Starting Evaluations: Input Shape 64x64

def create\_model():

    model = tf.keras.Sequential()

    model.add(tf.keras.layers.RandomFlip(mode = "horizontal\_and\_vertical", seed = 5))

    model.add(tf.keras.layers.RandomRotation(0.2, fill\_mode= "reflect", interpolation="bilinear", seed= 5, fill\_value=0.0))

    model.add(tf.keras.layers.Conv2D(filters =32, kernel\_size=(7,7), padding="same", activation="relu", input\_shape= (64,64,3)))

    model.add(tf.keras.layers.BatchNormalization(axis =1))

    model.add(tf.keras.layers.MaxPooling2D(pool\_size=(2,2)))

    model.add(tf.keras.layers.Dropout(0.8)) #Dropout layers prevent neurons from relying on one input because it might be dropped out at random

    model.add(tf.keras.layers.Conv2D(filters = 32, kernel\_size=(3,3), padding="same", activation = "relu"))

    model.add(tf.keras.layers.MaxPooling2D(pool\_size=(2,2)))

    #model.add(tf.keras.layers.Dropout(0.5)) #Adding a third Dropout Layers dramatically decreases test accuracy

    #model.add(Flatten())

    model.add(tf.keras.layers.Conv2D(filters = 32, kernel\_size = (3,3), padding="same", activation = "relu"))

    model.add(tf.keras.layers.MaxPooling2D(pool\_size=(2,2)))

    model.add(tf.keras.layers.Flatten())

    model.add(tf.keras.layers.Dropout(0.8))

    model.add(tf.keras.layers.Dense(28,activation = "relu"))

    model.add(tf.keras.layers.Dense(4, activation="softmax"))

    model.compile(optimizer = 'adam', loss = 'categorical\_crossentropy', metrics = ['accuracy'])

    #model.summary()

    return model

def train\_model(model, x\_train, y\_train):

    earlyStop = EarlyStopping(monitor = 'loss', patience = 17) #adjust based on numbers of Epoch (lower Epoch requires less patience, and vice versa)

    #batch\_size set to 10 to increase training time leading to better accuracy

    return model.fit(x=x\_train, y=y\_train, epochs = 201, validation\_split = 0.20, batch\_size = 10, callbacks = [earlyStop])

Figure\_1:

Epoch: 201

batch\_size: 10

Dropout (1st, 2nd): 0.8

correct: 39, wrong: 21

accuracy = 0.65

loss = 0.871159017086029

accuracy = 0.6499999761581421

Note: Accuracy at first attempts sits at 0.65, yet the graph does display both the accuracy and the validation line close together indicating minimal to no overfitting of graphs. Since Epoch starts off at 201, the next attempt will have a reduced Epoch quantity to 150. The goal of the next attempt is to determine the optimum epoch to utilize and obtain the best accuracy.

Figure\_2:

Epoch: 150

correct: 50, wrong: 10

accuracy = 0.8333333333333334

loss = 0.6081648468971252

accuracy = 0.8333333134651184

Note: Just by reducing Epoch to 150, the training model was able to obtain an accuracy of 0.833. This indication shows that the lower number of Epoch could provide a better accuracy. The goal of this training model is to be able to achieve above 90%. Therefore, the next attempt will have a reduced Epoch to 140, this is so that we can find whether the direction in which the model can achieve 90% is close to 150 Epoch.

Figure\_3:

Epoch: 140

correct: 48, wrong: 12

accuracy = 0.8

loss = 0.7506850957870483

accuracy = 0.800000011920929

Note: Reducing Epoch to 140 yielded a lower accuracy than previous result. This indicates that to obtain an accuracy over 90%, having a smaller Epoch than 150 may not be the best direction for this model. Therefore, Epoch will now increase to 160 to see if it yields a better result than Figure\_2.

