CNN

During the process of improving the CNN program there was overfitting. The augmentation technique was used to see how this would change the accuracy results of the data.

Augmentation implementation trial:

Tried out the augmentation method to see what change would happen when testing for accuracy.

```
[40] data_augmentation = tf.keras.Sequential(
        [
         layers.RandomFlip("horizontal", input_shape =(32,32,3)),
         layers.RandomRotation(0.1),
         layers.RandomZoom(0.1),
        ]
    )
```

```
model = models.Sequential()
model.add(data_augmentation)
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(32, 32, 3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
```

This is the resulting model summary for the neural network design, only addition is the augmented layer added known as the ‘sequential_16’.

```
Model: "sequential_17"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_16 (Sequential)  (None, 32, 32, 3)         0

 conv2d_27 (Conv2D)          (None, 30, 30, 32)        896

 max_pooling2d_18 (MaxPoolin  (None, 15, 15, 32)       0
 g2D)

 conv2d_28 (Conv2D)          (None, 13, 13, 64)        18496

 max_pooling2d_19 (MaxPoolin  (None, 6, 6, 64)         0
 g2D)

 conv2d_29 (Conv2D)          (None, 4, 4, 64)          36928

 flatten_3 (Flatten)         (None, 1024)              0

 dense_6 (Dense)             (None, 64)                65600

 dense_7 (Dense)             (None, 10)                650

=================================================================
Total params: 122,570
Trainable params: 122,570
Non-trainable params: 0
_________________________________________________________________
```

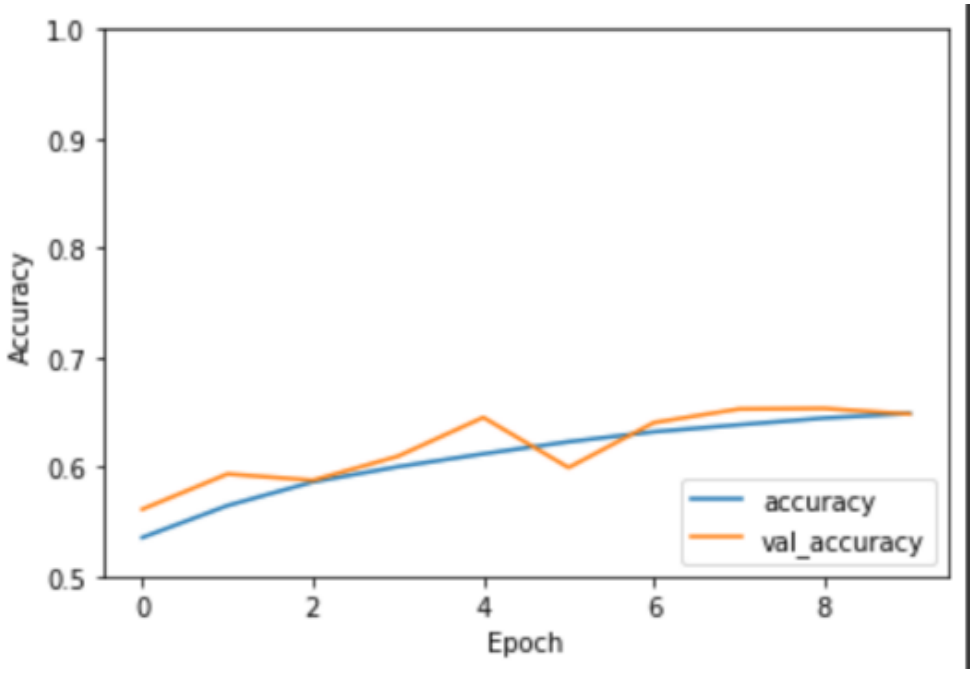
This method was not favorable for increasing the accuracy. As it seems, the accuracy has reduced down from the previous accuracy of around 70%.

```
[46] model.compile(optimizer='adam',
                   loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits=True),
                   metrics=['accuracy'])

     history = model.fit(train_images, train_labels, epochs=10,
                         validation_data=(test_images, test_labels))
```

```
Epoch 1/10
1563/1563 [==============================] - 78s 49ms/step - loss: 1.3055 - accuracy: 0.5354 - val_loss: 1.2493 - val_accuracy: 0.5612
Epoch 2/10
1563/1563 [==============================] - 78s 50ms/step - loss: 1.2299 - accuracy: 0.5646 - val_loss: 1.1647 - val_accuracy: 0.5935
Epoch 3/10
1563/1563 [==============================] - 78s 50ms/step - loss: 1.1737 - accuracy: 0.5862 - val_loss: 1.1581 - val_accuracy: 0.5877
Epoch 4/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.1334 - accuracy: 0.6001 - val_loss: 1.1288 - val_accuracy: 0.6097
Epoch 5/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.1014 - accuracy: 0.6119 - val_loss: 1.0086 - val_accuracy: 0.6452
Epoch 6/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.0716 - accuracy: 0.6229 - val_loss: 1.1530 - val_accuracy: 0.5993
Epoch 7/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.0506 - accuracy: 0.6320 - val_loss: 1.0287 - val_accuracy: 0.6404
Epoch 8/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.0240 - accuracy: 0.6384 - val_loss: 0.9958 - val_accuracy: 0.6529
Epoch 9/10
1563/1563 [==============================] - 77s 49ms/step - loss: 1.0091 - accuracy: 0.6446 - val_loss: 0.9902 - val_accuracy: 0.6534
Epoch 10/10
1563/1563 [==============================] - 77s 49ms/step - loss: 0.9977 - accuracy: 0.6488 - val_loss: 1.0086 - val_accuracy: 0.6483
```

