**Convolutional Neural Networks Assignment**

**4/1/2021**

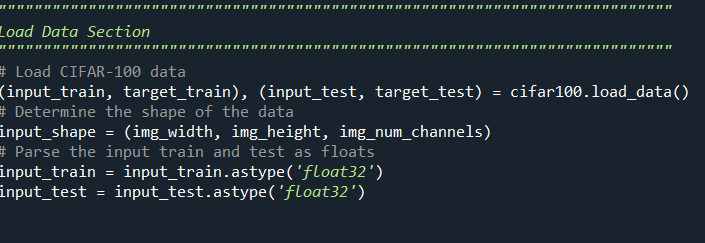
**Due at 11:59pm**

**Introduction:**

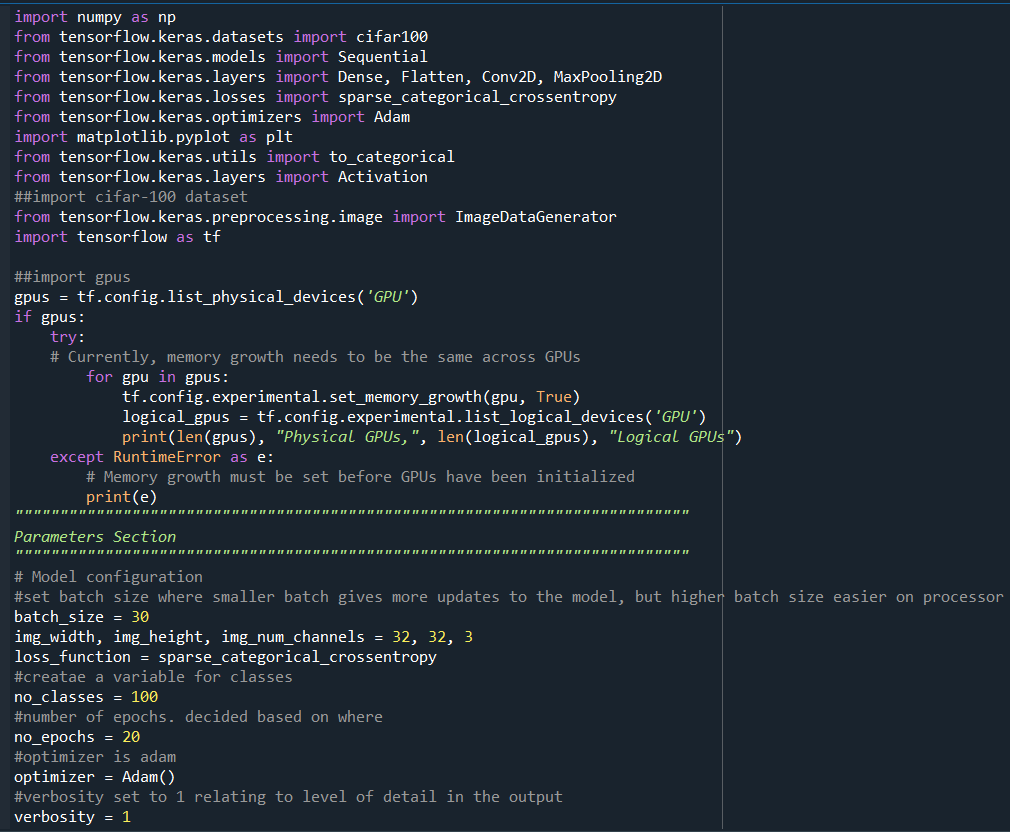
Cifar-10 and Cifar-100 are datasets that are commonly utilized when learning how to develop functional neural networks.  Both are made up of images and cifar-10 contains 10 different classes of images while cifar-100 contains 100 different classes.  Cifar-100 will be the focus of this assignment.  It should be noted that due to the greater number of classes the accuracy for models will be lower for cifar-100 than for cifar-10.  The objective is to make predictions of item classes with the cifar-100 dataset utilizing a convolutional neural network.  An initial model will be created (question 2) and then further experiments will be performed utilizing this model as a base with the goal of determining what changes improve the accuracy of the model.

