# Bird Classification and Location

電子碩一 顏郁苓 110368155

SPIE Lab

Signal Processing & Intelligent Electronics Lab

A Joint Research Group

# CONTENTS



Signal Processing & Intelligent Electronics Lab

# 01

# Brief view

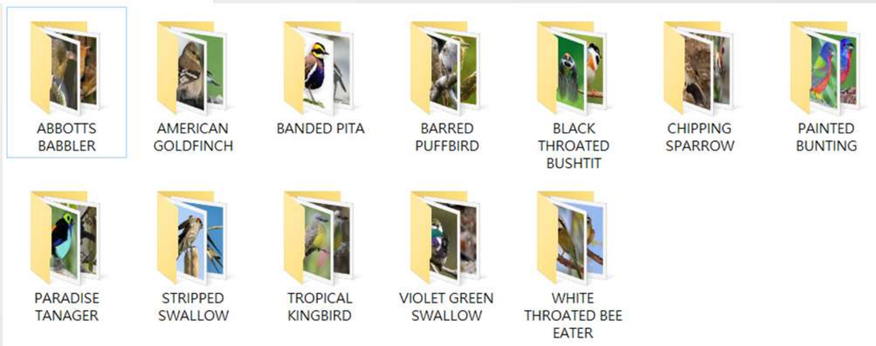
Signal
Processing
&
Intelligent
Electronics
Lab

# About this project

- → Idea comes from:
  - Location: Yolo
  - Classification: Homework2
- → Both classification and location

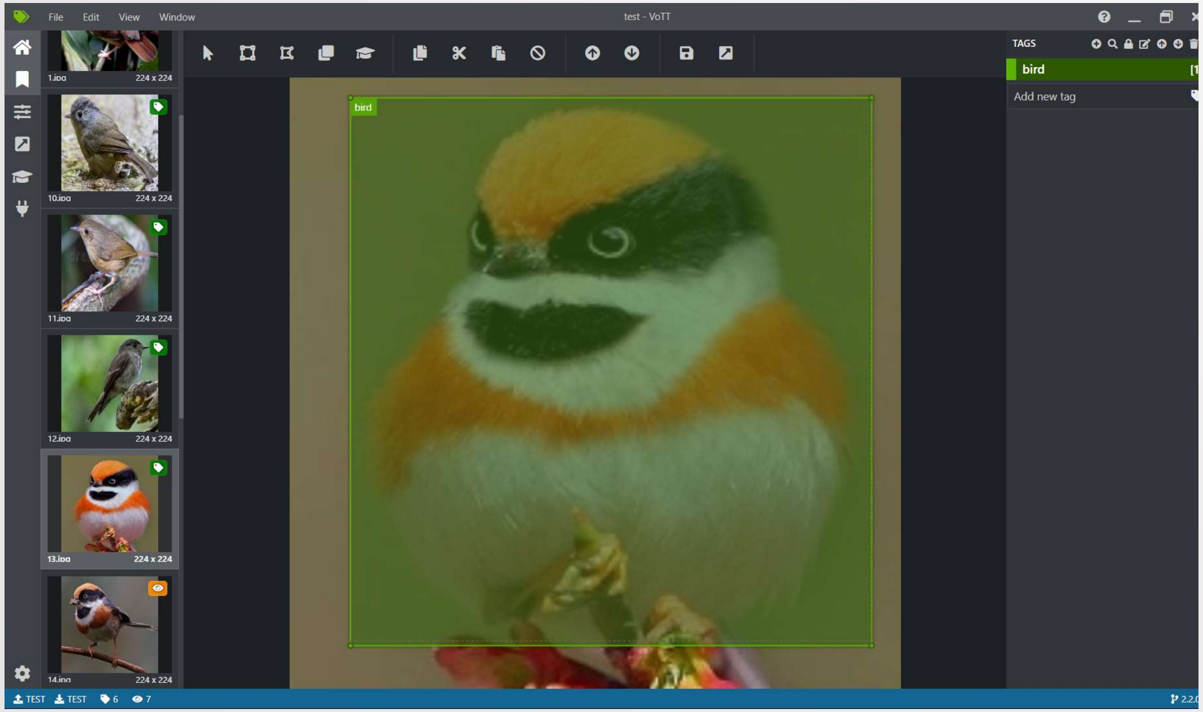
# Dataset

→ Dataset of bird classification on Kaggle
→ 12 species out of 400 species in the dataset as classification
→ Use VoTT to mark the location of birds
→All images are 224 X 224 X 3 color images in jpg format

