Winnie Leung
Professor Pham
EE104 Sec 01
April 30, 2022

LAB 8 DOCUMENTATION

Abstract

****

This lab allows users to utilize convolutional neural networks to recognize color images and try different methods to improve accuracy.

Objective

****

Fulfill all requirements listed in the table

Program or Requirement	Use Case	Earned Score / Max Score
Demonstration Video	You must submit a demonstration video or your score for this lab will be zero	
README file & Developer Documentation	README is a brief user guide and developer documentation so that the user can install the proper python packages and knows how to execute your program. The documentation section can contain sample screenshots with explanation.	_____ / 10
Check in your README file, documentation, and codes to Github	You will put all your work on your GitHub account and submit the same code to Canvas and share access to your GitHub with your instructor for grading purpose.	_____ / 5
CNN - Baseline + Increasing Dropout + Data Augmentation + Batch Normalization + Your own method	To achieve > 90% accuracy and recognize successfully 90% of the given test images. Video recording note: Because it will take a long time to run this test, you must submit a recording showing the result. The recording can be done by Zoom or any screen video capturing software. *Trim out the irrelevant contents and only submit the last few minutes showing you achieved your highest accuracy.* Achieve > 90% accuracy: Earn full 30 points Achieve > 87% accuracy: Earn 25 points Achieve > 84% accuracy: Earn 20 points Achieve > 80% accuracy: Earn 20 points Achieve > 77% accuracy: Earn 15 points Achieve > 74% accuracy: Earn 10 points Achieve > 70% accuracy: Earn 5 points Else: 0 point	_____ / 30
CNN - Challenge test	Recognize 5 unrecognizable images from the TEST IMAGES section below (3 points each) Hint: you can leverage and modify the test code from this GoogleColab page to test this lab: https://colab.research.google.com/github/tensorflow/docs/blob/master/site/en/tutorials/images/classification.ipynb#scrollTo=dC40sRITBSsQ	_____ / 15
Game Development – Balloon Flight	Leverage the base code from chapter Balloon Flight and add your own Hacks and Tweaks for any 4 of the options below: More High Scores, Lives, Speed It Up, Different Way to Score, File Handling, Add in Multiples of Each Obstacles, Level Up, Space Out the Obstacles	_____ / 40
	TOTAL	**100%**

Requirements

****

Google collab:

```
!pip install tensorflow
!pip install keras
!pip install h5py
!pip install Matplotlib
!pip install numpy
```

```
import tensorflow as tf
# baseline model with dropout and data augmentation on the cifar10 dataset
import sys
import numpy as np
from keras.datasets import cifar10
from tensorflow.keras.utils import to_categorical
from keras.models import Sequential
from keras.layers import Conv2D
from keras.layers import MaxPooling2D
from keras.layers import Dense
from keras.layers import Flatten
from tensorflow.keras.optimizers import SGD
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dropout
from keras.layers import BatchNormalization
from tensorflow.keras import datasets, layers, models
import matplotlib.pyplot as plt
```

Pygame:

```
import pgzrun
import pygame
import pgzero
import random
from pgzero.builtins import Actor
from random import randint
```

