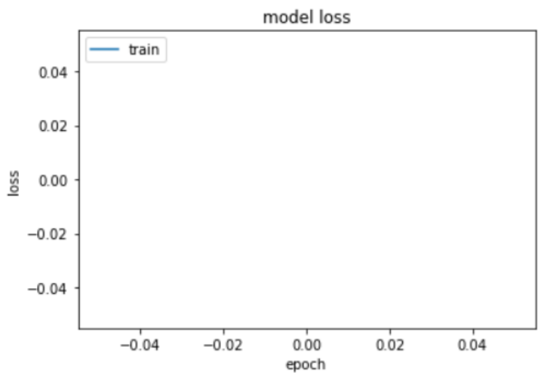
Case 7:

- Same summary as case 1
- Learning rate: 1e-1
- Batch size: 128
- Epochs: 10
- Num Params: 69.2e6

Results:

- ~460 seconds per batch
- 11.7% train accuracy
- 10.83% test accuracy
- nan train loss
- nan test loss





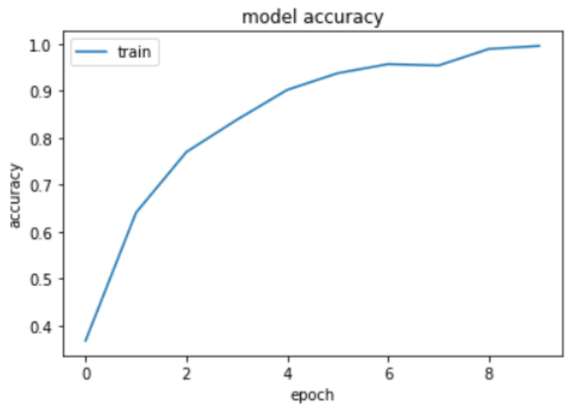
Case 8:

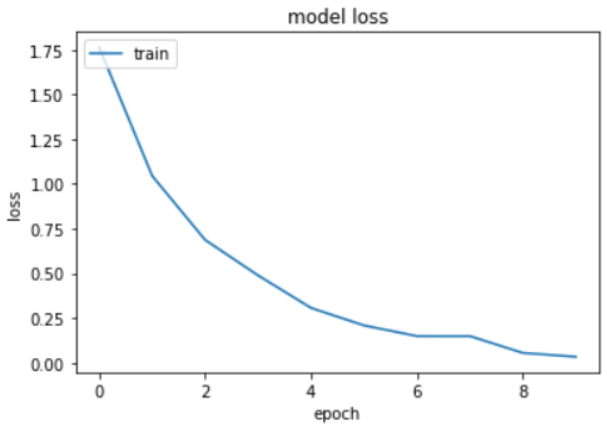
- Same summary as case 1

- Learning rate: 1e-5
- Batch size: 128
- Epochs: 10
- Num Params: 69.2e6

Results:

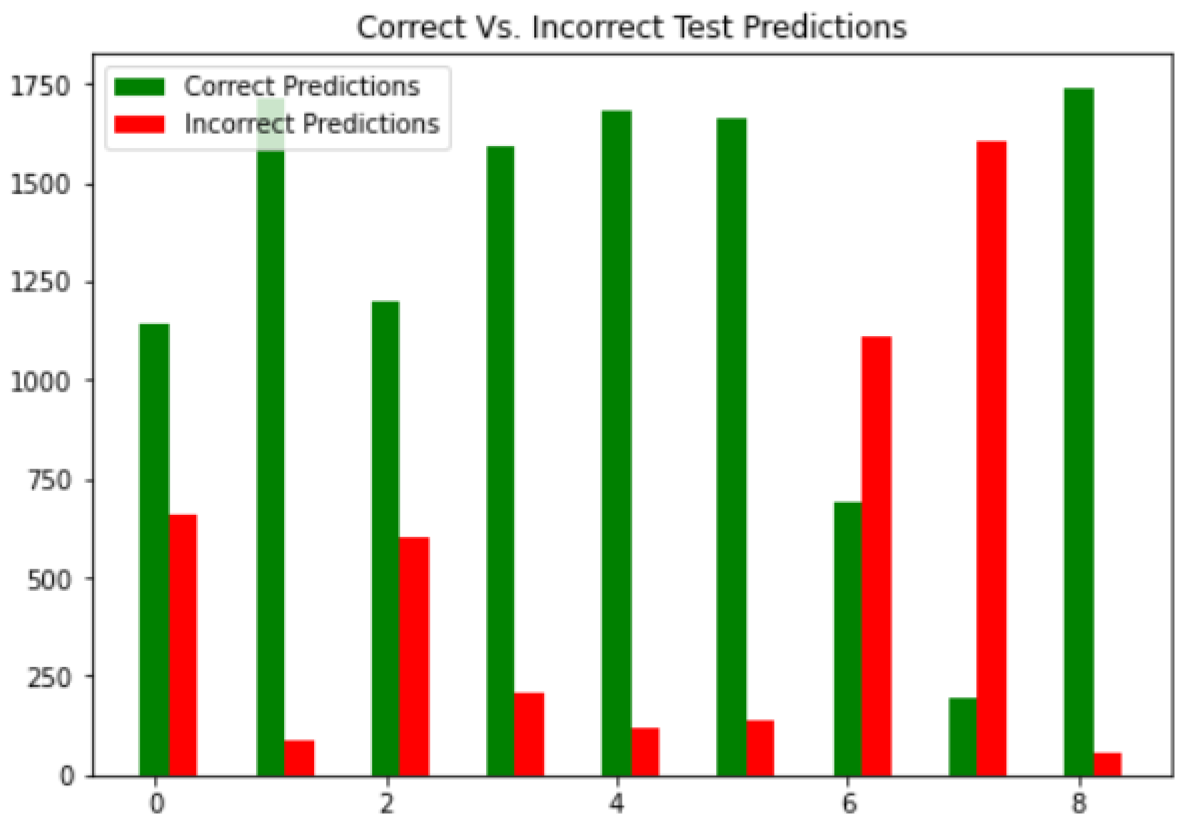
- ~450 seconds per batch
- 99.53% train accuracy
- 96.83% test accuracy
- .0367 train loss
- .1031 test loss





Test Results:

For each model, evaluated the results of the predictions and got the following chart. Note that the plot numbers correspond to the test cases shown above.



A Few Takeaways:

- The number of filters in the convolution and dense layers did result in a worse model as expected. The surprising part is that when I scaled down the filters by 2 (88% accuracy), I had far worse results then when I scaled them by 4 (96% accuracy). I assume this has to do with poor luck more than anything else.

- Learning rate makes an enormous difference when it comes to Deep Neural Networks. The results I received before and after the change were drastic for some trials. The first time I used a batch size of 32, the NN never came close to converging but after changing the learning rate, the results flew up to expected levels of accuracy. I had assumed that decreasing this parameter would cause the learning to drastically slow down, but it only caused a epoch rate 50 ms (11% higher.

- There is some luck when it comes to training. The accuracy of the model can differ with the same exact parameters and training set. This makes sense because gradient descent assumes that it is constantly in the lowest valley when it could potentially just be a local minimum.

- Batch normalization smoothed out the accuracy vs. epoch curve and had phenomenal results. It only worked well, however, when I decreased the learning rate.