# Software that are used in this project

→ Coding : VSCODE and GOOGLE COLLAB
→ Labeling: VOTT



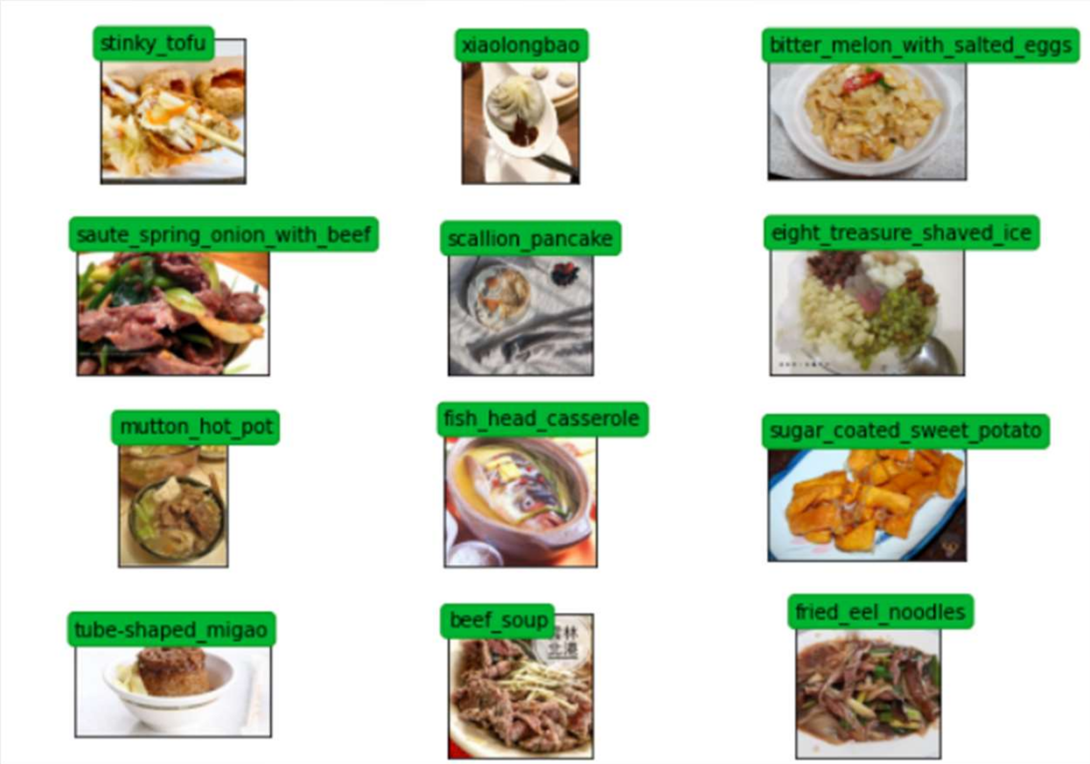

| image | xmin | ymin | xmax | ymax | label |
|---|---|---|---|---|---|
| ABBOTT | 2.665239 | 2.994652 | 222.8021 | 213.2193 | BIRD |
| ABBOTT | 13.92513 | 6.228878 | 221.6043 | 210.8235 | BIRD |
| ABBOTT | 38.33155 | 5.031017 | 219.7177 | 191.3583 | BIRD |
| ABBOTT | 28.4492 | 10.78075 | 214.9562 | 214.1176 | BIRD |
| ABBOTT | 18.02781 | 25.8738 | 215.615 | 195.5508 | BIRD |
| ABBOTT | 5.989305 | 28.50909 | 223.8203 | 202.139 | BIRD |
| ABBOTT | 14.43423 | 27.79038 | 209.3262 | 201.8396 | BIRD |
| ABBOTT | 4.791444 | 29.70695 | 215.4353 | 203.3369 | BIRD |
| ABBOTT | 12.99679 | 10.3016 | 220.4064 | 196.4492 | BIRD |
| ABBOTT | 44.14118 | 17.00963 | 211.7219 | 200.3422 | BIRD |
| ABBOTT | 17.66845 | 2.156151 | 218.7893 | 220.4064 | BIRD |
| ABBOTT | 44.32086 | 17.36898 | 197.6471 | 198.246 | BIRD |
| ABBOTT | 15.57219 | 15.21284 | 219.8075 | 152.1283 | BIRD |
| ABBOTT | 17.66845 | 22.04064 | 209.446 | 197.0481 | BIRD |
| ABBOTT | 15.27273 | 3.833156 | 209.2064 | 166.5027 | BIRD |
| ABBOTT | 4.851341 | 42.64385 | 211.6021 | 185.6684 | BIRD |
| ABBOTT | 4.192513 | 17.48877 | 222.6225 | 176.984 | BIRD |

# 02

# Knowledge used

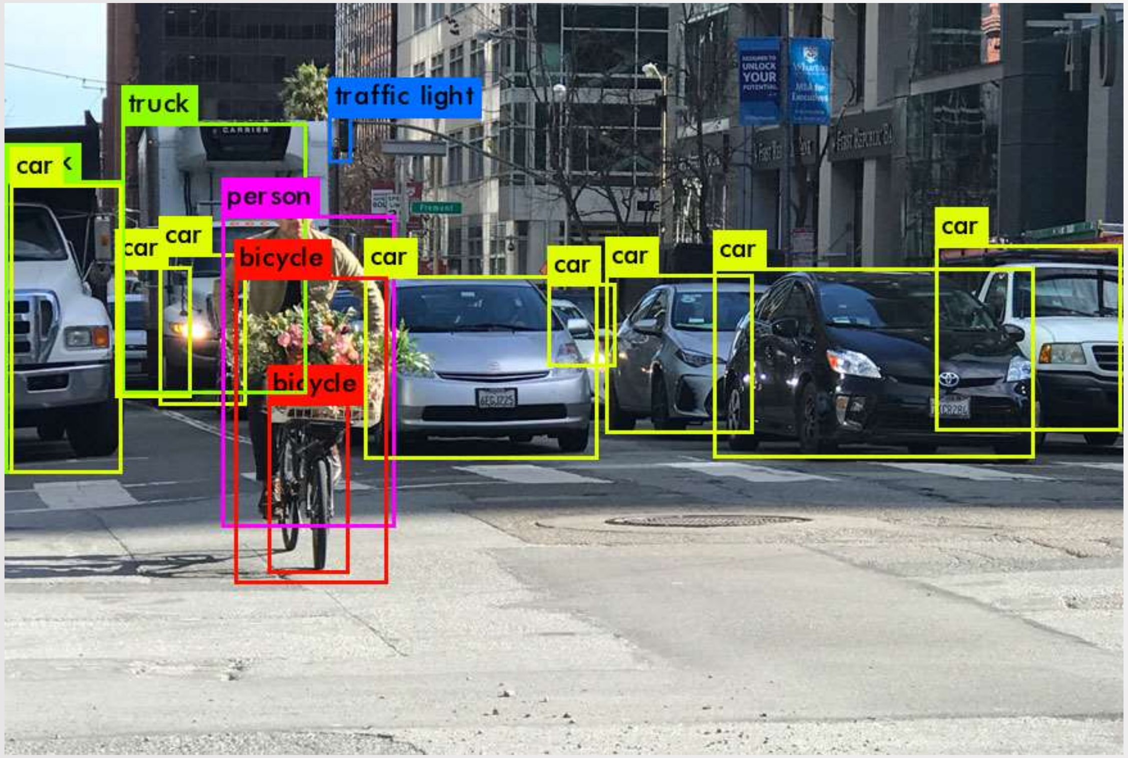
Signal
Processing
&
Intelligent
Electronics
Lab