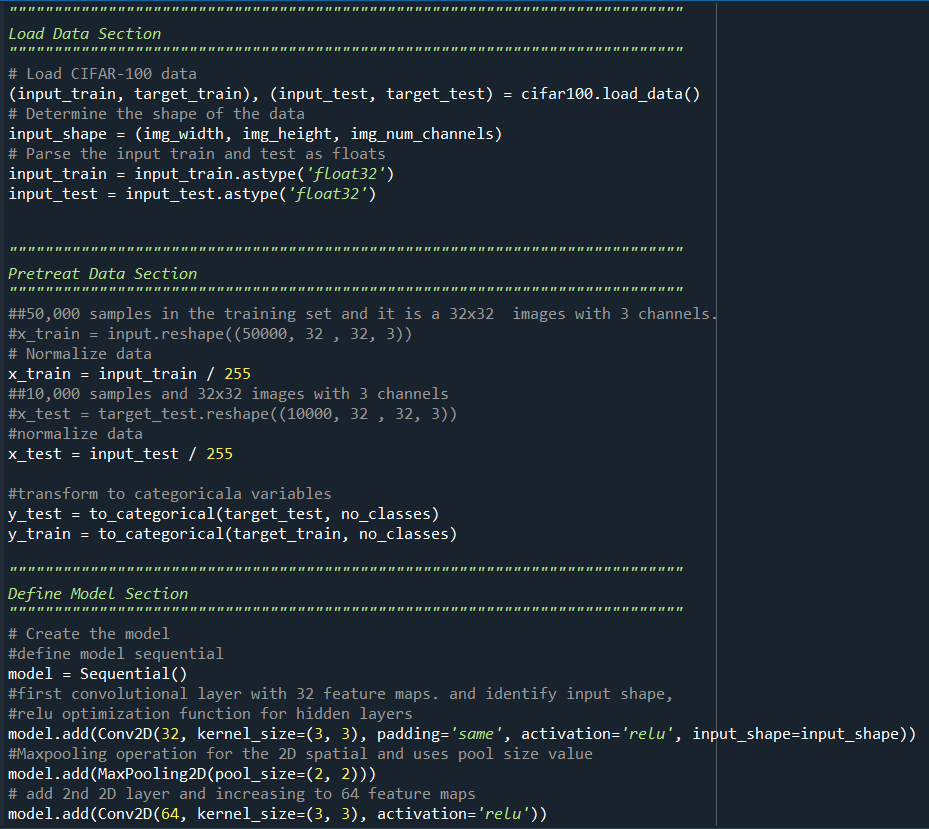
**Tasks and Experimentation:**

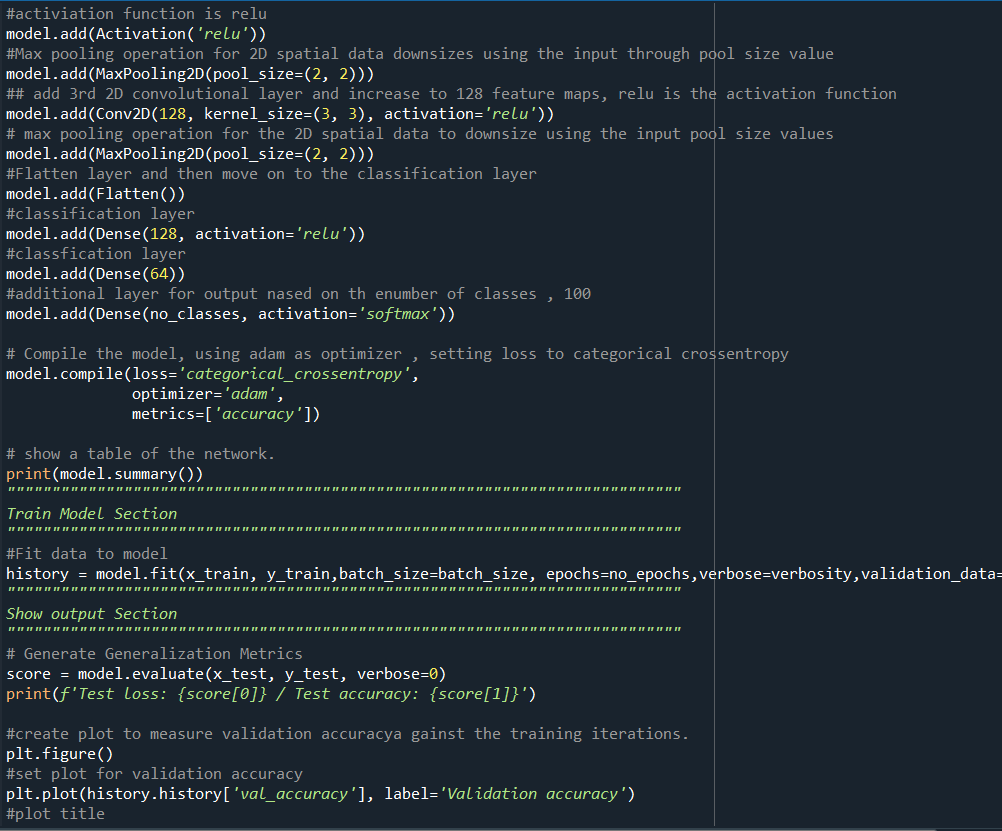
**Import CYFAR-10**

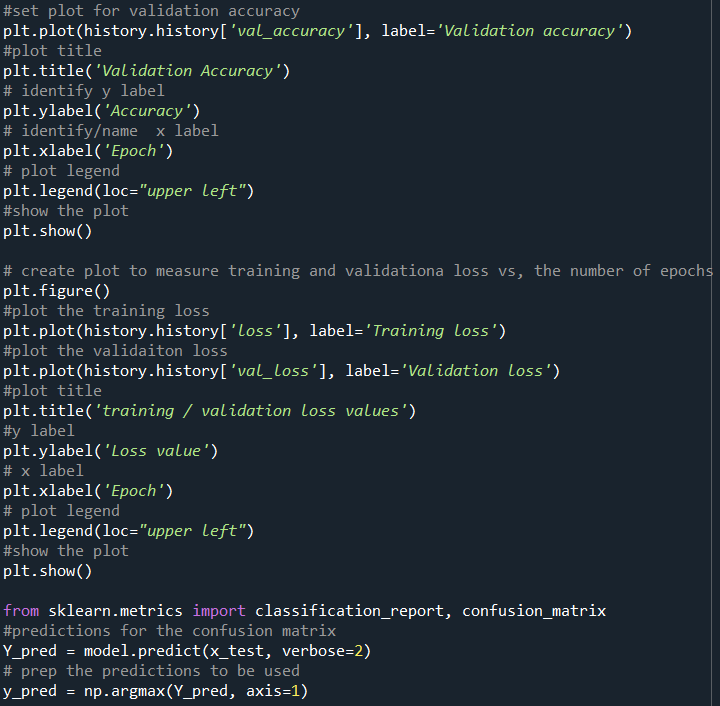


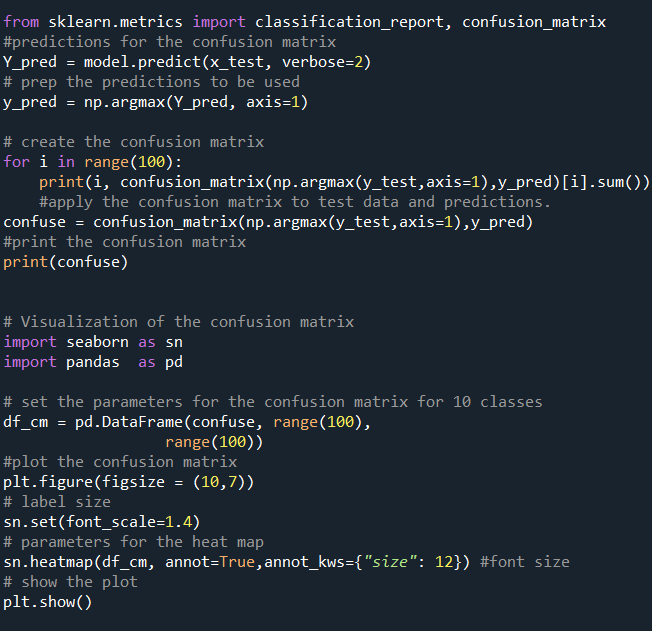
**Model 1:**











**Model 1 Training Procedure**

My training procedure began with importing the cifar-100 data set. I then set the parameters and the shape of the image. Early on, I realized that running larger epochs crashed my computer. Smaller batch sizes also were not compatible. I found improvement in my model between 10 and 18 epochs. With a batch size of 50. The initial model utilized the following parameters:

**Model 1 Table/ Summary**

conv2d\_6 (Conv2D)            (None, 32, 32, 32)        896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_6 (MaxPooling2 (None, 16, 16, 32)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_7 (Conv2D)            (None, 14, 14, 64)        18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation)    (None, 14, 14, 64)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_7 (MaxPooling2 (None, 7, 7, 64)          0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_8 (Conv2D)            (None, 5, 5, 128)         73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_8 (MaxPooling2 (None, 2, 2, 128)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_2 (Flatten)          (None, 512)               0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_7 (Dense)              (None, 128)               65664

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_8 (Dense)              (None, 64)                8256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_9 (Dense)              (None, 100)               6500

Total params: 173,668

Trainable params: 173,668

Non-trainable params: 0

Epoch 8/20

1667/1667 [==============================] - 20s 12ms/step - loss: 2.1515 - accuracy: 0.4313 - val\_loss: 2.4929 - val\_accuracy: 0.3734

Epoch 9/20

1667/1667 [==============================] - 21s 12ms/step - loss: 2.0689 - accuracy: 0.4427 - val\_loss: 2.4746 - val\_accuracy: 0.3732

Epoch 10/20

1667/1667 [==============================] - 19s 12ms/step - loss: 2.0005 - accuracy: 0.4613 - val\_loss: 2.4654 - val\_accuracy: 0.3821

Epoch 11/20

1667/1667 [==============================] - 20s 12ms/step - loss: 1.9326 - accuracy: 0.4783 - val\_loss: 2.4904 - val\_accuracy: 0.3743

Epoch 12/20

1667/1667 [==============================] - 19s 11ms/step - loss: 1.8815 - accuracy: 0.4861 - val\_loss: 2.5810 - val\_accuracy: 0.3687

Epoch 13/20

1667/1667 [==============================] - 20s 12ms/step - loss: 1.8155 - accuracy: 0.5027 - val\_loss: 2.5498 - val\_accuracy: 0.3800

Epoch 14/20

1667/1667 [==============================] - 20s 12ms/step - loss: 1.7385 - accuracy: 0.5204 - val\_loss: 2.5839 - val\_accuracy: 0.3753

Epoch 15/20

1667/1667 [==============================] - 20s 12ms/step - loss: 1.7096 - accuracy: 0.5272 - val\_loss: 2.5450 - val\_accuracy: 0.3857

Epoch 16/20

1667/1667 [==============================] - 22s 13ms/step - loss: 1.6486 - accuracy: 0.5397 - val\_loss: 2.6093 - val\_accuracy: 0.3776

Epoch 17/20

1667/1667 [==============================] - 19s 12ms/step - loss: 1.6073 - accuracy: 0.5510 - val\_loss: 2.6487 - val\_accuracy: 0.3739

Epoch 18/20

1667/1667 [==============================] - 18s 11ms/step - loss: 1.5618 - accuracy: 0.5598 - val\_loss: 2.6427 - val\_accuracy: 0.3795

Epoch 19/20

1667/1667 [==============================] - 21s 13ms/step - loss: 1.5241 - accuracy: 0.5677 - val\_loss: 2.7095 - val\_accuracy: 0.3725

Epoch 20/20

1667/1667 [==============================] - 21s 13ms/step - loss: 1.4768 - accuracy: 0.5786 - val\_loss: 2.7443 - val\_accuracy: 0.3746

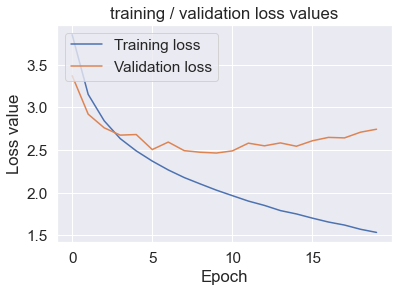
Test loss: 2.744290351867676 / Test accuracy: 0.37459999322891235

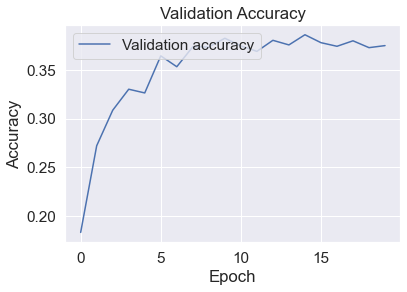
313/313 - 1s

**Model 1: Summary**

Next, I loaded the training and test sets to the model to train my model. This is based on the parameters. Below contains graphs of the Training and Validation Loss and the Classification Accuracy on the test set vs. training iterations. These plots helped visualize the training process and overfitting/underfitting on my models which I used to change parameters, improve my CNN layers, and compare augmentation.