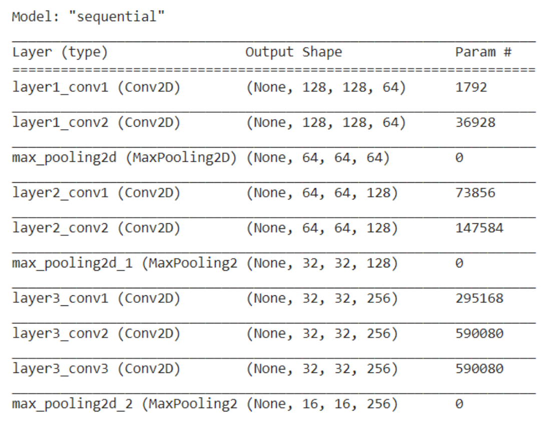
- In case 6, the train accuracy was good at 96% but the test results were horrendous (38%). I have no idea what caused this. It could be an error on my part, or it could be grossly overfitting the data somehow. This does not make much sense because batch normalization is a form of regularization so, if anything, it should be less prone to overfitting.

- ML libraries such as Tensorflow and Keras make this process a lot easier. This process seemed to be a lot more daunting than in ended up being.

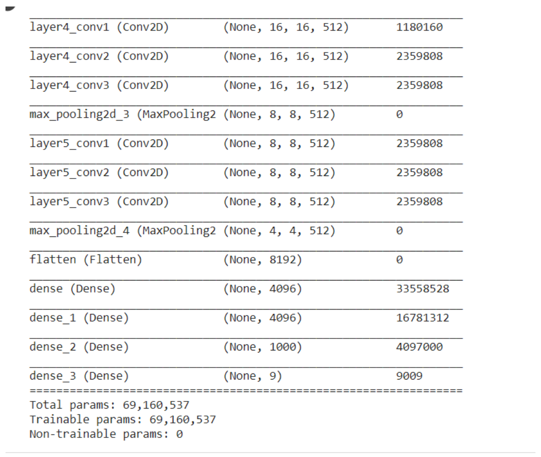
- The winner:
    - Learning rate of 1e-5 and with the use of batch normalization!

Model Summaries

Case 1:

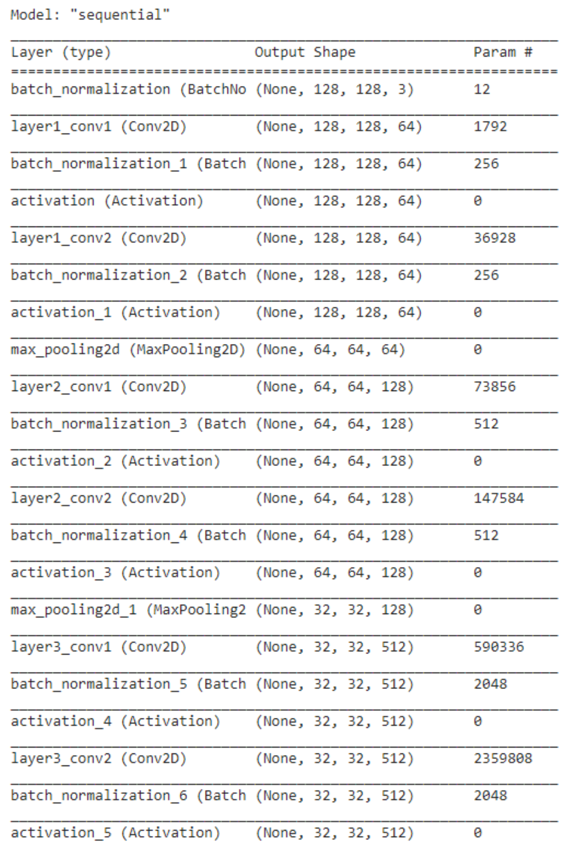


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 256)       295168
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 256)       590080
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 256)       0
```

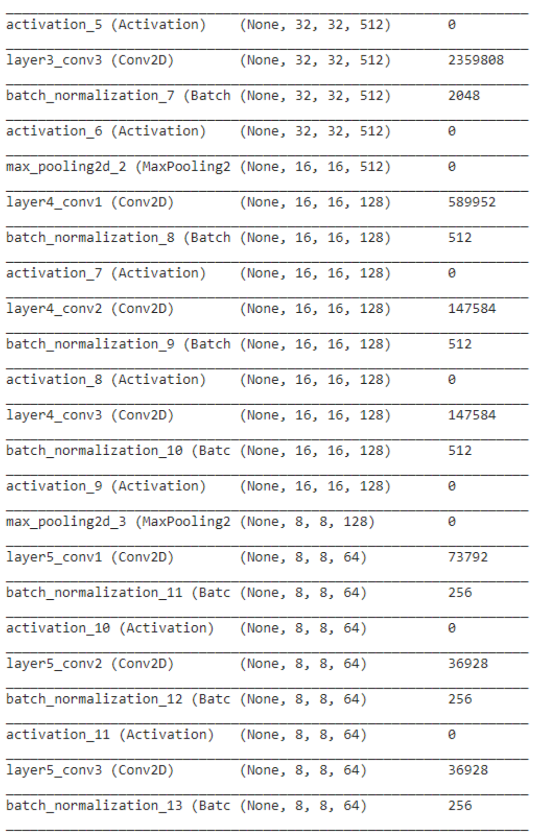


```
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 512)       1180160
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 512)       2359808
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 512)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 512)         2359808
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 512)         0
_________________________________________________________________
flatten (Flatten)            (None, 8192)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              33558528
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 9009
=================================================================
Total params: 69,160,537
Trainable params: 69,160,537
Non-trainable params: 0
```

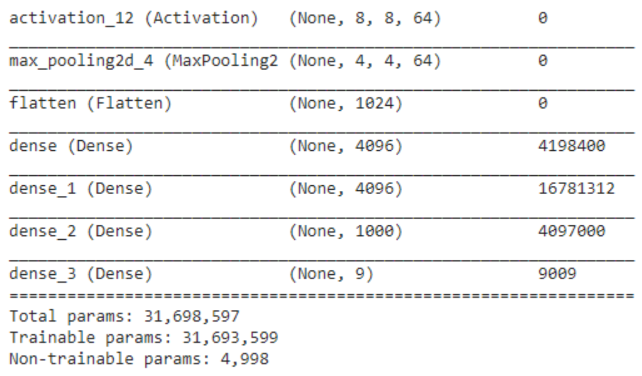
Case 2:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization (BatchNo (None, 128, 128, 3)       12
_________________________________________________________________
layer1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
activation (Activation)      (None, 128, 128, 64)      0
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 512)       590336
_________________________________________________________________
batch_normalization_5 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_4 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
batch_normalization_6 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_5 (Activation)    (None, 32, 32, 512)       0
```

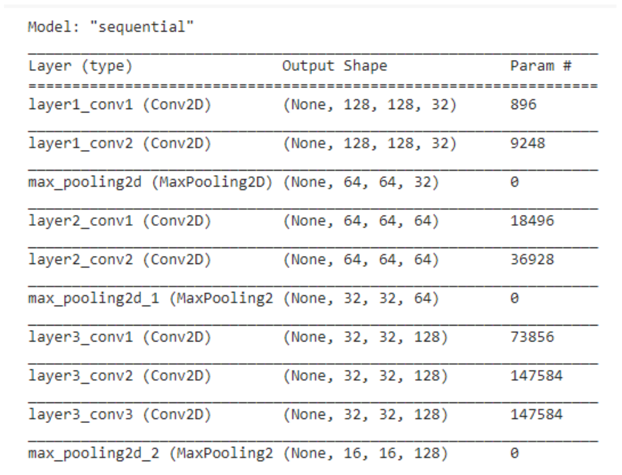


```
_________________________________________________________________
activation_5 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
batch_normalization_7 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_6 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 512)       0
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 128)       589952
_________________________________________________________________
batch_normalization_8 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_7 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_9 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_8 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_10 (Batc (None, 16, 16, 128)       512
_________________________________________________________________
activation_9 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 64)          73792
_________________________________________________________________
batch_normalization_11 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
activation_10 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_12 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
activation_11 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_13 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
```

