CS 4661 Project Report

Yum or Yuck Butterfly Mimic 2022

Fall 2022

Dr. Mohammad Pourhomayoun

Hugo Izquierdo, Karen Quan, Kenneth Lieu, Minsu Lee, Peter Han

Group 04

Project Description:

This project is about identifying the different species of butterflies by training the AI. Images of butterflies will be given to help train the AI to spot their characteristics. This allows the AI to understand which characteristics belong to which butterfly, effectively classifying them in a quick and orderly manner. By classifying the butterfly we are able to realize which butterfly will be Yum (Non-poisonous) or Yuck (Poisonous) based on the butterfly characteristics.

Details:

Some butterflies lay eggs purposely on toxic plants like milkweed and pipevine. When the eggs hatch and the caterpillars eat their host plant, it causes them to taste terrible to predators. Not all butterflies use this method to discourage predators, however. Some caterpillars that are not born on toxic host plants grow to mimic caterpillars that are. These caterpillars' mimicry is so similar that birds have trouble telling the difference between the two types of caterpillars. In our dataset, some are toxic (yuck), while some only resemble toxic butterflies (yum). Being able to accurately identify the different types of butterflies is key to our project.

Goals:

To use basic machine learning algorithms to make a model for classifying butterflies based on images. Based on our chosen feature sets, we aim to label butterflies as "yum" or "yuck" based on the physical patterns on the butterflies themselves, which can be seen in the images. We will use the techniques learned throughout the course to evaluate the accuracy of our machine learning model, as well as plot the ROC and AUC curves.

Data:

The data that we were provided with in our *Yum or Yuck Butterfly Mimic 2022* Dataset include:

- Image - image ID (ex. ggc1e08cbc)
- Name - butterfly species name (ex. Monarch)
- Stage - stage of butterfly lifecycle (All Adults)
- Side - visible wing side in the image (ex. Ventral, Dorsal, Both)

The data used for Random Forest Classifier, Logistic Regression Classifier, and KNN Classifier:

- Name - butterfly species name (ex. Monarch)
- Side - visible wing side in the image (ex. Ventral, Dorsal, Both)
- Pixel array - reshaped one-dimensional array of pixels after reading each image

Shape of X: (853, 150531)

Shape of y: (853,)

Developed Methods:

Using TensorFlow and Keras, we will first create an image classifier, then create a Feature Matrix with the identifying attributes of butterflies in our images. Then we will label each as “yum” or “yuck”.

Using Random Forest Classifier from sci kit learn ensemble and Logistic Regression from sci kit learn linear model. First, create a dataframe using the given label csv. Then, read each image using matplotlib image, reshape the returned array of pixels into a one dimensional array. The feature matrix will be created using these one dimensional arrays. Each column in the feature matrix will represent a pixel from the array. Additionally, one hot encoding for categorical features such as the side of the image displayed will be added to the feature matrix. Labels, yum or yuck, are then assigned according to the butterfly name. The dataset is split into training and testing sets, and used accordingly into each model. The training dataset is used to train the model, and the testing dataset is used to test the model. Finally, the accuracy of the model is evaluated and the Roc Curve and AUC are calculated and plotted.

Note - 1: only files obtained from kaggle project page intended for training were used in this algorithm, the testing files from this project page were not used.

Note - 2: images within the training file are not of uniform shape, images larger than (224,224,3) were resized using image.resize from matplotlib and Image.ANTIALIAS from PIL

Developed Algorithms:

The algorithms that we worked with in our final project include:

- Random Forest Model
- KNN Classification

- Image Classifiers
- Logistic Regression

Tools:

The tools that we used in our final project include:

- Jupyter notebook
- scikit-learn
- Pandas
- Numpy
- TensorFlow
- Google Collab/Doc/Slides
- Kaggle
- Matplotlib: image, pyplot
- PIL: Image

Developed Codes:

Logistic Regression, KNN, and Random Forest:

After clicking on the link, click on google colab to view.