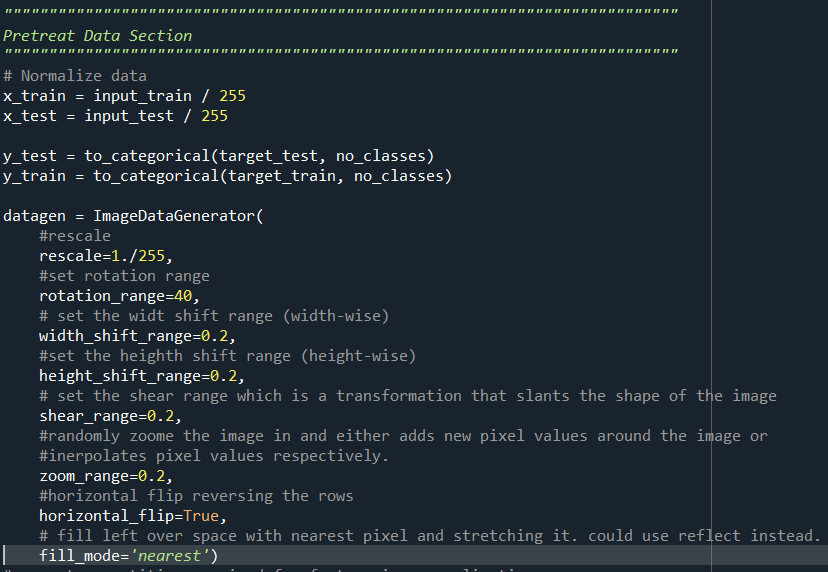
**Model 1: Plots**

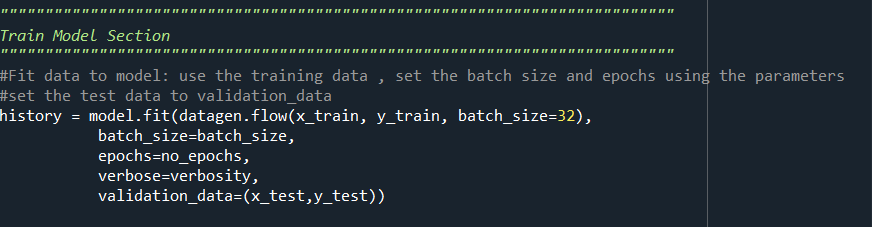




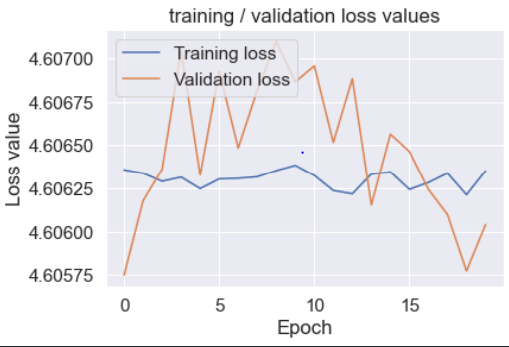
I tried SGD, RMSprop, and Adam. The problem with SGD was that I wanted a smaller learning rate which required higher epochs around the 100s range, meaning a higher runtime. I then decided to work with RMSprop and Adam. Although I chose Adam, RMSprop and Adam seemed to give similar results. Model 1 produced a Test loss of 2.638 and Test accuracy of .38.

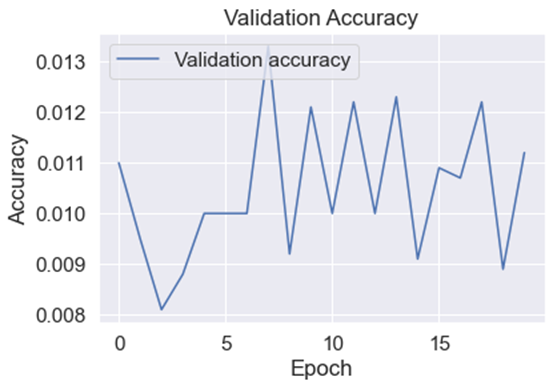
**Model 2: Data Augmentation**





**Model 2 Plots:**





These changes were done to utilize data augmentation. This creates noise into the images due to having a high number of parameters in my model. Flips, translations, and reflections can create minor changes. I also analyzed height and width range as well as rotation range. Altering these numbers only identified what made my model worse. Nothing in Model 2 was significantly improved through data augmentation. I will elaborate on this in later steps.

**Model 2 Parameters:**

Model two’s training process is shown in the log below.  The main parameter difference was the code utilized to perform data augmentation by introducing noise into the model.

**Model 2 Partial Training Log**

Epoch 14/20

1563/1563 [==============================] - 126s 80ms/step - loss: 4.6060 - accuracy: 0.0103 - val\_loss: 4.6062 - val\_accuracy: 0.0123

Epoch 15/20

1563/1563 [==============================] - 129s 83ms/step - loss: 4.6060 - accuracy: 0.0092 - val\_loss: 4.6066 - val\_accuracy: 0.0091

Epoch 16/20

1563/1563 [==============================] - 126s 80ms/step - loss: 4.6059 - accuracy: 0.0088 - val\_loss: 4.6065 - val\_accuracy: 0.0109

Epoch 17/20

1563/1563 [==============================] - 123s 79ms/step - loss: 4.6060 - accuracy: 0.0092 - val\_loss: 4.6062 - val\_accuracy: 0.0107

Epoch 18/20

1563/1563 [==============================] - 114s 73ms/step - loss: 4.6062 - accuracy: 0.0096 - val\_loss: 4.6061 - val\_accuracy: 0.0122

Epoch 19/20

1563/1563 [==============================] - 116s 74ms/step - loss: 4.6061 - accuracy: 0.0101 - val\_loss: 4.6058 - val\_accuracy: 0.0089

Epoch 20/20

1563/1563 [==============================] - 113s 72ms/step - loss: 4.6061 - accuracy: 0.0089 - val\_loss: 4.6060 - val\_accuracy: 0.0112

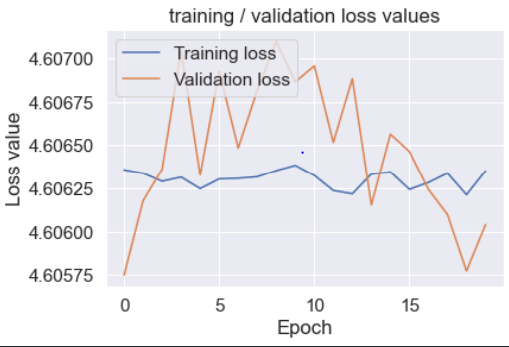
**Model 2 Test Accuracy Vs. Loss:**

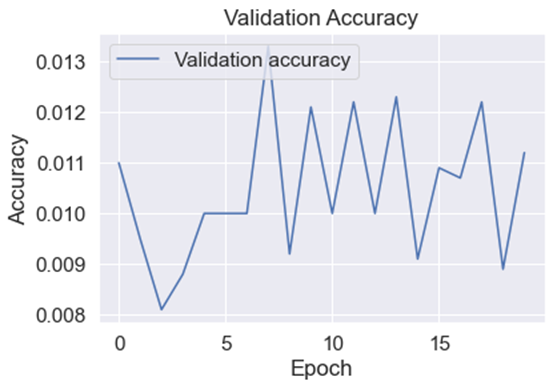
**Test loss: 4.606041431427002 / Test accuracy: 0.01119999960064888**