# Classification




- → Homework2:food classification
- → Use Keras API in Homework2 (keras.applications.inception_v3.InceptionV3)
- → Use another architecture in this program

# Location




→ Often use Yolo API in object location

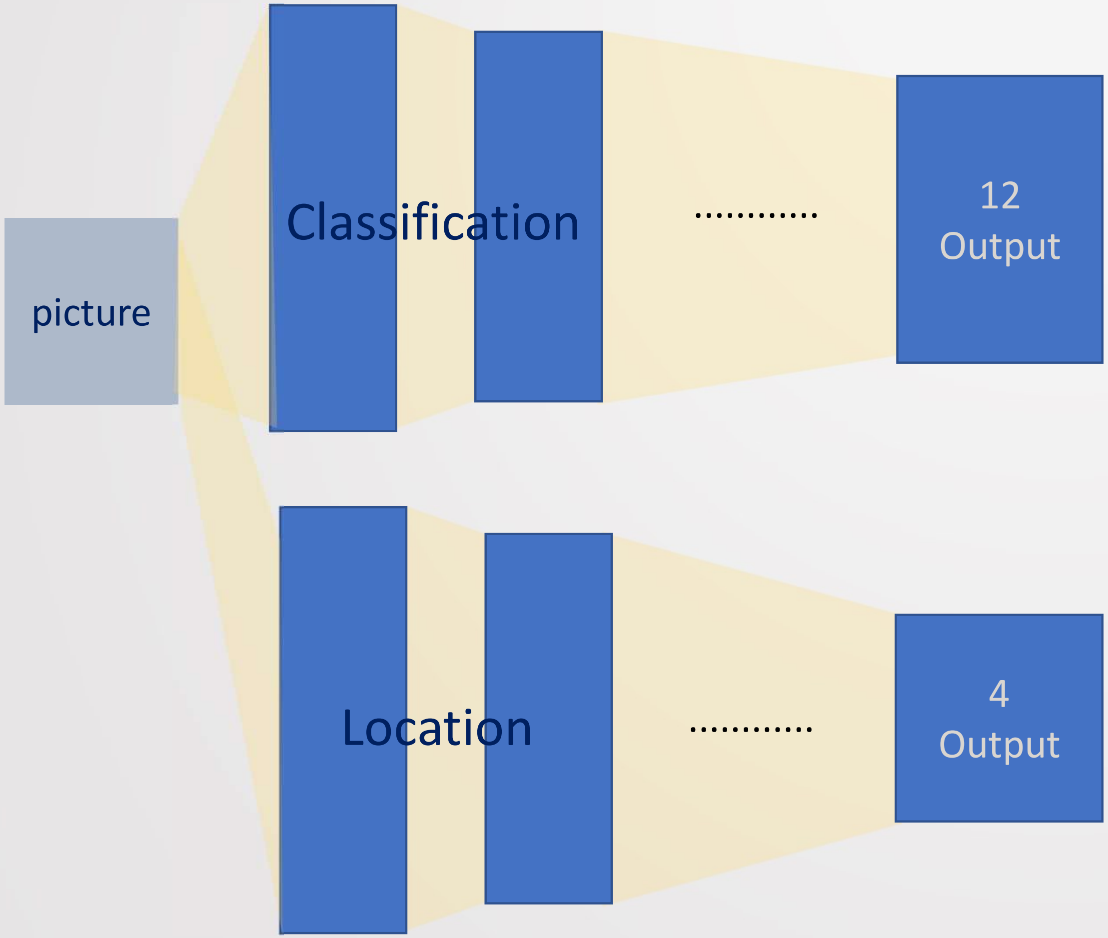
→ Use another architecture in this program