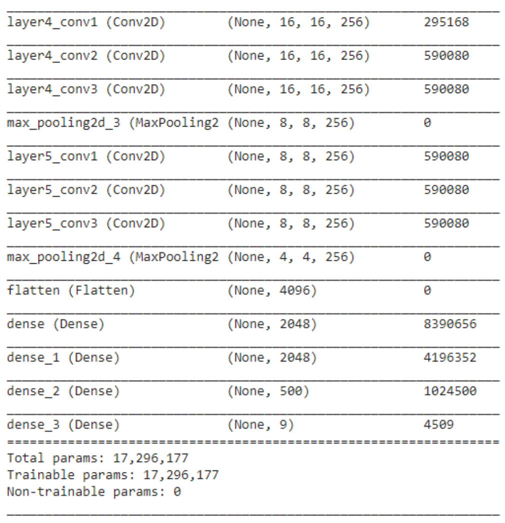


```
activation_12 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              4198400
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 9009
=================================================================
Total params: 31,698,597
Trainable params: 31,693,599
Non-trainable params: 4,998
```

Case 3:

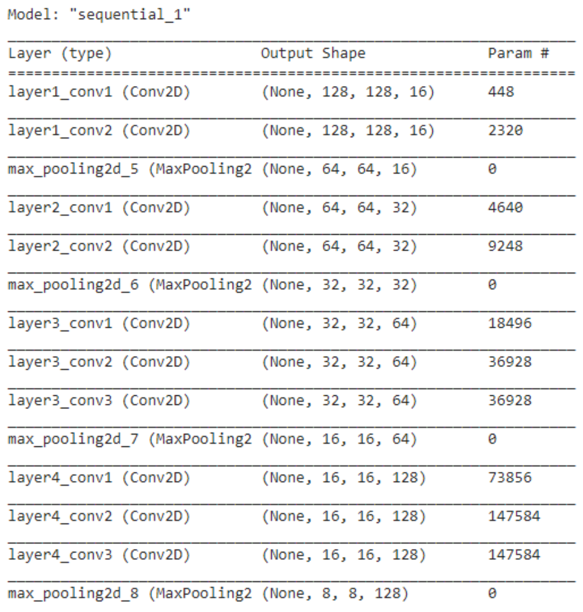


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv1 (Conv2D)        (None, 128, 128, 32)      896
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 32)      9248
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 32)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 64)        18496
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 64)        36928
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 64)        0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 128)       73856
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 128)       147584
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 128)       0
```

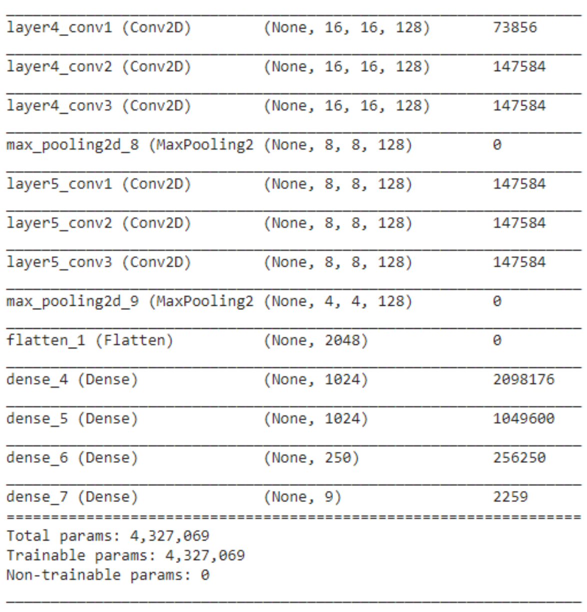