Figure\_4:

Epoch: 160

correct: 52, wrong: 8

accuracy = 0.8666666666666667

2/2 [==============================] - 0s 22ms/step - loss: 0.7970 - accuracy: 0.8667

loss = 0.7970461249351501

accuracy = 0.8666666746139526

Epoch 129/160

Note: Epoch 160 yields a better result than Figure\_2 affirms the theory mentioned in the result of Figure\_3. For the next result, Epoch will increase to 170. If the result continues to increase, then the desired epoch is likely between 200 and 170.

Figure\_5:

Epoch: 170

correct: 49, wrong: 11

accuracy = 0.8166666666666667

2/2 [==============================] - 0s 22ms/step - loss: 0.7558 - accuracy: 0.8167

loss = 0.7557841539382935

accuracy = 0.8166666626930237

Epoch 89/170

Note: It seems that raising the Epoch quantity is not a correct direction above 170 since it has yielded a lower accuracy result. Epoch will be reduced to and between 150 and 160 to continue finding the optimal epoch to train the data.

Figure\_6:

Epoch: 155

correct: 50, wrong: 10

accuracy = 0.8333333333333334

2/2 [==============================] - 0s 27ms/step - loss: 0.7985 - accuracy: 0.8333

loss = 0.7984545826911926

accuracy = 0.8333333134651184

Note: Since going above Epoch 160 has yielded a lower result, the result of this training shows that the higher accuracy Epoch range is likely between 155 and 160. Therefore, multiple runs will be made with each one’s Epoch ranging from 156-159 since the result of 155 and 160 is already present.

Figure\_7:

Epoch: 158

correct: 52, wrong: 8

accuracy = 0.8666666666666667

2/2 [==============================] - 0s 28ms/step - loss: 0.4821 - accuracy: 0.8667

loss = 0.48209577798843384

accuracy = 0.8666666746139526

Epoch 139/158

Note: After testing Epoch range between 156 to 159, the Epoch that yielded the best result is at 158. To push the accuracy further, other values such as Dropout can be adjusted to meet the 90% goal. By default, Dropout value sits at 0.8 for both layers. Therefore, both the Dropout value will reduce to 0.7 to determine the type of changes in the accuracy.

Figure\_8:

Epoch: 158

Dropout (1st,2nd): 0.7

correct: 53, wrong: 7

accuracy = 0.8833333333333333

2/2 [==============================] - 0s 25ms/step - loss: 0.8889 - accuracy: 0.8833

loss = 0.8888686895370483

accuracy = 0.8833333253860474

Epoch 96/158

Note: Reducing the Dropout layers to 0.7 increased the accuracy by roughly 2 percent. However, the graph displays overfitting as the validation and train line deviates from each other, but in order to see if the model can achieve accuracy at least 90% Dropout layers will be reduced to 0.6.

Figure\_9:

Epoch: 158

Dropout (1st): 0.7

Dropout (2nd): 0.6

correct: 55, wrong: 5

accuracy = 0.9166666666666666

2/2 [==============================] - 0s 25ms/step - loss: 1.0054 - accuracy: 0.9167

loss = 1.0053658485412598

accuracy = 0.9166666865348816

Epoch 115/158

Note: Having achieved above 91.6% comes with a drawback of overfitting the graph. The challenge now is to maintain accuracy above 90% while reducing overfitting on the graph. In this case, batch\_size will be increased since it is one of the ways to reduce overfitting other than changing the Dropout layer again. Early Stopping patience will be set lower to avoid over fitting the data when accuracy runs into diminishing return.

Figure\_10:

Epoch: 158

Dropout 1st: 0.7

Dropout 2nd: 0.6

batch\_size: 16

Early stopping: 15

correct: 54, wrong: 6

accuracy = 0.9

2/2 [==============================] - 0s 30ms/step - loss: 0.5254 - accuracy: 0.9000

loss = 0.5253992676734924

accuracy = 0.8999999761581421

Epoch 103/158

Note: The increase in batch\_size as well as the change in Early Stopping have reduced the accuracy to exactly 90%. However, the issues of overfitting has become slightly less as a result.