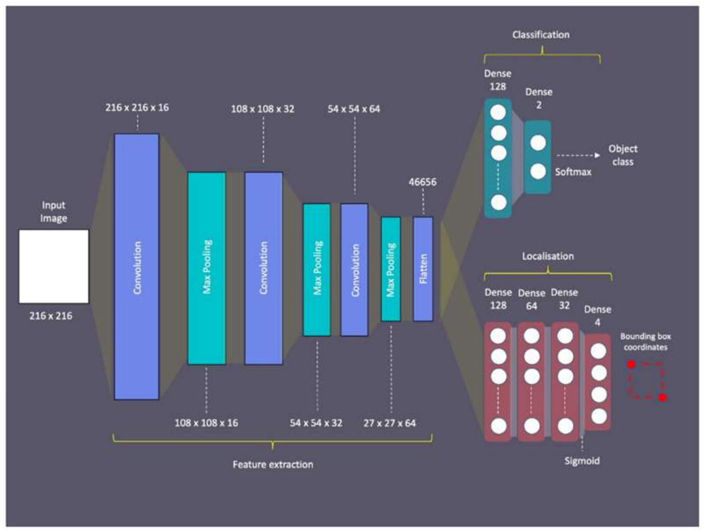
# 03

# Architecture

Signal
Processing
&
Intelligent
Electronics
Lab

# Architecture

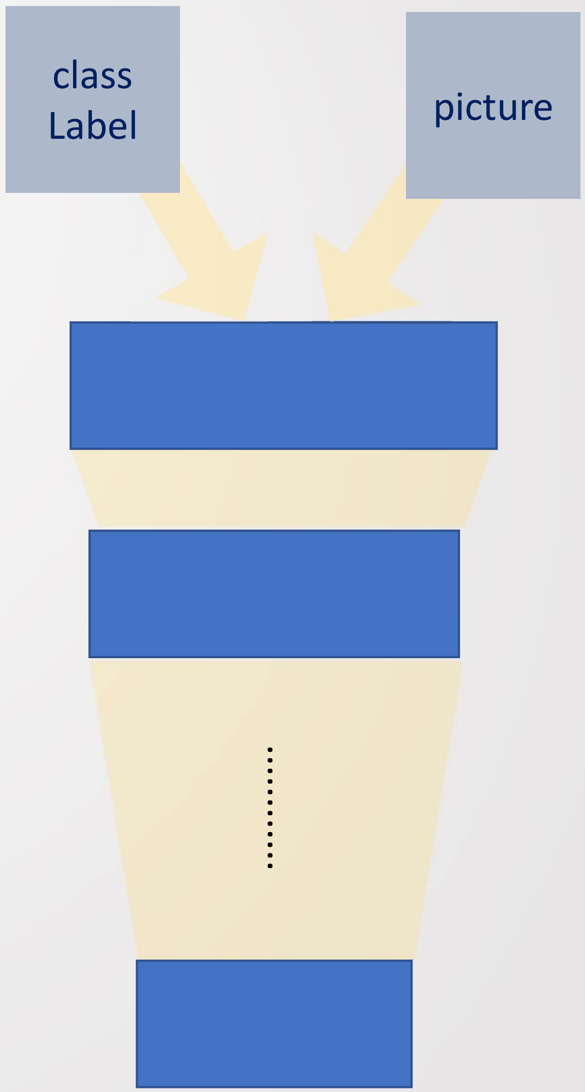


Architecture of the dual-output neural network

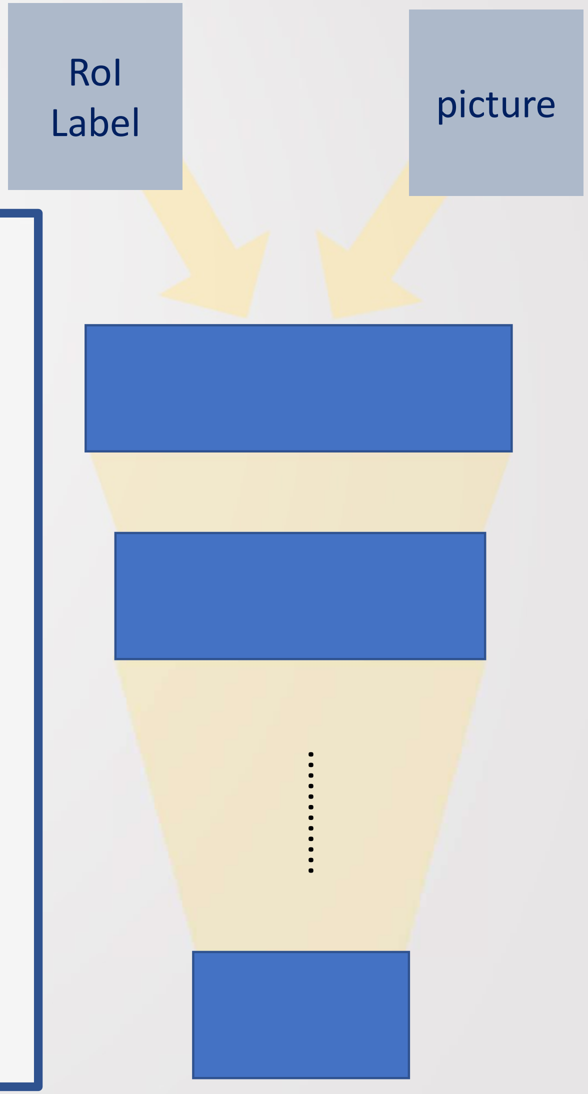
# Classification

```
##  MODEL  ##

#create the common input layer
input_shape = (height, width, 3)
input_layer = tf.keras.layers.Input(input_shape)

#create the base layers
base_layers = layers.Rescaling(1./255, name='bl_1')(input_layer)
base_layers = layers.Conv2D(64, (3,3), padding='same', activation='relu', name='bl_2')(base_layers)
base_layers = layers.BatchNormalization()(base_layers)
base_layers = layers.MaxPooling2D(name='bl_3')(base_layers)
base_layers = layers.Conv2D(64, (3, 3), activation='relu', name='bl_4')(base_layers)
base_layers = layers.MaxPooling2D(pool_size=(2, 2),name='bl_5')(base_layers)
base_layers = layers.BatchNormalization()(base_layers)
base_layers = layers.Conv2D(64, (3, 3), padding='same',activation='relu', name='bl_6')(base_layers)
base_layers = layers.MaxPooling2D(name='bl_7')(base_layers)
base_layers = layers.Conv2D(64, (3, 3),  activation='relu', name='bl_8')(base_layers)
base_layers = layers.MaxPooling2D(pool_size=(2, 2),name='bl_9')(base_layers)
base_layers = layers.BatchNormalization()(base_layers)
base_layers = layers.Dropout(0.35)(base_layers)
base_layers = layers.Conv2D(64, (3,3), padding='same', activation='relu', name='bl_10')(base_layers)
base_layers = layers.MaxPooling2D(name='bl_11')(base_layers)
#base_layers = layers.Flatten(name='bl_12')(base_layers)

#create the classifier branch
#classifier_branch = base_model.output
classifier_branch = layers.Flatten()(base_layers)
classifier_branch = layers.Dropout(0.5)(classifier_branch)
classifier_branch = layers.Dense(512, activation='relu',name='cl_1')(classifier_branch)
classifier_branch = layers.BatchNormalization()(classifier_branch)
prediction_ans = layers.Dense(Category_count, activation='softmax',name='cl_head')(classifier_branch)


model_class = tf.keras.Model(input_layer, outputs=[prediction_ans])


print(model_class.summary())
print("prediction_ans=",prediction_ans)
```