Instruction

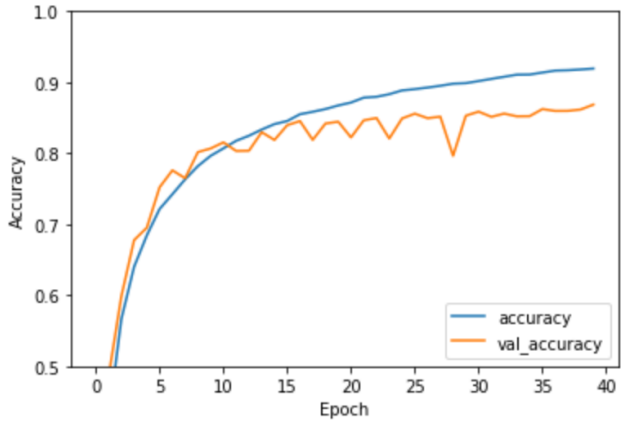
****

- Ensure all packages are installed prior to running an programs
- CNN
    - Add optimization techniques learned from 6_NeuralNetwork_Optimization.ppt and https://machinelearningmastery.com/how-to-develop-a-cnn-from-scratch-for-cifar-10-photo-classification/
    - Make modifications to the convolutional base and change the epoch value to achieve a high accuracy value

```
model = models.Sequential()
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same', input_shape=(32, 32, 3)))
model.add(BatchNormalization())
model.add(Conv2D(32, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.2))
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(64, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.3))
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(128, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.4))
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(Conv2D(256, (3, 3), activation='relu', kernel_initializer='he_uniform', padding='same'))
model.add(BatchNormalization())
model.add(MaxPooling2D((2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(256, activation='relu', kernel_initializer='he_uniform'))
model.add(BatchNormalization())
model.add(Dropout(0.5))
model.add(Dense(10, activation='softmax'))
```

```
Epoch 29/40
1563/1563 [==============================] - 40s 26ms/step - loss: 0.3047 - accuracy: 0.8980 - val_loss: 0.6744 - val_accuracy: 0.7964
Epoch 30/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2994 - accuracy: 0.8987 - val_loss: 0.4800 - val_accuracy: 0.8529
Epoch 31/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2916 - accuracy: 0.9017 - val_loss: 0.4468 - val_accuracy: 0.8586
Epoch 32/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2827 - accuracy: 0.9047 - val_loss: 0.4853 - val_accuracy: 0.8513
Epoch 33/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2737 - accuracy: 0.9077 - val_loss: 0.4585 - val_accuracy: 0.8558
Epoch 34/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2692 - accuracy: 0.9107 - val_loss: 0.4960 - val_accuracy: 0.8520
Epoch 35/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2646 - accuracy: 0.9109 - val_loss: 0.4957 - val_accuracy: 0.8522
Epoch 36/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2565 - accuracy: 0.9137 - val_loss: 0.4438 - val_accuracy: 0.8622
Epoch 37/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2509 - accuracy: 0.9164 - val_loss: 0.4795 - val_accuracy: 0.8596
Epoch 38/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2483 - accuracy: 0.9169 - val_loss: 0.4601 - val_accuracy: 0.8597
Epoch 39/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2431 - accuracy: 0.9180 - val_loss: 0.4633 - val_accuracy: 0.8616
Epoch 40/40
1563/1563 [==============================] - 39s 25ms/step - loss: 0.2392 - accuracy: 0.9193 - val_loss: 0.4550 - val_accuracy: 0.8685
```

```
313/313 - 2s - loss: 0.4550 - accuracy: 0.8685 - 2s/epoch - 7ms/step
```



- CNN Challenge
  - Jaguar car

```python
image_url = "https://images.all-free-download.com/images/graphiclarge/classic_jaguar_210354.jpg"
image_path = tf.keras.utils.get_file('classic_jaguar_210354.jpg', origin=image_url)

img = tf.keras.utils.load_img(
    image_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
print(score)
```

```
This image most likely belongs to automobile with a 99.61 percent confidence.
tf.Tensor(
[1.0455601e-04 9.9605584e-01 2.9018420e-06 3.7188004e-06 2.1646790e-06
 2.8373606e-06 2.3319881e-05 3.9860945e-05 1.3030920e-04 3.6344097e-03], shape=(10,), dtype=float32)
```

  - Devel-motors sixteen

```
image_url = "https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/devel-motors-sixteen-1540564064.jpg"
image_path = tf.keras.utils.get_file('devel-motors-sixteen-1540564064.jpg', origin=image_url)

img = tf.keras.utils.load_img(
    image_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
print(score)
```

