In [ ]:
```
import pandas as pd #to import data
import os
import sys
import matplotlib.pyplot as plt
import cv2 as cv
import numpy as np
from sklearn.utils import shuffle   #unsure if you need this
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, Dropout
from tensorflow.keras.optimizers import Adam
import warnings
warnings.filterwarnings('ignore')
```

In [ ]:
```
df_train = pd.read_csv('train/Training_set.csv')
df_test = pd.read_csv('test/Testing_set.csv')
df_train.sample(5)
```

Out[ ]:

	filename	label
2667	Image_2668.jpg	GREAT EGGFLY
1215	Image_1216.jpg	ULYSES
349	Image_350.jpg	RED POSTMAN
2288	Image_2289.jpg	GREEN CELLED CATTLEHEART
1520	Image_1521.jpg	BECKERS WHITE

In [ ]:
```
df_test.head()
```

Out[ ]:

	filename
0	Image_1.jpg
1	Image_2.jpg
2	Image_3.jpg
3	Image_4.jpg
4	Image_5.jpg

In [ ]:
```
df_train.info(), df_train.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 6499 entries, 0 to 6498
Data columns (total 2 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   filename  6499 non-null   object
 1   label     6499 non-null   object
dtypes: object(2)
memory usage: 101.7+ KB
```

Out[ ]: (None, (6499, 2))

In [ ]:
```
df_test.info(), df_test.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 2786 entries, 0 to 2785
Data columns (total 1 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   filename  2786 non-null   object
dtypes: object(1)
memory usage: 21.9+ KB
```

Out[ ]: (None, (2786, 1))

In [ ]:
```
df = pd.concat([df_train[['filename']], df_test[['filename']]], axis='rows')
df['label'] = [1]*len(df)
df.sample(3)
```

Out[ ]:

	filename	label
4614	Image_4615.jpg	1
994	Image_995.jpg	1
645	Image_646.jpg	1

In [ ]:
```
from sklearn.preprocessing import LabelEncoder

encoder = LabelEncoder()
df_train['label_en'] = encoder.fit_transform(df_train['label'])
df_train.sample(3)
```

Out[ ]:

	filename	label	label_en
6137	Image_6138.jpg	EASTERN COMA	26
4496	Image_4497.jpg	IPHICLUS SISTER	36
3650	Image_3651.jpg	CLEOPATRA	17

In [ ]:
```
non_target = []
MAIN_FILE = 'notbutterfly/'
for img in os.listdir(MAIN_FILE):
    non_target.append(img)
```

In [ ]:
```
non_target[:10]
```

Out[ ]:
```
['23933.jpg',
 '20584.jpg',
 '21842.jpg',
 '22393.jpg',
 '22387.jpg',
 '20590.jpg',
 '21856.jpg',
 '23927.jpg',
 '20221.jpg',
 '22436.jpg']
```

In [ ]:
```
non_target_labels = np.c_[non_target, [0]*len(non_target)]
non_df = pd.DataFrame(non_target_labels, columns=['filename', 'label'])
non_df.head()
```

Out[ ]:

	filename	label
0	23933.jpg	0
1	20584.jpg	0
2	21842.jpg	0
3	22393.jpg	0
4	22387.jpg	0

In [ ]:
```
non_target = []
IMAGE_SIZE = (40, 40)

for img in non_df['filename']:
    img = cv.imread('notbutterfly/' + img)
    img = cv.resize(img, IMAGE_SIZE)
    non_target.append(img/255.0)

from tensorflow.keras.preprocessing.image import ImageDataGenerator


datagen = ImageDataGenerator(
    rotation_range=30,  # Rotate images by 30 degrees @ random
    width_shift_range=0.2,  # Shift images horizontally 20% of total width @ random
    height_shift_range=0.2,  # Shift images vertically 20% of total height @ random
    shear_range=0.3,  # Apply shear transformation with a shear angle of 30 degrees
    zoom_range=0.3,  # Zoom images by up to 30% @ random
    horizontal_flip=True,  # Flip images horizontally @ random
    vertical_flip=False  # Do not perform vertical flips
)

# grayscale numpy array of images
images = non_target

# Generate augmented grayscale images
augmented_images = []
```

```
for image in images:
    num_generated_images = 0

    while num_generated_images < 60:
        augmented_image = datagen.random_transform(image)
        augmented_images.append(augmented_image)
        num_generated_images += 1

# Convert augmented grayscale images back to a NumPy array
augmented_images = np.array(augmented_images)
non_target = np.squeeze(augmented_images)
print(non_target.shape)
plt.imshow(non_target[0])
plt.axis('off')
plt.show()
```

```
(180000, 40, 40, 3)
```