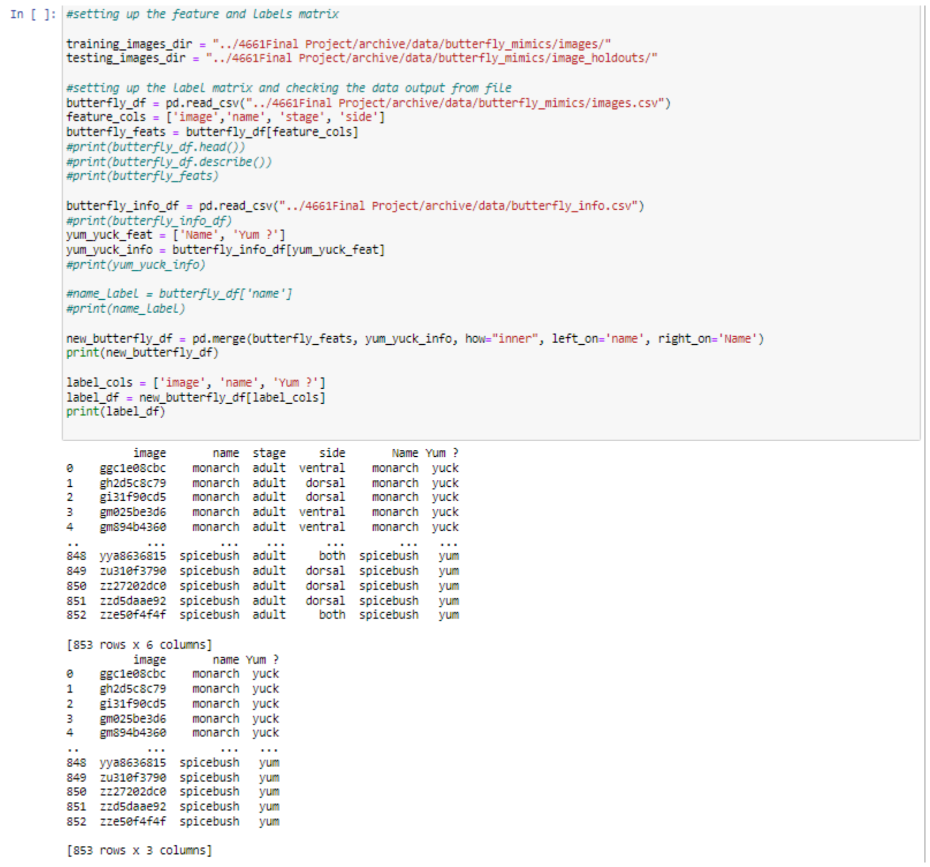
https://drive.google.com/file/d/1ntwG9-GStK_GzmSJUbknlK_ZR2Zfc-i4/view?usp=sharing

Tensor Flow and Keras:

We cover the major concepts developed. For the full set of implemented code, click the link below to the .ipynb file in google collab.

https://colab.research.google.com/drive/1krToFM5Vt4fa9yK3f0hOJgBDkh1Uk1dm?usp=sharing

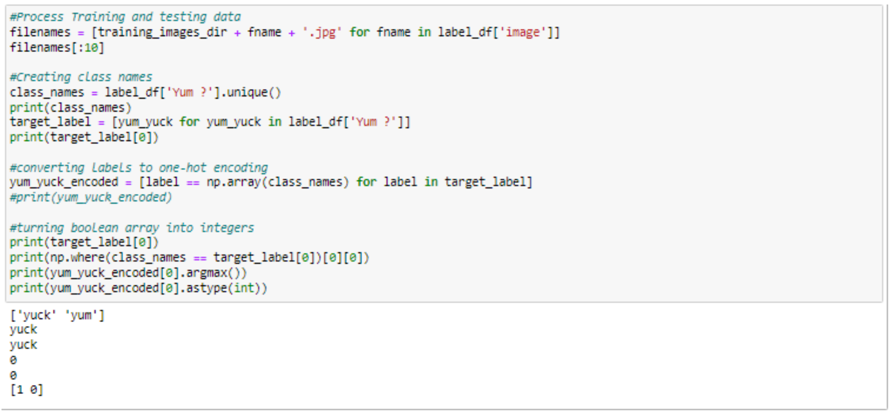
We import the provided data sets to create and examine our dataframe.



```
In [ ]: #setting up the feature and labels matrix

        training_images_dir = "../4661Final Project/archive/data/butterfly_mimics/images/"
        testing_images_dir = "../4661Final Project/archive/data/butterfly_mimics/image_holdouts/"

        #setting up the label matrix and checking the data output from file
        butterfly_df = pd.read_csv("../4661Final Project/archive/data/butterfly_mimics/images.csv")
        feature_cols = ['image','name', 'stage', 'side']
        butterfly_feats = butterfly_df[feature_cols]
        #print(butterfly_df.head())
        #print(butterfly_df.describe())
        #print(butterfly_feats)

        butterfly_info_df = pd.read_csv("../4661Final Project/archive/data/butterfly_info.csv")
        #print(butterfly_info_df)
        yum_yuck_feat = ['Name', 'Yum ?']
        yum_yuck_info = butterfly_info_df[yum_yuck_feat]
        #print(yum_yuck_info)

        #name_label = butterfly_df['name']
        #print(name_label)

        new_butterfly_df = pd.merge(butterfly_feats, yum_yuck_info, how="inner", left_on='name', right_on='Name')
        print(new_butterfly_df)

        label_cols = ['image', 'name', 'Yum ?']
        label_df = new_butterfly_df[label_cols]
        print(label_df)
```