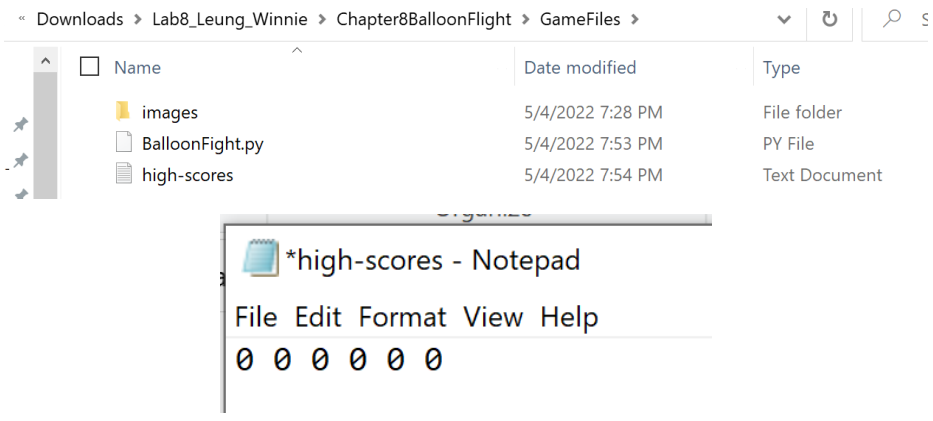
```
Downloading data from https://hips.hearstapps.com/hmg-prod.s3.amazonaws.com/images/devel-motors-sixteen-1540564064.jpg
1335296/1333618 [==============================] - 0s 0us/step
1343488/1333618 [==============================] - 0s 0us/step
This image most likely belongs to automobile with a 99.90 percent confidence.
tf.Tensor(
[6.44805550e-05 9.98974562e-01 2.19593244e-06 2.68521694e-06
 1.84617898e-06 2.09927771e-06 1.92146726e-05 2.66038642e-05
 1.03100574e-04 8.03270203e-04], shape=(10,), dtype=float32)
```

- Valkyrie-spider

```
image_url = "https://amsc-prod-cd.azureedge.net/-/media/aston-martin/images/default-source/models/valkyrie/new/valkyrie-spider_f02-169v2.jpg?mw=1980&rev=-1&hash=
image_path = tf.keras.utils.get_file('valkyrie-spider_f02-169v2.jpg', origin=image_url)

img = tf.keras.utils.load_img(
    image_path, target_size=(32, 32)
)
img_array = tf.keras.utils.img_to_array(img)
img_array = tf.expand_dims(img_array, 0) # Create a batch

predictions = model.predict(img_array)
score = tf.nn.softmax(predictions[0])

print(
    "This image most likely belongs to {} with a {:.2f} percent confidence."
    .format(class_names[np.argmax(score)], 100 * np.max(score))
)
print(score)
```

```
Downloading data from https://amsc-prod-cd.azureedge.net/-/media/aston-martin/images/default-source/models/valkyrie/new/valkyrie-spider_f02-169v2.jpg?mw=1980&rev
729088/727309 [==============================] - 0s 0us/step
737280/727309 [==============================] - 0s 0us/step
This image most likely belongs to automobile with a 99.90 percent confidence.
tf.Tensor(
[6.4726781e-05 9.9897408e-01 2.2255635e-06 2.7941962e-06 1.8732994e-06
 2.2226513e-06 1.9458708e-05 2.7087044e-05 1.0330604e-04 8.0223172e-04], shape=(10,), dtype=float32)
```

- Pygame- Balloon Fight
  - Download the program BaloonFight.py run and observe the code
  - Add high score list
    - Create a high-scores.txt file and type in 0 0 0 0 0 0 with spaces in between each zero. The number of zeros indicates the number of high score values saved, in this case there is 6



    - Add the directory in the code, update the .txt file, and display the high scores at the end of the game

```python
def update_high_scores():
    global score, scores
    filename = r"C:/Users/winni/Downloads/Lab8_Leung_Winnie/Chapter8BalloonFlight/GameFiles/high-scores.txt"
    scores = []
    with open(filename, "r") as file:
        line = file.readline()
        high_scores = line.split()
        for high_score in high_scores:
            if(score > int(high_score)):
                scores.append(str(score) + " ")
                score = int(high_score)
            else:
                scores.append(str(high_score) + " ")
    with open(filename, "w") as file:
        for high_score in scores:
            file.write(high_score)

def display_high_scores():
    screen.draw.text("HIGH SCORES", (350, 150), color="black")
    y = 175
    position = 1
    for high_score in scores:
        screen.draw.text(str(position) + ". " + high_score, (350, y), color="black")
        y += 25
        position += 1
```

- Speed it up
- Add additional obstacles
    - Create copies of the given obstacles