In [ ]:
```
IMAGE_SIZE = (40,40)
IMAGE_SIZE_CLASS = (60,60)

train = []

train_class = []
train_labels = []

test = []

for img,label in zip(df_train['filename'], df_train['label_en']):
    img = cv.imread('train/'+img)
    det = cv.resize(img, IMAGE_SIZE)
    clas = cv.resize(img, IMAGE_SIZE_CLASS)
    train.append(det/255.0)
    train_class.append(clas/255.0)
    train_labels.append(label)

for img in df_test['filename']:
    img = cv.imread('test/' + img)
    img = cv.resize(img, IMAGE_SIZE)
    test.append(img/255.0)

plt.imshow(train[0])
plt.axis('off')
plt.show()
```

In [ ]:
```
images = list(train)+list(test)

# Generate augmented images
augmented_images_target = []
for image in images:
    num_generated_images = 0

    while num_generated_images < 6:
        augmented_image = datagen.random_transform(image)
        augmented_images_target.append(augmented_image)
        num_generated_images += 1

augmented_images_target = np.array(augmented_images_target)
target = np.squeeze(augmented_images_target)
print(target.shape)
```

```
(55710, 40, 40, 3)
```

In [ ]:
```
X = np.array(list(target)+ list(non_target))
Y = np.array([1]*len(target)+[0]*len(list(non_target)))

X.shape, Y.shape
```

Out[ ]:
```
((235710, 40, 40, 3), (235710,))
```

In [ ]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, stratify=Y)
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[ ]:
```
((176782, 40, 40, 3), (58928, 40, 40, 3), (176782,), (58928,))
```

In [ ]:
```
import tensorflow as tf
from tensorflow import keras
```

In [ ]:
```
# Define the CNN model
model = keras.Sequential()
model.add(keras.layers.Conv2D(32, kernel_size=(3, 3), activation='relu', input_shape=(40, 40, 3)))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(keras.layers.Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(keras.layers.MaxPooling2D(pool_size=(2, 2)))
model.add(keras.layers.Flatten())
model.add(keras.layers.Dense(64, activation='relu'))
model.add(keras.layers.Dense(1, activation='sigmoid'))

# Compile the model
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['accuracy'])

# Train the model
model.fit(X_train, Y_train, epochs=6, batch_size=32)
```

```
Epoch 1/6
5525/5525 [==============================] - 65s 12ms/step - loss: 0.1423 - accuracy: 0.9451
Epoch 2/6
5525/5525 [==============================] - 68s 12ms/step - loss: 0.0871 - accuracy: 0.9679
Epoch 3/6
5525/5525 [==============================] - 69s 12ms/step - loss: 0.0673 - accuracy: 0.9752
Epoch 4/6
5525/5525 [==============================] - 68s 12ms/step - loss: 0.0552 - accuracy: 0.9799
Epoch 5/6
5525/5525 [==============================] - 68s 12ms/step - loss: 0.0456 - accuracy: 0.9836
Epoch 6/6
5525/5525 [==============================] - 68s 12ms/step - loss: 0.0377 - accuracy: 0.9864
```

Out[ ]: <keras.src.callbacks.History at 0x2a8e0b8d0>

In [ ]:
```
model.evaluate(X_test, Y_test)
```

```
1842/1842 [==============================] - 7s 4ms/step - loss: 0.0531 - accuracy: 0.9805
```

Out[ ]: [0.05313996970653534, 0.9805185794830322]