**Model 2: Summary**

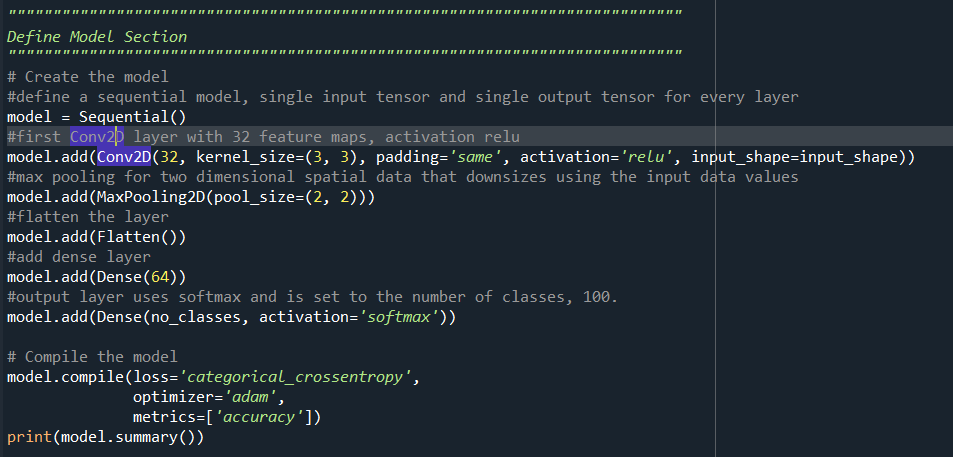
Model 2 Data augmentation gave poor validation accuracy of .0111 and a high test loss of 4.6. This proves to be a weak model as well as overfitting. I tried multiple changes within the data augmentation, and nothing seemed to improve my model. This could have potentially improved by using SGD and setting a low learning rate, but that would run into run time issues. The model with data augmentation performed weaker than Model 1 overall, but it seems to improve the model’s training speed.

**Model 2:  Plots**





**Model 3: Only One Hidden Convolutional Layer**



**Model 3 : Parameters / Summary**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape          Param #

=================================================================

conv2d (Conv2D)              (None, 32, 32, 32)    896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 16, 16, 32)    0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten)            (None, 8192)          0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense)                (None, 64)                524352

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense)              (None, 100)               6500

=================================================================

Total params: 531,748

Trainable params: 531,748

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Model 3 Partial Training Log**

Epoch 16/20

1563/1563 [==============================] - 28s 18ms/step - loss: 2.8912 - accuracy: 0.2918 - val\_loss: 4.4247 - val\_accuracy: 0.0411

Epoch 17/20

1563/1563 [==============================] - 31s 20ms/step - loss: 2.8715 - accuracy: 0.2943 - val\_loss: 4.6609 - val\_accuracy: 0.0416

Epoch 18/20

1563/1563 [==============================] - 28s 18ms/step - loss: 2.8680 - accuracy: 0.2951 - val\_loss: 4.5914 - val\_accuracy: 0.0409

Epoch 19/20

1563/1563 [==============================] - 29s 18ms/step - loss: 2.8538 - accuracy: 0.2984 - val\_loss: 4.4786 - val\_accuracy: 0.0577

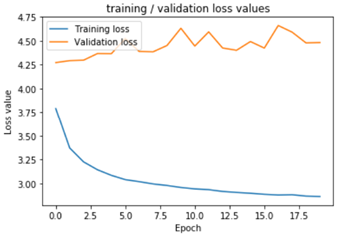
Epoch 20/20

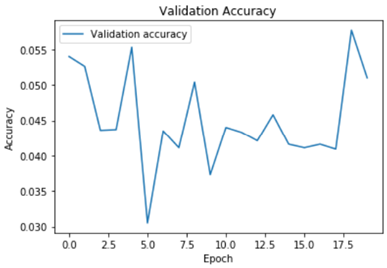
1563/1563 [==============================] - 29s 18ms/step - loss: 2.8597 - accuracy: 0.2936 - val\_loss: 4.4826 - val\_accuracy: 0.0510

**Test loss: 4.482575416564941 / Test accuracy: 0.050999999046325684**

313/313 - 1s

**Model 3 Plots:**

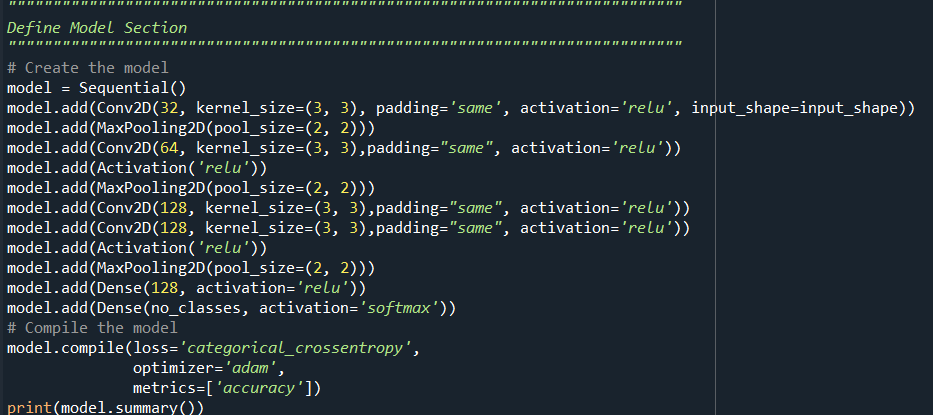




**Model 3: Summary**

Model 3 performed slightly better than Model 2, but they were both really bad. I assume it is due to data augmentation. The training accuracy for this model was a little better than training accuracy from Model 2. Overall, there are benefits to adding layers, but it ultimately depends on the complexity on the training set and other parameters such as batch size and epochs, and I say this because I read that for better fitted data, you would set batch size in the fitted model to batch\_size/epochs. Still, models should have depth for better performance.

**Model 4: One Added Hidden Layer than the First Model**



**Model 4: Parameters**

a.      Train your (fourth) new network.