    - Draw the actors and its initial position

```python
bird = Actor("bird-up")
bird.pos = randint(800, 1600), randint(10, 150)

bird2 = Actor("bird2-up")
bird2.pos = randint(800, 1600), randint(5, 100)

house = Actor("house")
house.pos = randint(810, 1600), 460

house2 = Actor("house2")
house2.pos = randint(860, 1600), 460

tree = Actor("tree")
tree.pos = randint(800, 1600), 450

tree2 = Actor("tree2")
tree2.pos = randint(850, 1600), 450
```

    - Animate the bird image to flap its wings, create global bird2_up

```
38
39  bird_up = True
40  bird2_up = True
```

```
97  def flap():
98      global bird_up
99      global bird2_up
100     if bird_up:
101         bird.image = "bird-down"
102         bird_up = False
103     else:
104         bird.image = "bird-up"
105         bird_up = True
106     if bird2_up:
107         bird2.image = "bird2-down"
108         bird2_up = False
109     else:
110         bird2.image = "bird2-up"
111         bird2_up= True
```

- Make the obstacles move towards the left

```
113 def update():
114     global game_over, score, number_of_updates
115     if not game_over:
116         if not up:
117             balloon.y += 2 #speed up the float
118 
119         if bird.x > 0:
120             bird.x -= 5 #speed it up bird
121             if number_of_updates == 9:
122                 flap()
123                 number_of_updates = 0
124             else:
125                 number_of_updates += 1
126         else:
127             bird.x = randint(800, 1600)
128             bird.y = randint(10, 150)
129             score += 1
130             number_of_updates = 0
131 
132         if bird2.x > 0:
133             bird2.x -= 5 #speed it up bird
134             if number_of_updates == 9:
135                 flap()
136                 number_of_updates = 0
137             else:
138                 number_of_updates += 1
139         else:
140             bird2.x = randint(800, 1600)
141             bird2.y = randint(5, 100)
142             score += 1
143             number_of_updates = 0
144 
145         if house.right > 0:
146             house.x -= 3 #speed it up movement to left
147         else:
148             house.x = randint(800, 1600)
149             score += 1
150
```

```
151         if house2.right > 0:
152             house2.x -= 3 #speed it up movement to left
153         else:
154             house2.x = randint(1050, 1600)
155             score += 1
156 
157         if tree.right > 0:
158             tree.x -= 3 #speed it up movement to left
159         else:
160             tree.x = randint(950, 1600)
161             score += 1
162 
163         if tree2.right > 0:
164             tree2.x -= 3 #speed it up movement to left
165         else:
166             tree2.x = randint(1000, 1600)
167             score += 1
168 
```

- Extra lives
    - Create a global lives

```
38 
39  bird_up = True
40  bird2_up = True
41  up = False
42  game_over = False
43  score = 0
44  number_of_updates = 0
45  lives=3  # 3 lives for the game
46 
```

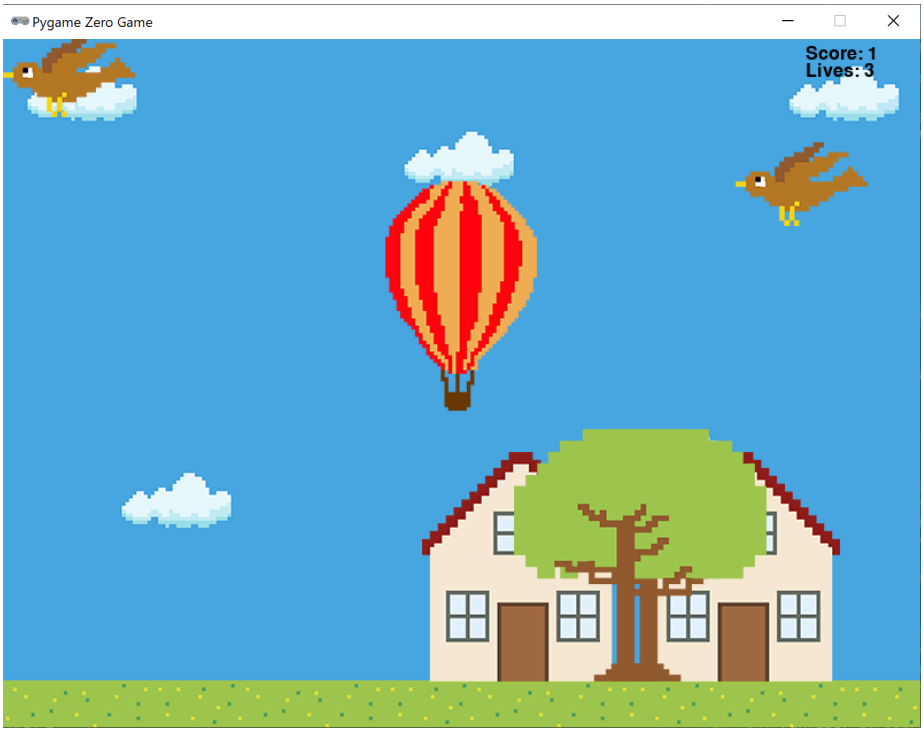
    - Add lives score count at the top right corner