# Location

```
##model##
input_shape = (height, width, 3)
input_layer = tf.keras.layers.Input(input_shape)

model_located = layers.Rescaling(1./255, name='ml_1')(input_layer)
model_located = layers.Conv2D(16, 3, padding='same', activation='relu', name='ml_2')(model_located)
model_located = layers.MaxPooling2D(name='ml_3')(model_located)
model_located = layers.Conv2D(32, 3, padding='same', activation='relu', name='ml_4')(model_located)
model_located = layers.MaxPooling2D(name='ml_5')(model_located)
model_located = layers.Conv2D(64, 3, padding='same', activation='relu', name='ml_6')(model_located)
model_located = layers.MaxPooling2D(name='ml_7')(model_located)
model_located = layers.Conv2D(128, 3, padding='same', activation='relu', name='ml_8')(model_located)
model_located = layers.MaxPooling2D(name='ml_9')(model_located)
model_located = layers.Conv2D(256, 3, padding='same', activation='relu', name='ml_10')(model_located)
model_located = layers.MaxPooling2D(name='ml_11')(model_located)
model_located = layers.Flatten(name='ml_12')(model_located)
model_located = layers.Dense(128, activation='relu', name='ml_13')(model_located)
model_located = layers.Dense(64, activation='relu', name='ml_14')(model_located)
model_located = layers.Dense(32, activation='relu', name='ml_15')(model_located)
locator_ans = layers.Dense(4, activation='sigmoid', name='ml_head')(model_located)

model_located = tf.keras.Model(input_layer,outputs=[locator_ans])


print(model_located.summary())
print("locator_ans=",locator_ans)
```



# 04

# Code and Experimental Results

# Code

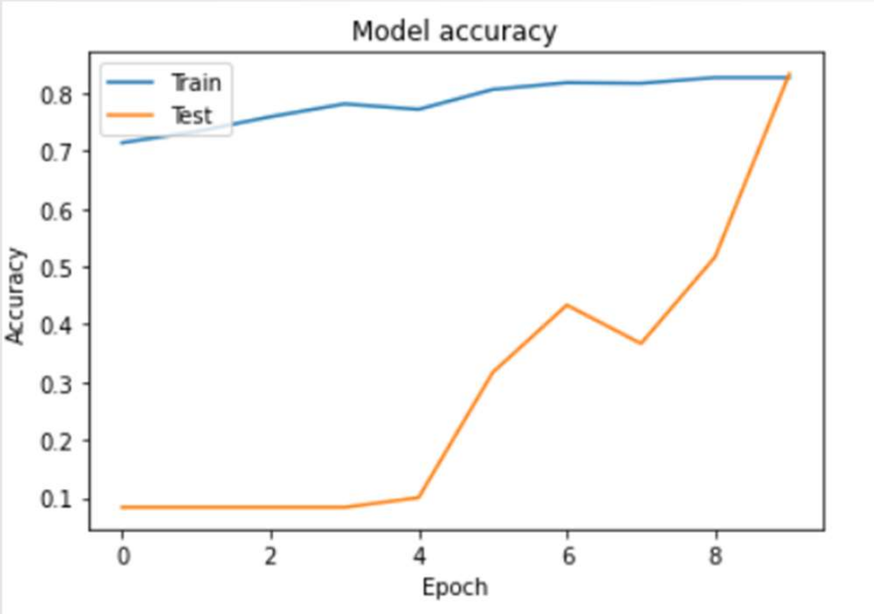
```python
In [1]:
import matplotlib.pyplot as plt
import numpy as np
import os
import PIL
import tensorflow as tf
from tensorflow import keras
import pandas as pd
from PIL import Image
from PIL.ImageDraw import Draw
import matplotlib.pyplot as plt
import cv2
import pandas as pd
import random

In [13]:
model_class = tf.keras.models.load_model('bird_test_class.h5')
model_located = tf.keras.models.load_model('bird_test_located.h5')
cl_train_dir = 'D:\program\Images\cl_data_train'
cl_valid_dir = "D:\program\Images\cl_data_valid"

test_dir = 'D:\program\Images\\test'

test_dict = {}
for root, dirs, files in os.walk("D:\program\Images\\test"):
    for filename in files:
        test_id, file_ext = os.path.splitext(filename)
        test_dict[test_id] = filename

In [14]:
def load_img(filename, target_w=224, target_h=224):
   np_image = Image.open(filename)
   np_image = np.array(np_image).astype('float32')/255
   np_image = np.reshape(np_image,(224,224,3))
   np_image = np.expand_dims(np_image, axis=0)

   return np_image

In [15]:
CATEGORIES = os.listdir(cl_train_dir)
print(str(len(CATEGORIES)),'CATEGORIES are ', CATEGORIES)
Category_count = len(CATEGORIES)

#pic_list = TEST資料夾中的所有檔案名稱
pic_list = os.listdir(test_dir)

12 CATEGORIES are  ['ABBOTTS BABBLER', 'AMERICAN GOLDFINCH', 'BANDED PITA', 'BARRED PUFFBIRD', 'BLACK THROATED BUSHTIT', 'CHIPPING SPARROW', 'PAINTED BUNTING', 'PARADISE TANAGER', 'STRIPPED SWALLOW', 'TROPICAL KINGBIRD', 'VIOLET GREEN SWALLOW', 'WHITE THROATED BEE EATER']

In [19]:
# Read images in order and make predictions
results = []
CATE_result =[]
s_point =[]
e_point =[]
plt.figure(figsize=(224,224))

#測試test資料夾裡面所有資料
for i in range(len(test_dict)):
    print(test_dict[str(i)])
    img = load_img("D:\program\Images\\test\\" + test_dict[str(i)])
    ret1 = model_class.predict(img)
    ret2 = model_located.predict(img)
    A = ret1[0]
    B = ret2[0]
    print("A=")
    print(A)
    print("B=")
    print(B)
    maxiem =np.argmax(A)#最大值的編號->從CATEGORIES對應標籤
    results.append(maxiem)
    CATE_result.append(CATEGORIES[int(maxiem)])
    start_point = (int(B[0]*224),int(B[1]*224))
    end_point   = (int(B[2]*224),int(B[3]*224))
    s_point.append(start_point)
    e_point.append(end_point)

print(results)
print(CATE_result)

0.jpg
1/1 [==============================] - 0s 46ms/step
1/1 [==============================] - 0s 29ms/step
```