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape          Param #

=================================================================

conv2d (Conv2D)              (None, 32, 32, 32)    896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D)            (None, 32, 32, 32)        9248

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 16, 16, 32)    0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D)            (None, 14, 14, 64)    18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation (Activation)      (None, 14, 14, 64)    0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 7, 7, 64)     0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D)            (None, 5, 5, 128)         73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 2, 2, 128)     0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten)            (None, 512)           0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense)                (None, 128)               65664

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense)              (None, 64)                8256

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense)              (None, 100)           6500

=================================================================

Total params: 182,916

Trainable params: 182,916

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

**Model 4: Partial Training Log**

Epoch 8/18

1667/1667 [==============================] - 33s 20ms/step - loss: 1.9890 - accuracy: 0.4600 - val\_loss: 2.4331 - val\_accuracy: 0.3820

Epoch 9/18

1667/1667 [==============================] - 41s 25ms/step - loss: 1.8960 - accuracy: 0.4843 - val\_loss: 2.4742 - val\_accuracy: 0.3813

Epoch 10/18

1667/1667 [==============================] - 32s 19ms/step - loss: 1.8170 - accuracy: 0.5033 - val\_loss: 2.5303 - val\_accuracy: 0.3750

Epoch 11/18

1667/1667 [==============================] - 39s 24ms/step - loss: 1.7411 - accuracy: 0.5151 - val\_loss: 2.5299 - val\_accuracy: 0.3774

Epoch 12/18

1667/1667 [==============================] - 35s 21ms/step - loss: 1.6784 - accuracy: 0.5355 - val\_loss: 2.5325 - val\_accuracy: 0.3919

Epoch 13/18

1667/1667 [==============================] - 40s 24ms/step - loss: 1.5921 - accuracy: 0.5514 - val\_loss: 2.5492 - val\_accuracy: 0.3880

Epoch 14/18

1667/1667 [==============================] - 44s 26ms/step - loss: 1.5305 - accuracy: 0.5645 - val\_loss: 2.6235 - val\_accuracy: 0.3897

Epoch 15/18

1667/1667 [==============================] - 45s 27ms/step - loss: 1.4601 - accuracy: 0.5852 - val\_loss: 2.6304 - val\_accuracy: 0.3816

Epoch 16/18

1667/1667 [==============================] - 45s 27ms/step - loss: 1.4166 - accuracy: 0.5930 - val\_loss: 2.7181 - val\_accuracy: 0.3855

Epoch 17/18

1667/1667 [==============================] - 46s 28ms/step - loss: 1.3600 - accuracy: 0.6054 - val\_loss: 2.7269 - val\_accuracy: 0.3885A: 28s - loss: 1.2938 - accuracy: 0.6246

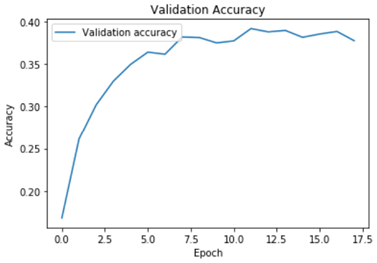
Epoch 18/18

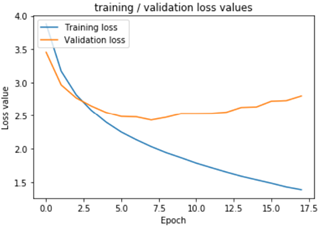
1667/1667 [==============================] - 51s 31ms/step - loss: 1.3125 - accuracy: 0.6181 - val\_loss: 2.7969 - **val\_accuracy: 0.3776TA**: 41s - loss: 1.2533 - accuracy: 0.6326 506/1667 [========>.....................] - ETA: 41s - loss: 1.2534 - accuracy: 0.6326 508/1667 [========>.....................] - ETA: 41s - loss: 1.2535 - accuracy: 0.6326

Test loss: 2.7969229221343994 / Test accuracy: 0.3776000142097473

313/313 - 2s

**Model 4 Plots:**





**Model 4: Summary**

Model 4 performed slightly worse than model one with validation accuracy of .377 and validation loss of 2.79 compared to the Model 1 .38 and 2.638 comparatively. This model did outperform the other models due to hidden layers. I could have removed the data augmentation for better accuracy as well. The additional layer did not seem to have a significant impact.

**Model 5 Dropout with the noise:**

This model was horrible, and I decided to remove the noise to better analyze dropout. Attached are a partial training log of this data.

**Partial Training Log Model 5 Dropout with the Noise:**

Epoch 16/20

1563/1563 [==============================] - 126s 80ms/step - loss: 4.6059 - accuracy: 0.0088 - val\_loss: 4.6065 - val\_accuracy: 0.0109

Epoch 17/20

1563/1563 [==============================] - 123s 79ms/step - loss: 4.6060 - accuracy: 0.0092 - val\_loss: 4.6062 - val\_accuracy: 0.0107

Epoch 18/20

1563/1563 [==============================] - 114s 73ms/step - loss: 4.6062 - accuracy: 0.0096 - val\_loss: 4.6061 - val\_accuracy: 0.0122

Epoch 19/20

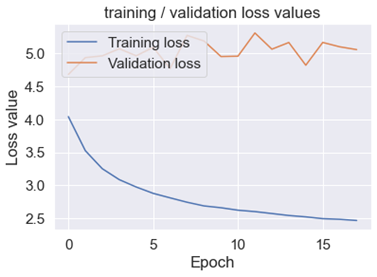
1563/1563 [==============================] - 116s 74ms/step - loss: 4.6061 - accuracy: 0.0101 - val\_loss: 4.6058 - val\_accuracy: 0.0089

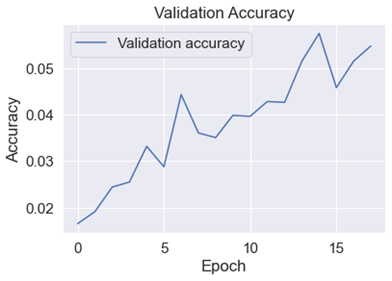
Epoch 20/20

1563/1563 [==============================] - 113s 72ms/step - loss: 4.6061 - **accuracy: 0.0089 - val\_loss: 4.6060 - val\_accuracy: 0.0112**