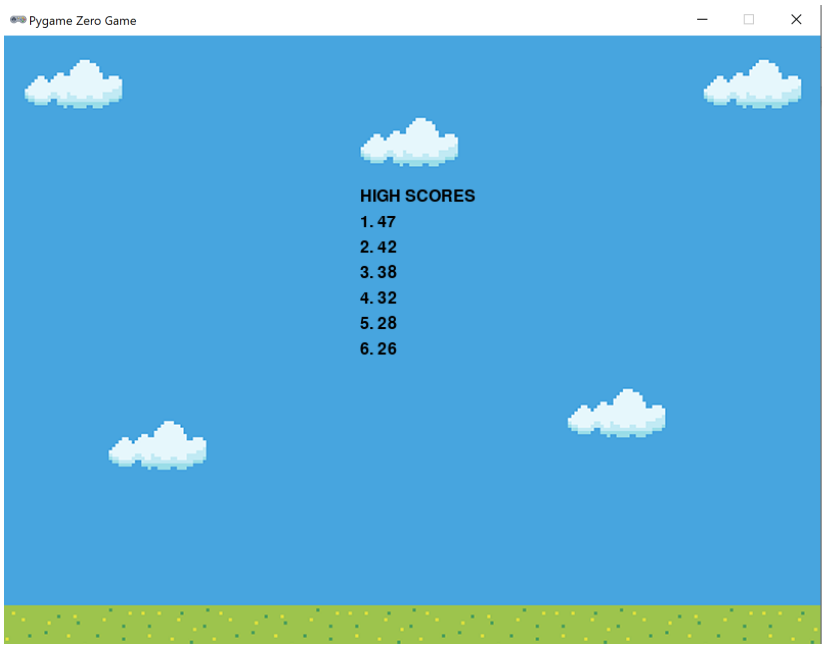
```
86          screen.draw.text("Lives: " + str(lives), (700, 20), color="black")
```

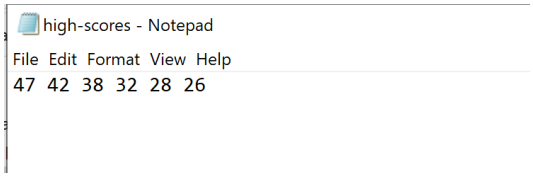
    - If balloon touches obstacle minus one life and reset all obstacles to the right, if no lives left game over

```
175         if balloon.collidepoint(bird2.x, bird2.y) or\
176             balloon.collidepoint(bird.x, bird.y) or \
177             balloon.collidepoint(house.x, house.y) or \
178             balloon.collidepoint(tree.x, tree.y):
179                 if (lives>1):  #if collision happend and lives is greater than 1 position restarts
180                     bird.x = randint(800, 1600)
181                     bird.y = randint(10, 150)
182                     bird2.x = randint(800, 1600)
183                     bird2.y = randint(5, 100)
184                     house.x = randint(800, 1600)
185                     house2.x = randint(1050, 1600)
186                     tree.x = randint(950, 1600)
187                     tree2.x = randint(1000, 1600)
188                     lives-=1 # live score decreases by 1
189                 else: #no more lives left, game over
190                     game_over = True
191                     update_high_scores()
192 
```

- Results should look like the following:







.txt file updates to display the scores in the game