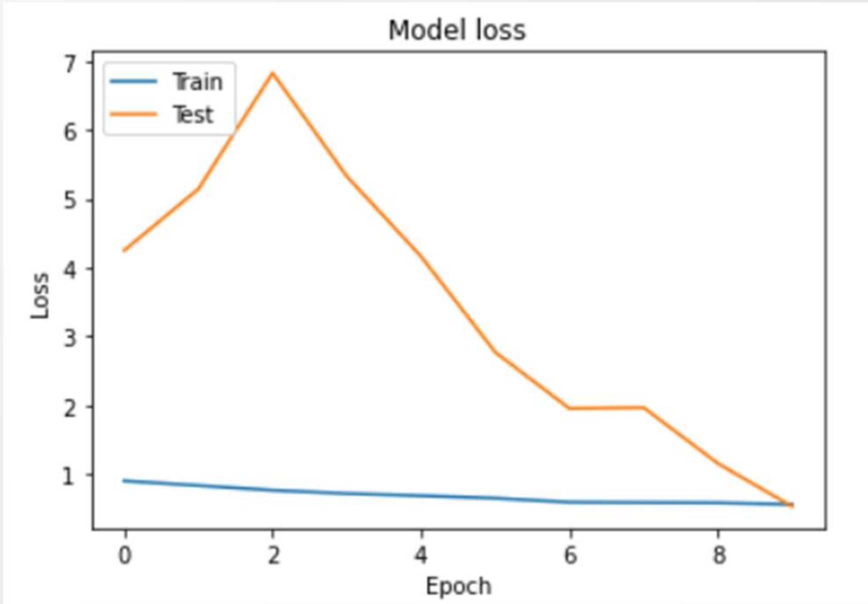
→Part of the code

→Input pictures and output pictures, label with bounding box

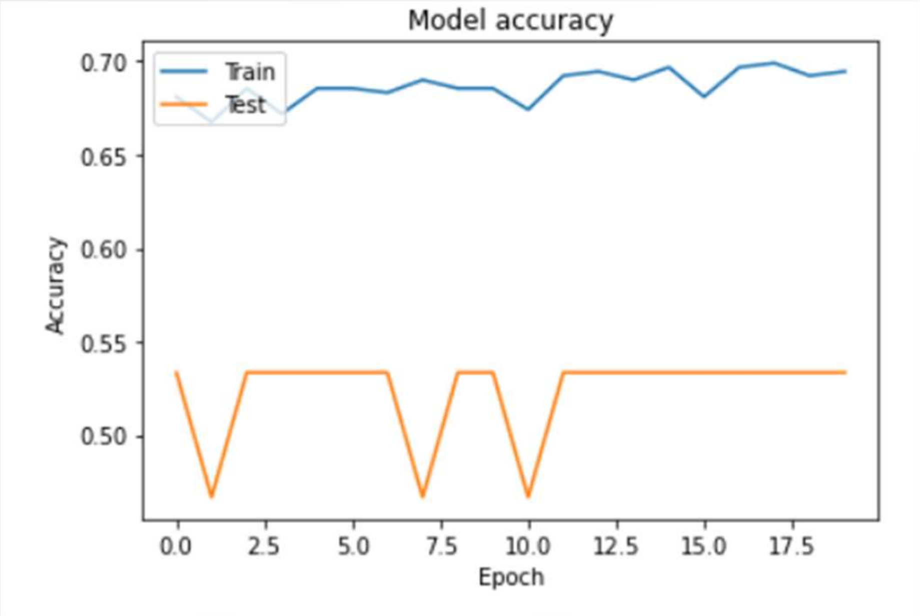
→Run the models separated and save as model file, in this main code use the function of load model to load the the model in for use.