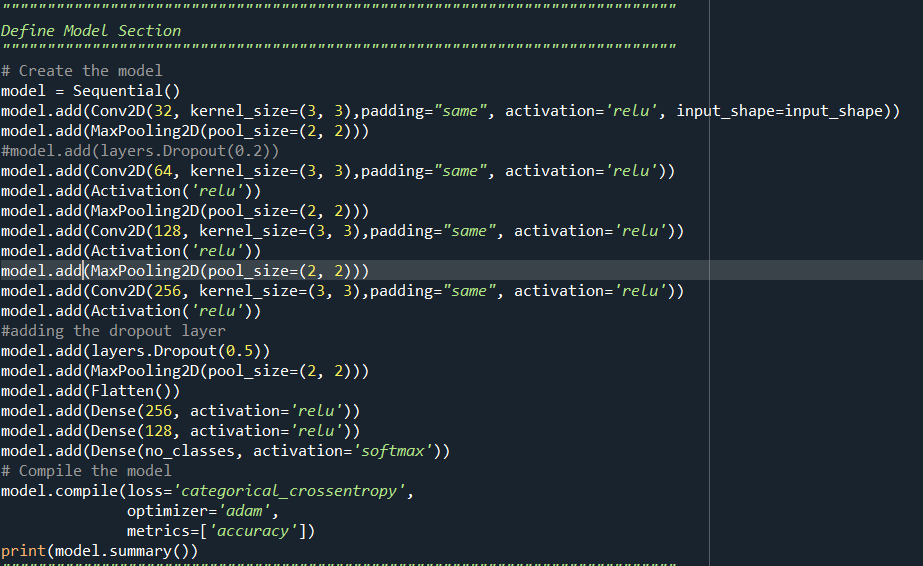
**Test loss: 4.606041431427002 / Test accuracy: 0.01119999960064888**

**Model 5 Dropout with the Noise: Plots**

****

****

**Model 5 Without Noise and only one Dropout .5:**

****

**Model 5 Without Noise and One Dropout .5 : Parameters**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape              Param #

=================================================================

conv2d (Conv2D)              (None, 32, 32, 32)        896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d (MaxPooling2D) (None, 16, 16, 32)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_1 (Conv2D)            (None, 16, 16, 64)        18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation (Activation)      (None, 16, 16, 64)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_1 (MaxPooling2 (None, 8, 8, 64)          0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_2 (Conv2D)            (None, 8, 8, 128)         73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_1 (Activation)    (None, 8, 8, 128)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_2 (MaxPooling2 (None, 4, 4, 128)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_3 (Conv2D)            (None, 4, 4, 256)         295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_2 (Activation)    (None, 4, 4, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout)            (None, 4, 4, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_3 (MaxPooling2 (None, 2, 2, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten (Flatten)            (None, 1024)              0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense (Dense)                (None, 256)               262400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_1 (Dense)              (None, 128)               32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_2 (Dense)              (None, 100)               12900

=================================================================

Total params: 696,612

Trainable params: 696,612

Non-trainable params: 0

**Model 5 Without Noise and One Dropout .5: Partial Training Log**

Epoch 13/18

1000/1000 [==============================] - 31s 31ms/step - loss: 1.6980 - accuracy: 0.5214 - val\_loss: 2.2672 - val\_accuracy: 0.4197

Epoch 14/18

1000/1000 [==============================] - 37s 37ms/step - loss: 1.6432 - accuracy: 0.5365 - val\_loss: 2.2638 - val\_accuracy: 0.4173

Epoch 15/18

1000/1000 [==============================] - 38s 38ms/step - loss: 1.5913 - accuracy: 0.5485 - val\_loss: 2.2349 - val\_accuracy: 0.4259

Epoch 16/18

1000/1000 [==============================] - 34s 34ms/step - loss: 1.5378 - accuracy: 0.5611 - val\_loss: 2.2422 - val\_accuracy: 0.4239

Epoch 17/18

1000/1000 [==============================] - 33s 33ms/step - loss: 1.4877 - accuracy: 0.5713 - val\_loss: 2.2565 - val\_accuracy: 0.4266

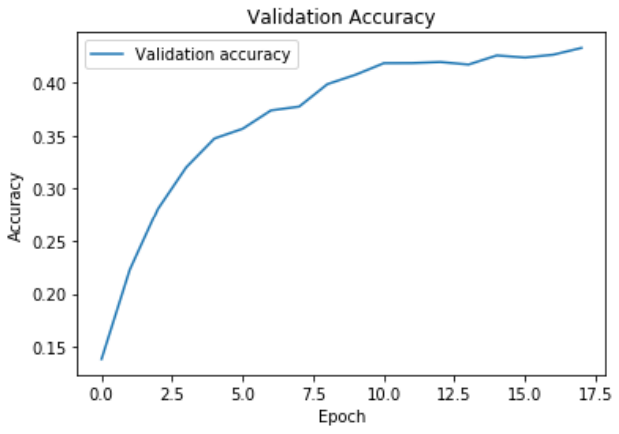
Epoch 18/18

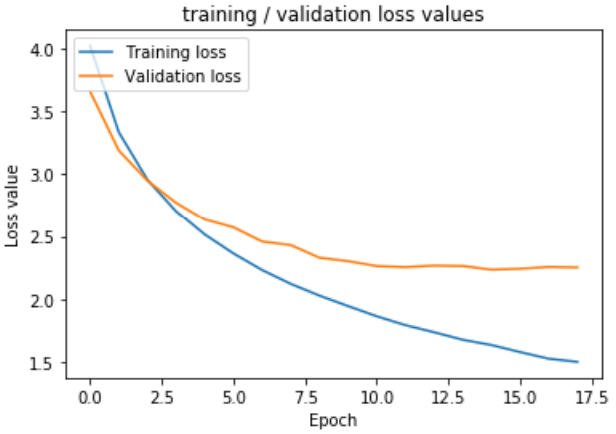
1000/1000 [==============================] - 34s 34ms/step - loss: 1.4673 - accuracy: 0.5766 - val\_loss: 2.2524 - val\_accuracy: 0.4330