Even though the accuracy did not increase, it has increased the stability of the val_accuracy line to match up with the accuracy.



After some trial and error with different model implementations, I have found a model that I believe that would help achieve the accuracy I need. First was to make an augmentation layer definition with the input shape as 32,32,3 as that represents the picture format for the images.

```
data_augmentation = tf.keras.Sequential(
    [
     layers.RandomFlip("horizontal", input_shape =(32,32,3)),
     layers.RandomRotation(0.1),
     layers.RandomZoom(0.1),
    ]
)
```

Then I included the augmentation layer into the convolution block by ‘model.add’. I changed the Conv2D values to be a bit higher and also added some dense layers in order to increase the total params for training. Added a drop layer of 0.5 to drop 50% data values to help improve accuracy.

```
[72] model = models.Sequential()
     model.add(data_augmentation)
     model.add(layers.Conv2D(192, (3, 3), activation='relu', input_shape=(32, 32, 3)))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dense(32, activation='relu'))
     model.add(layers.Conv2D(384, (3, 3), activation='relu'))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dense(416, activation='relu'))
     model.add(layers.Conv2D(320, (3, 3), activation='relu'))
     model.add(layers.MaxPooling2D((2, 2)))
     model.add(layers.Dropout(0.5))
```

Then I added some more dense layers to further increase the total params for testing. A rescaling layer was added to help normalize the data and the last dense layer of 10 represents the 10 classifications for the images.

```
[74] model.add(layers.Flatten())
     model.add(layers.Dense(736, activation='relu'))
     model.add(layers.Dense(672, activation='relu'))
     model.add(layers.Dense(256, activation='relu'))
     model.add(layers.Rescaling(1./255))
     model.add(layers.Dense(10))
```