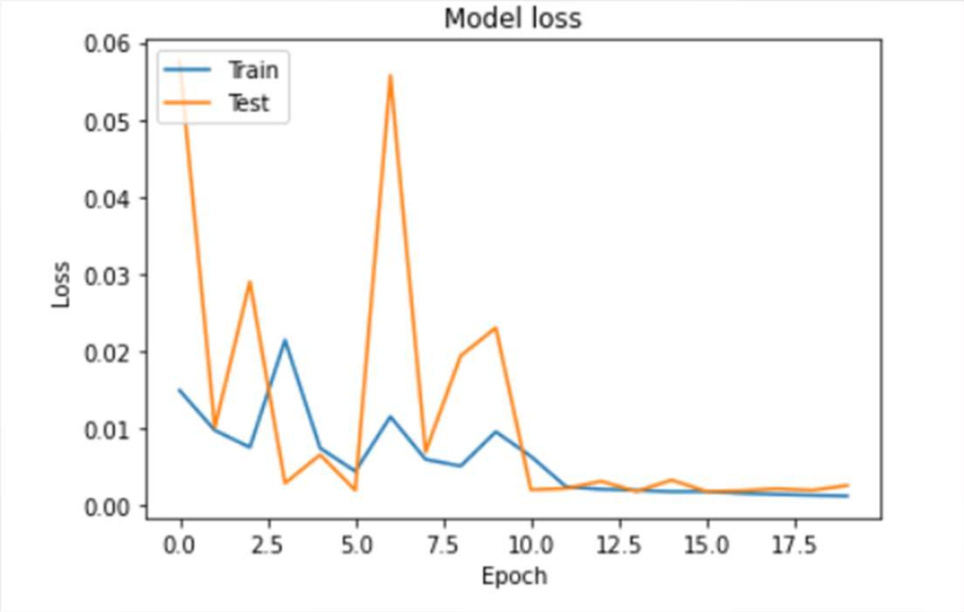
# Experiment Result-Classification



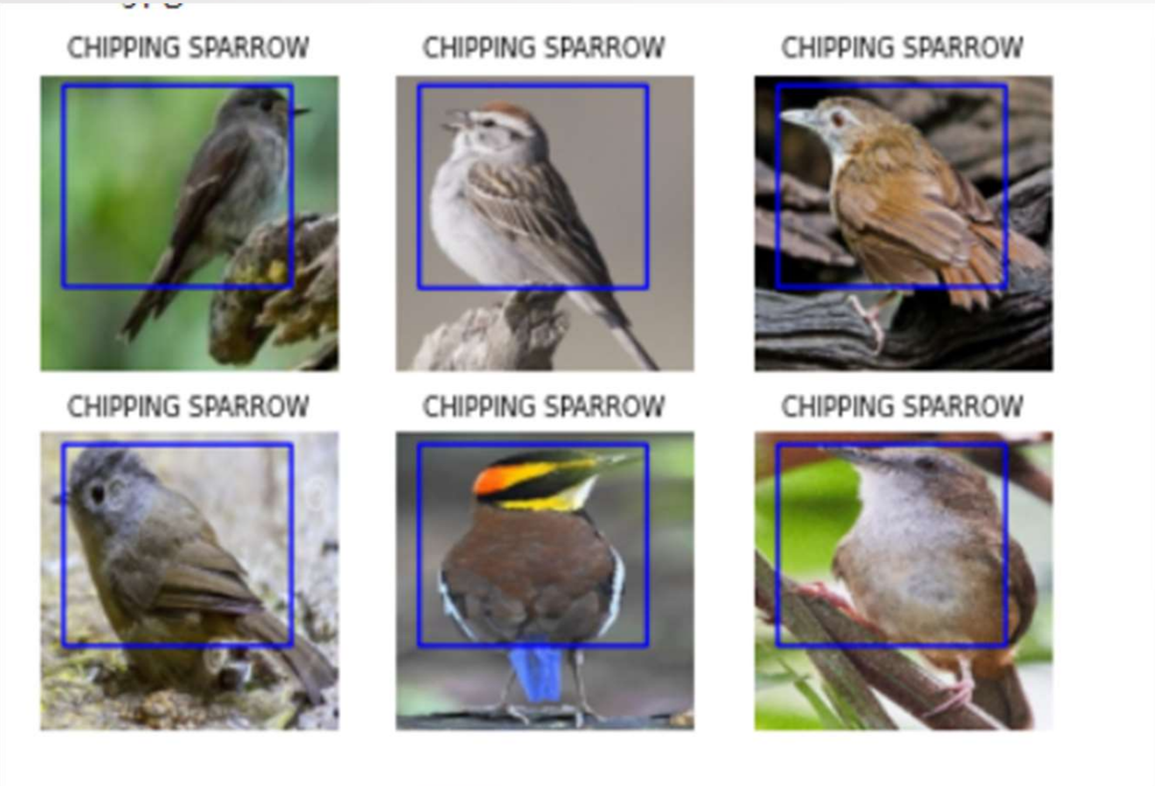



# Experiment Result-Location



# Experiment Result-Full Program




→ Use Google collab to show this result

→ The bounding box show the location of the birds

→ The tag above shows the species of the bird in the picture

# Disadvantages, Reasons and ways to improve

→ **Classification is not correct:**
The data set isn't big enough and birds are too familiar
Or not deep or complex enough to classify those species

→ **Location have nearby location**
Birds in the pictures have familiar location so can't really show if it is correct just from the pictures in the dataset