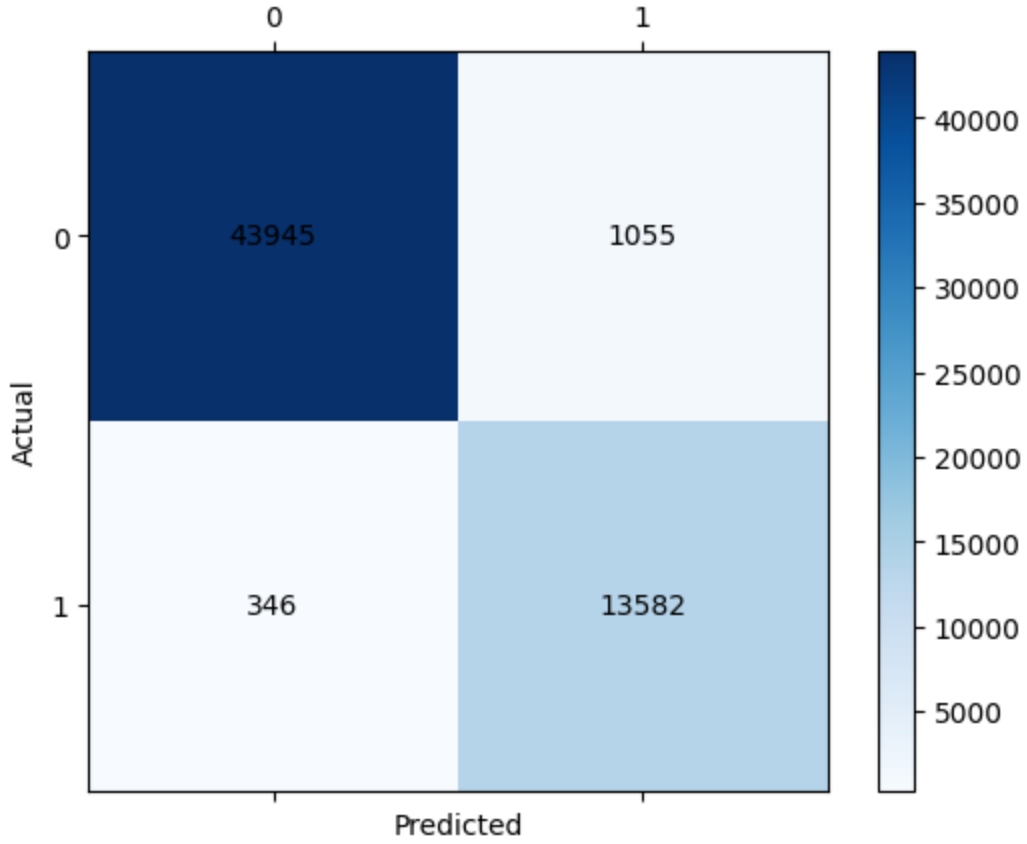
In [ ]:
```
tensors = tf.convert_to_tensor(np.array(X_test))
probabilities = model.predict(tensors)
threshold = 0.3
Y_pred = (probabilities > threshold).astype(int)[:,0]
Y_pred
```

```
1842/1842 [==============================] - 7s 4ms/step
```

Out[ ]: array([0, 0, 0, ..., 0, 0, 0])

In [ ]:
```
import matplotlib.pyplot as plt
from sklearn.metrics import confusion_matrix, classification_report
import numpy as np

cm = confusion_matrix(Y_test, Y_pred)

fig, ax = plt.subplots()
cax = ax.matshow(cm, cmap=plt.cm.Blues)
plt.colorbar(cax)

classes = np.unique(Y_test)
ax.set_xticks(np.arange(len(classes)))
ax.set_yticks(np.arange(len(classes)))
```

```
ax.set_xticklabels(classes)
ax.set_yticklabels(classes)

for i in range(len(classes)):
    for j in range(len(classes)):
        plt.text(j, i, str(cm[i, j]), ha='center', va='center')

plt.xlabel("Predicted")
plt.ylabel("Actual")
plt.show()
```



In [ ]:
```
cr = classification_report(Y_test, Y_pred)
print(cr)
```

	precision	recall	f1-score	support
0	0.99	0.98	0.98	45000
1	0.93	0.98	0.95	13928
accuracy			0.98	58928
macro avg	0.96	0.98	0.97	58928
weighted avg	0.98	0.98	0.98	58928