**Test loss: 2.252423048019409 / Test accuracy: 0.43299999833106995**

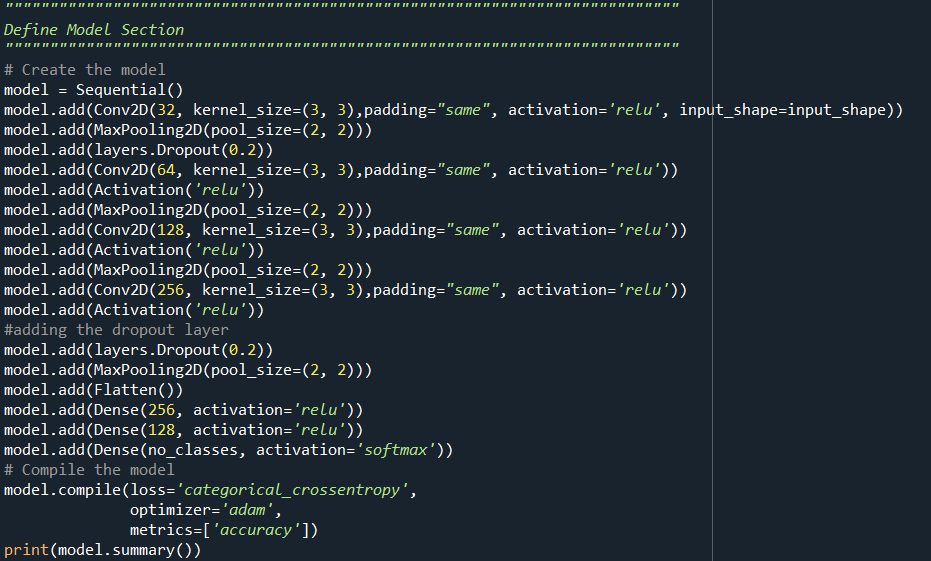
**313/313 - 3s**

**Model 5 Without Noise and One Dropout .5: Plots**





**Model 5 Two .2 Dropouts and No Data Augmentation: Code**

****

**Model 5 Two .2 Dropouts and No Data Augmentation: Parameters**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type)                 Output Shape              Param #

=================================================================

conv2d\_12 (Conv2D)           (None, 32, 32, 32)        896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_12 (MaxPooling (None, 16, 16, 32)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout (Dropout)            (None, 16, 16, 32)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_13 (Conv2D)           (None, 16, 16, 64)        18496

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_4 (Activation)    (None, 16, 16, 64)        0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_13 (MaxPooling (None, 8, 8, 64)          0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_14 (Conv2D)           (None, 8, 8, 128)         73856

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_5 (Activation)    (None, 8, 8, 128)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_14 (MaxPooling (None, 4, 4, 128)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

conv2d\_15 (Conv2D)           (None, 4, 4, 256)         295168

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

activation\_6 (Activation)    (None, 4, 4, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dropout\_1 (Dropout)          (None, 4, 4, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

max\_pooling2d\_15 (MaxPooling (None, 2, 2, 256)         0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

flatten\_4 (Flatten)          (None, 1024)              0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_12 (Dense)             (None, 256)               262400

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_13 (Dense)             (None, 128)               32896

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

dense\_14 (Dense)             (None, 100)               12900

=================================================================

Total params: 696,612

Trainable params: 696,612

Non-trainable params: 0

**Model 5 Two .2 Dropouts and No Data Augmentation: Partial Training Log**

Epoch 12/18

1000/1000 [==============================] - 15s 15ms/step - loss: 1.6847 - accuracy: 0.5269 - val\_loss: 2.4395 - val\_accuracy: 0.3915

Epoch 13/18

1000/1000 [==============================] - 15s 15ms/step - loss: 1.6248 - accuracy: 0.5408 - val\_loss: 2.3148 - val\_accuracy: 0.4069

Epoch 14/18

1000/1000 [==============================] - 15s 15ms/step - loss: 1.5716 - accuracy: 0.5531 - val\_loss: 2.3283 - val\_accuracy: 0.4124

Epoch 15/18

1000/1000 [==============================] - 17s 17ms/step - loss: 1.5274 - accuracy: 0.5673 - val\_loss: 2.3859 - val\_accuracy: 0.4053

Epoch 16/18

1000/1000 [==============================] - 16s 16ms/step - loss: 1.4581 - accuracy: 0.5800 - val\_loss: 2.5620 - val\_accuracy: 0.3837

Epoch 17/18

1000/1000 [==============================] - 17s 17ms/step - loss: 1.4180 - accuracy: 0.5940 - val\_loss: 2.4508 - val\_accuracy: 0.4116

Epoch 18/18

1000/1000 [==============================] - 18s 18ms/step - loss: 1.3610 - accuracy: **0.6092 - val\_loss: 2.3665 - val\_accuracy: 0.4214**

**Test loss: 2.3664824962615967 / Test accuracy: 0.4214000105857849**

313/313 - 1s

**Model 5 Two .2 Dropouts and No Data Augmentation: Analysis**

This model that introduced dropout and no data augmentation had a validation loss of 2.3 and validation accuracy of .42 which significantly beat Model 1. This was an 18 ms/step on average which is a short runtime comparatively. I found that this was a high accuracy especially considering the loss of 2.3. This also had a lower overfitting issue than the next model that has two dropouts  but with only .2.The goal of this process was to reduce overfitting and to improve validation accuracy. Although My model seems to be overfit, I was about to get the validation accuracy to .42 which is higher than my other models. Had I been able to implement SGD with higher epochs, I believe that could have further improved this model.

**Conclusion**

Through this assignment, I hoped to reduce overfitting and underfitting while still improving my validation accuracy through the implementation of a convolutional neural network. Convolutional neural networks take an ample amount of time and manipulation of parameters to assess optimal models, and I found difficulty with improving my model and validation accuracy without overfitting.

Depth, data augmentation, batch size, features, and overall convolutional layers/filters order were important aspects when through the discovery phase of this assignment. Convolutional layer depth depends on the complexity of the data set and problem in question. The importance to the depth mainly has to do with their ability to learn many features at different layers of abstraction. I found that manipulating max pooling helped with overfitting. I continued to have an overfitting problem with my models and think they could have best been improved with increasing max pooling implementation and early stopping. Implementation of a higher dropout to about .6 or .7 could have further improved overfitting while coupled with reducing my epochs to around 9. Another aspect worth considering is poor data augmentation and improving upon this aspect to better generate more training data such as trying reflection for the fill-mode and other such implementations.

Overall, I found that my best model was model 5 without data augmentation and with one .5 dropout. This beat the accuracy of my other models, but I think there is room to improve as previously stated. The figure below contains Comparison of the models to show overfitting and a ratio of loss to the validation accuracy as a risk reward metric. This is with 20 epochs, and again, if I had more time, I would go back and rerun all of these at around 10 epochs.