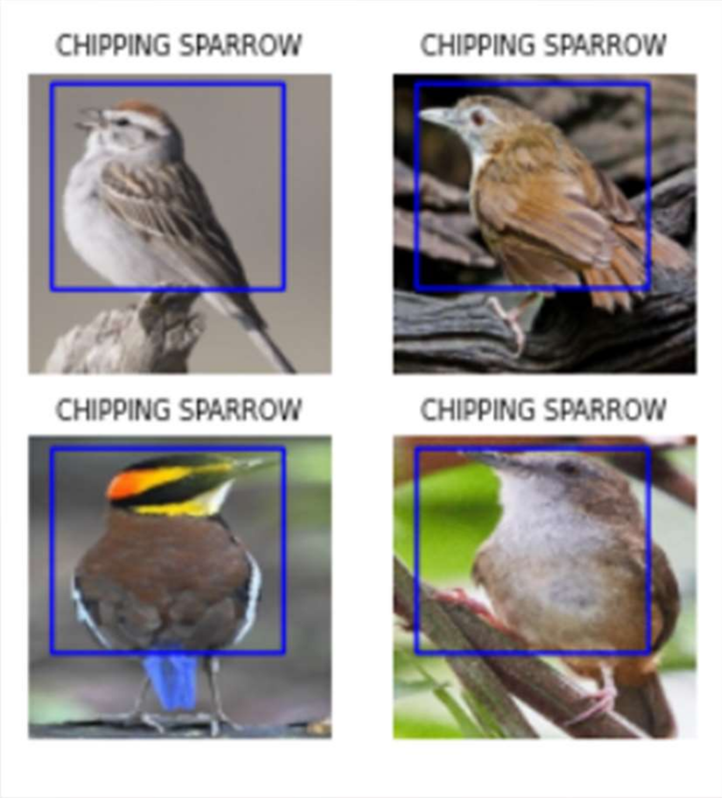
→ **Ways to improve:**

Extend the dataset :

- Make the dataset bigger
- Get new picture of birds that is in another location

Change architecture:

- Make the architecture more complex
- Use APIs in the program



# 05

# Conclusions & Future Work

Signal
Processing
&
Intelligent
Electronics
Lab

# Conclusions and future work

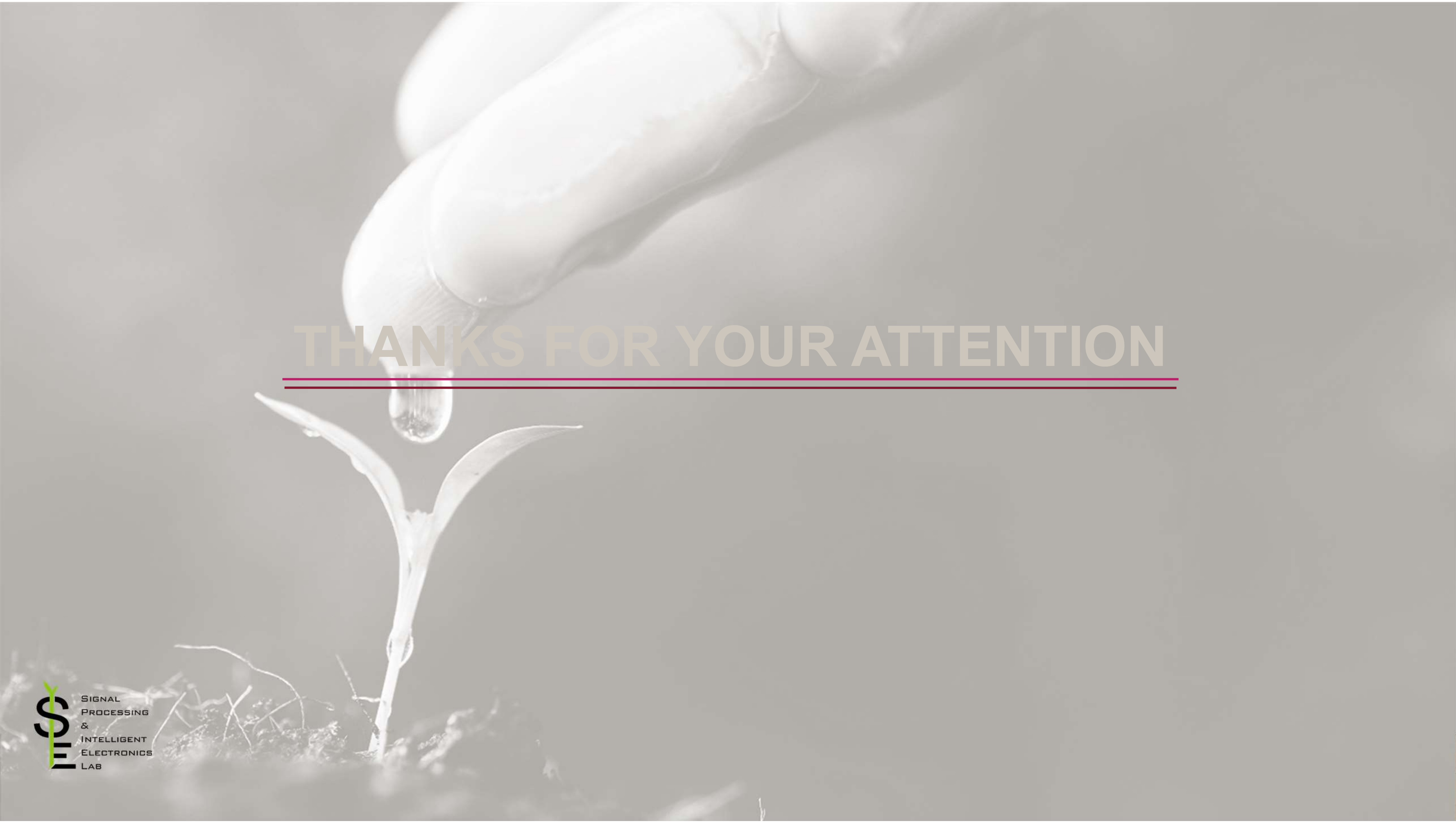
→ Combine two function in machine learning and also combine the knowledge that learned during the semester

→ Learn many information in this project

→ If the correction rate is higher, maybe it can be useful on birds finding and location on bird-watching or researching

# Reference

Dataset
https://www.kaggle.com/datasets/gpiosenka/100-bird-species
Code as reference
https://medium.com/nerd-for-tech/building-an-object-detector-in-tensorflow-using-bounding-box-regression-2bc13992973f
https://www.kaggle.com/code/gowrav143/final-birds-cnn/notebook
https://www.kaggle.com/code/samuiyoru/notebook6561b42683


THANKS FOR YOUR ATTENTION
Signal
Processing
&
Intelligent
Electronics
Lab