```
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 256)       295168
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 256)       590080
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 256)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 256)         590080
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 4096)              0
_________________________________________________________________
dense (Dense)                (None, 2048)              8390656
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              4196352
_________________________________________________________________
dense_2 (Dense)              (None, 500)               1024500
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 4509
=================================================================
Total params: 17,296,177
Trainable params: 17,296,177
Non-trainable params: 0
_________________________________________________________________
```

Case 4:

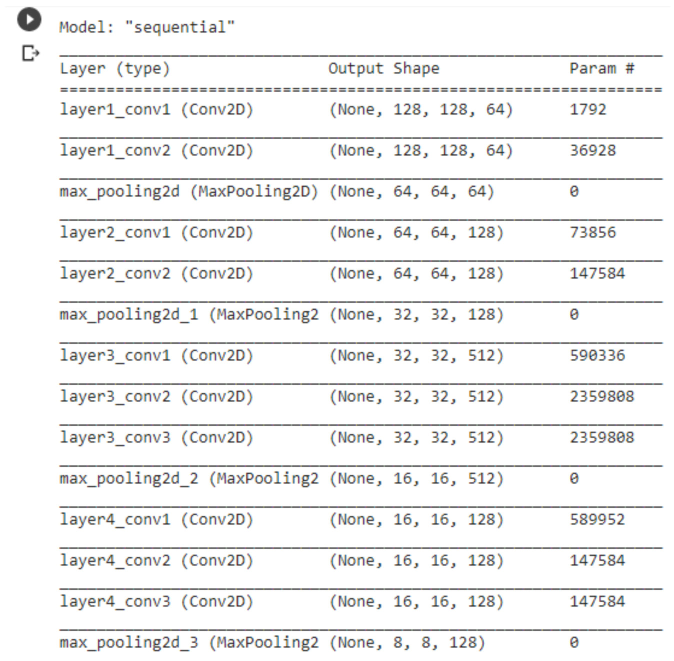


```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv1 (Conv2D)        (None, 128, 128, 16)      448
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 16)      2320
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 64, 64, 16)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 32)        4640
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 32)        9248
_________________________________________________________________
max_pooling2d_6 (MaxPooling2 (None, 32, 32, 32)        0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 64)        18496
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 64)        36928
_________________________________________________________________
max_pooling2d_7 (MaxPooling2 (None, 16, 16, 64)        0
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 128)         0
```

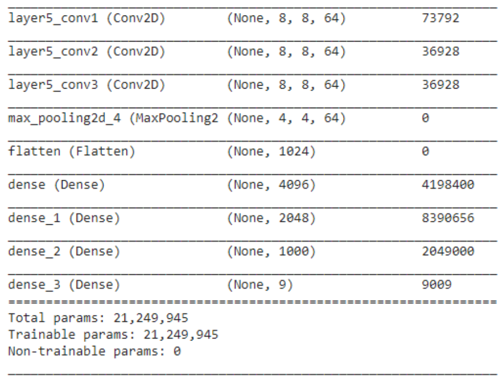


```
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 128)       73856
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_8 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 128)         147584
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 128)         147584
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 128)         147584
_________________________________________________________________
max_pooling2d_9 (MaxPooling2 (None, 4, 4, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 2048)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              2098176
_________________________________________________________________
dense_5 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dense_6 (Dense)              (None, 250)               256250
_________________________________________________________________
dense_7 (Dense)              (None, 9)                 2259
=================================================================
Total params: 4,327,069
Trainable params: 4,327,069
Non-trainable params: 0
_________________________________________________________________
```

Case 5:

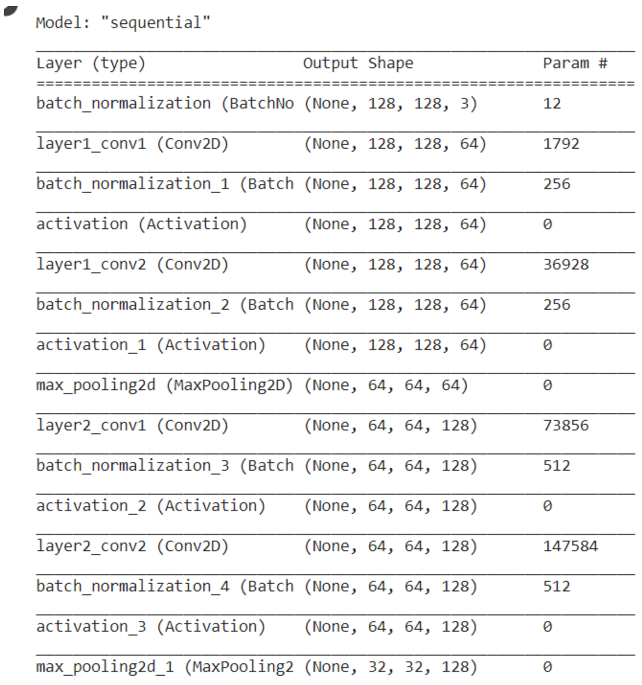