After executing each line, this is the model that I was using that is displayed by the model summary.

```
 Layer (type)                Output Shape              Param #
=================================================================
 sequential_23 (Sequential)  (None, 32, 32, 3)         0

 conv2d_108 (Conv2D)         (None, 30, 30, 192)       5376

 max_pooling2d_40 (MaxPoolin  (None, 15, 15, 192)      0
 g2D)

 dense_30 (Dense)            (None, 15, 15, 32)        6176

 conv2d_109 (Conv2D)         (None, 13, 13, 384)       110976

 max_pooling2d_41 (MaxPoolin  (None, 6, 6, 384)        0
 g2D)

 dense_31 (Dense)            (None, 6, 6, 416)         160160

 conv2d_110 (Conv2D)         (None, 4, 4, 320)         1198400

 max_pooling2d_42 (MaxPoolin  (None, 2, 2, 320)        0
 g2D)

 dropout_26 (Dropout)        (None, 2, 2, 320)         0

 flatten_7 (Flatten)         (None, 1280)              0

 dense_32 (Dense)            (None, 736)               942816

 dense_33 (Dense)            (None, 672)               495264

 dense_34 (Dense)            (None, 256)               172288

 rescaling (Rescaling)       (None, 256)               0

 dense_35 (Dense)            (None, 10)                2570

=================================================================
Total params: 3,094,026
Trainable params: 3,094,026
Non-trainable params: 0
```

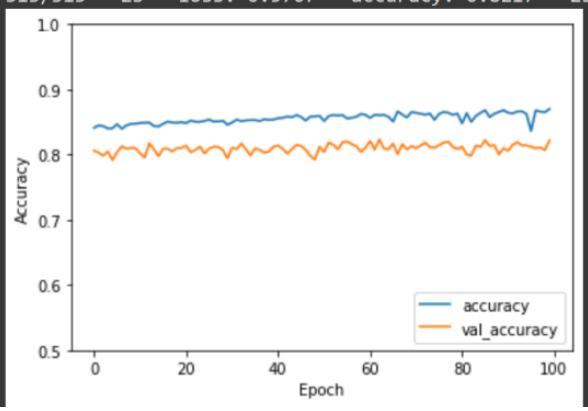
After trying many different methods I found that increasing the params and implementing more dense blocks and dropout layers would help improve the accuracy. I also found that increasing the epoch will allow the accuracy to improve with time, so having a high epoch value is key to improving accuracy.

After running the epoch, I ran this line of code to plot the ‘accuracy’ and ‘val_acuracy’ of the model over the epoch as it improved with time. The accuracy is normalized where 1 is 100% accuracy. The epoch in this graph will say 0 to 100 but this is after running a previous 100 epoch, thus the actual epoch will be 200.

```
plt.plot(history.history['accuracy'], label='accuracy')
plt.plot(history.history['val_accuracy'], label = 'val_accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='lower right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 2s - loss: 0.5707 - accuracy: 0.8217 - 2s/epoch - 5ms/step
```

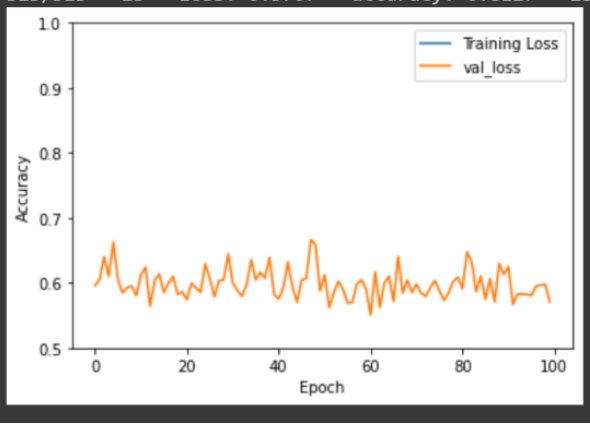


The second line of code was used to plot the ‘Training Loss’ and ‘val_loss’ for accuracy vs epoch.
The training loss gradually went down but the val_loss gradually went down as well but hovered around 0.6. This was taken after 200 epochs were run.

```
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label = 'val_loss')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.ylim([0.5, 1])
plt.legend(loc='upper right')

test_loss, test_acc = model.evaluate(test_images,  test_labels, verbose=2)
```

```
313/313 - 2s - loss: 0.5707 - accuracy: 0.8217 - 2s/epoch - 5ms/step
```