```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
layer1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 512)       590336
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 512)       0
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 128)       589952
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
```

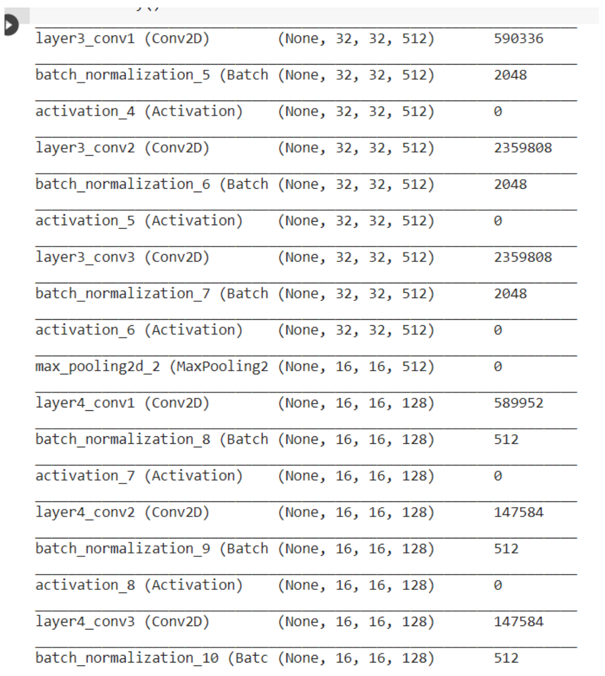


```
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 64)          73792
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              4198400
_________________________________________________________________
dense_1 (Dense)              (None, 2048)              8390656
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              2049000
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 9009
=================================================================
Total params: 21,249,945
Trainable params: 21,249,945
Non-trainable params: 0
_________________________________________________________________
```

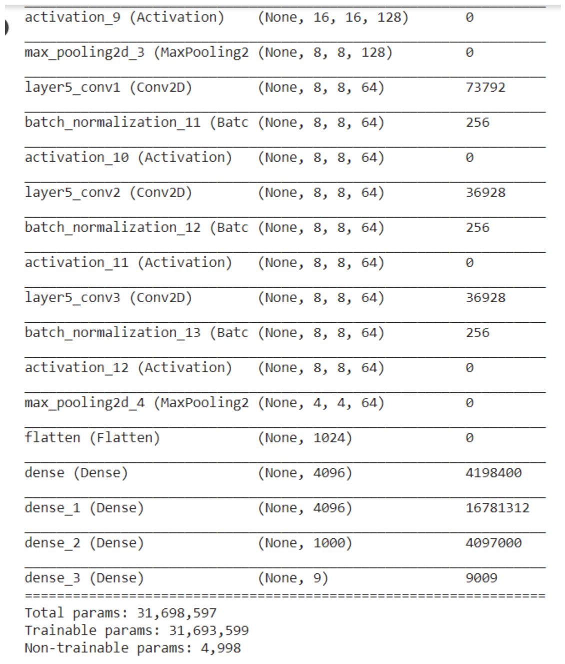
Case 6:



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
batch_normalization (BatchNo (None, 128, 128, 3)       12
_________________________________________________________________
layer1_conv1 (Conv2D)        (None, 128, 128, 64)      1792
_________________________________________________________________
batch_normalization_1 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
activation (Activation)      (None, 128, 128, 64)      0
_________________________________________________________________
layer1_conv2 (Conv2D)        (None, 128, 128, 64)      36928
_________________________________________________________________
batch_normalization_2 (Batch (None, 128, 128, 64)      256
_________________________________________________________________
activation_1 (Activation)    (None, 128, 128, 64)      0
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 64, 64, 64)        0
_________________________________________________________________
layer2_conv1 (Conv2D)        (None, 64, 64, 128)       73856
_________________________________________________________________
batch_normalization_3 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_2 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
layer2_conv2 (Conv2D)        (None, 64, 64, 128)       147584
_________________________________________________________________
batch_normalization_4 (Batch (None, 64, 64, 128)       512
_________________________________________________________________
activation_3 (Activation)    (None, 64, 64, 128)       0
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 32, 32, 128)       0
```



```
_________________________________________________________________
layer3_conv1 (Conv2D)        (None, 32, 32, 512)       590336
_________________________________________________________________
batch_normalization_5 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_4 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
layer3_conv2 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
batch_normalization_6 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_5 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
layer3_conv3 (Conv2D)        (None, 32, 32, 512)       2359808
_________________________________________________________________
batch_normalization_7 (Batch (None, 32, 32, 512)       2048
_________________________________________________________________
activation_6 (Activation)    (None, 32, 32, 512)       0
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 16, 16, 512)       0
_________________________________________________________________
layer4_conv1 (Conv2D)        (None, 16, 16, 128)       589952
_________________________________________________________________
batch_normalization_8 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_7 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
layer4_conv2 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_9 (Batch (None, 16, 16, 128)       512
_________________________________________________________________
activation_8 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
layer4_conv3 (Conv2D)        (None, 16, 16, 128)       147584
_________________________________________________________________
batch_normalization_10 (Batc (None, 16, 16, 128)       512
```



```
activation_9 (Activation)    (None, 16, 16, 128)       0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 8, 8, 128)         0
_________________________________________________________________
layer5_conv1 (Conv2D)        (None, 8, 8, 64)          73792
_________________________________________________________________
batch_normalization_11 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
activation_10 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
layer5_conv2 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_12 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
activation_11 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
layer5_conv3 (Conv2D)        (None, 8, 8, 64)          36928
_________________________________________________________________
batch_normalization_13 (Batc (None, 8, 8, 64)          256
_________________________________________________________________
activation_12 (Activation)   (None, 8, 8, 64)          0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 4, 4, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 4096)              4198400
_________________________________________________________________
dense_1 (Dense)              (None, 4096)              16781312
_________________________________________________________________
dense_2 (Dense)              (None, 1000)              4097000
_________________________________________________________________
dense_3 (Dense)              (None, 9)                 9009
=================================================================
Total params: 31,698,597
Trainable params: 31,693,599
Non-trainable params: 4,998
```

Case 7:

Same as Case 1

Case 8:

Same as Case 1

References

- Google Collab
- O.Ulucan, D.Karakaya, and M.Turkan.(2020) A large-scale dataset for fish segmentation and classification.In Conf. Innovations Intell. Syst. Appli. (ASYU)
- https://github.com/Hvass-Labs/TensorFlow-Tutorials/blob/master/03C_Keras_API.ipynb
- https://www.geeksforgeeks.org/vgg-16-cnn-model/
- https://www.tensorflow.org/guide/keras/save_and_serialize
- https://towardsdatascience.com/installing-tensorflow-with-cuda-cudnn-and-gpu-support-on-windows-10-60693e46e781
- https://stackoverflow.com/questions/53698035/failed-to-get-convolution-algorithm-this-is-probably-because-cudnn-failed-to-in