This resulted in the accuracy after running 200 epochs of 82.17% test accuracy.

```
[87] print(test_acc)

0.8216999769210815
```

Balloon Flight

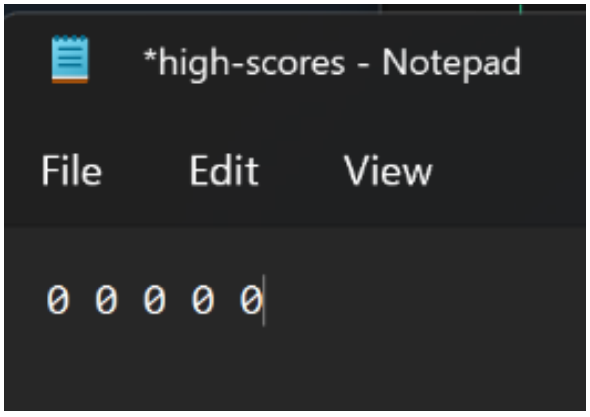
The Balloon Flight game was successfully coded. There are four tweaks that were made to the game after the game was successfully coded.

Tweak1:

The program originally had 3 high score values in the text file separated by spaces. When reading in the file in line 44, it reads line by line and displays the scores in the file, separated by spaces. This is indicated in line 46. When a new high score is higher than the one on file, it is replaced otherwise the score previously there will be shown.

```
40  def update_high_scores():
41      global score, scores
42      filename = r"high-scores.txt"##tweak1 added more high scores
43      scores = []
44      with open(filename, "r") as file:
45          line = file.readline()
46          high_scores = line.split()
47          for high_score in high_scores:
48              if(score > int(high_score)):
49                  scores.append(str(score) + " ")
50                  score = int(high_score)
51              else:
52                  scores.append(str(high_score) + " ")
53      with open(filename, "w") as file:
54          for high_score in scores:
55              file.write(high_score)
56
```

So if the scoreboard needed to increase or decrease, just remove zeros or added zeros separated by space.



Tweak2:

Added lives to the game by making a lives hold the value of 3, this can be increased or decreased. Also made a hit counter to indicate collision.

```
35  scores = []
36  lives = 3##tweak2 added lives to the game
37  hit=0## tweak2 serves as an indicator for collision
```

Added the lives indicator text to screen right under score and changed the color to red to better see the difference between them easily. In this game, the positive y axis goes from top to bottom which is why it is 20 for line 81.

```
80          screen.draw.text("Score: " + str(score), (700, 5), color="black")
81          screen.draw.text("Lives: " + str(lives), (700,20), color="red")##tweak 2
```

I utilized the 'ballon.collidepoint' if statement to indicate when lives were lost. When a collision is detected, the hit value will decrease by -1 every time. When the hit value reaches -1 then a life will be lost. When there is no collision, the hit value will be reset to 0 this will help detect when to take lives away at a consistent rate by collision no matter how fast the objects move.

```
142         if balloon.collidepoint(bird.x, bird.y) or balloon.collidepoint(house.x, house.y) or \
143             balloon.collidepoint(tree.x, tree.y):
144                 hit=hit-1##tweak2 when hit, the counter will tick
145                 if hit==-1:##when tick reaches a certain point, life is loss
146                     lives=lives-1
147                 elif lives==0:##when lives reach 0, game over
148                     game_over = True
149                     update_high_scores()
150         else:#when there is no collision reset hit counter
151             hit=0
```

Tweak3:

I increased the speed of the objects coming coming towards the balloon by increasing the + value number in the objects .x variables. For example, originally 'bird.x -=4' moves the bird that was randomly placed by the randint to the left due to the number being positive which is to the right. This happens at a constant rate as the page refreshes to make the bird move to the left as long as the bird.x value is greater than 0 which it will be until it is off to the left of the screen. This is true for the other objects as well. So to increase the rate in which the objects moved, increase the value of the number.

```
103 def update():
104     global game_over, score, number_of_updates, lives, hit, lvl
105     if not game_over:
106         if not up:
107             balloon.y += 1
108 
109         if bird.x > 0:
110             bird.x -=(lvl*2)+8##tweak3 speed it up/tweak 4 increase difficulty each level
111             if number_of_updates == 9:
112                 flap()
113                 number_of_updates == 0
114             else:
115                 number_of_updates += 1
116 
117         else:
118             bird.x = randint(800, 1600)
119             bird.y = randint(10, 200)
120             score += 1
121             number_of_updates = 0
122             lvlup()
123         
124         if house.right > 0:
125             house.x -=(lvl*2)+4##tweak3 speed up house/tweak 4 increase difficulty each leve
126         else:
127             house.x = randint(800, 1600)
128             score += 1
129             lvlup()
130             
131         if tree.right > 0:
132             tree.x -=(lvl*2)+4##tweak3 speed up tree/tweak 4 increase difficulty each level
133         else:
```

Tweak4:

First make a level up indicator to start off at in this case it is 1.

```
38 lvl=1## tweak4 Level up, Level indicator
```

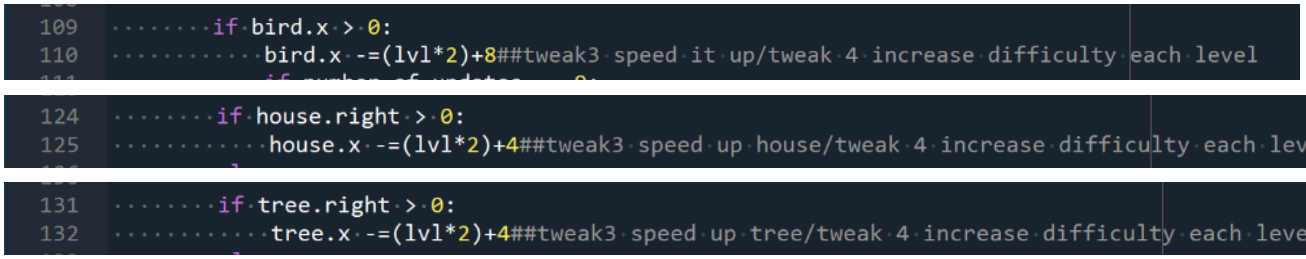
Then define a ‘lvlup’ function to indicate when to increase the level. Make lvl, and score global in order to access their contents in this function. Then made an if statement that if the score is divisible by 10 then the lvl increase by 1. Also if lvl is greater than 10, set the limit by making lvl only equal to 10.

```
66 def lvlup():##tweak 4 define level up check when ever score is increased
67     global lvl,score
68     if score%10==0:##tweak 4 for every 10pts score, level increase by 1
69         lvl +=1
70         if lvl >10:##tweak 4 max level is set to 10
71             lvl=10
72             
```

Added the Level indicated by making a screen.draw command to show the player the current level to the left of the screen on the same level as score. Which is why the position is set to 0,5 on line 82.

```
80         screen.draw.text("Score: " + str(score), (700, 5), color="black")
81         screen.draw.text("Lives: " + str(lives), (700,20), color="red")##tweak 2
82         screen.draw.text("Level: " + str(lvl), (0, 5), color="black")##tweak 4
```

Then similar to tweak#3, I increased the speed every time the level increases by adding the level multiplied by 2 to the default increased speed from tweak#3



```
109         if bird.x > 0:
110             bird.x -=(lvl*2)+8##tweak3 speed it up/tweak 4 increase difficulty each level
```



```
124         if house.right > 0:
125             house.x -=(lvl*2)+4##tweak3 speed up house/tweak 4 increase difficulty each leve
```



```
131         if tree.right > 0:
132             tree.x -=(lvl*2)+4##tweak3 speed up tree/tweak 4 increase difficulty each level
```

The rate in which the speed is increased for the tree based on the lvl can be adjusted to be more impactful or less impactful by the number multiplied.