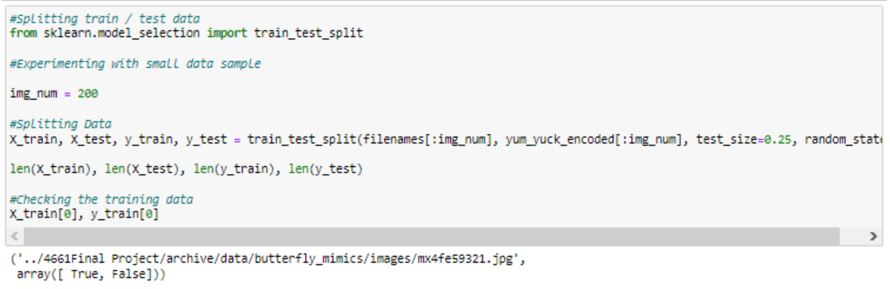
```
          image       name  stage     side       Name Yum ?
0    ggc1e08cbc    monarch  adult  ventral    monarch  yuck
1    gh2d5c8c79    monarch  adult   dorsal    monarch  yuck
2    gi31f90cd5    monarch  adult   dorsal    monarch  yuck
3    gm025be3d6    monarch  adult  ventral    monarch  yuck
4    gm894b4360    monarch  adult  ventral    monarch  yuck
..          ...        ...    ...      ...        ...   ...
848  yya8636815  spicebush  adult     both  spicebush   yum
849  zu310f3790  spicebush  adult   dorsal  spicebush   yum
850  zz27202dc0  spicebush  adult   dorsal  spicebush   yum
851  zzd5daae92  spicebush  adult   dorsal  spicebush   yum
852  zze50f4f4f  spicebush  adult     both  spicebush   yum

[853 rows x 6 columns]
          image       name Yum ?
0    ggc1e08cbc    monarch  yuck
1    gh2d5c8c79    monarch  yuck
2    gi31f90cd5    monarch  yuck
3    gm025be3d6    monarch  yuck
4    gm894b4360    monarch  yuck
..          ...        ...   ...
848  yya8636815  spicebush   yum
849  zu310f3790  spicebush   yum
850  zz27202dc0  spicebush   yum
851  zzd5daae92  spicebush   yum
852  zze50f4f4f  spicebush   yum

[853 rows x 3 columns]
```

Then we one hot encode the labels (species) to be used in classifying.



```
#Process Training and testing data
filenames = [training_images_dir + fname + '.jpg' for fname in label_df['image']]
filenames[:10]

#Creating class names
class_names = label_df['Yum ?'].unique()
print(class_names)
target_label = [yum_yuck for yum_yuck in label_df['Yum ?']]
print(target_label[0])

#converting labels to one-hot encoding
yum_yuck_encoded = [label == np.array(class_names) for label in target_label]
#print(yum_yuck_encoded)

#turning boolean array into integers
print(target_label[0])
print(np.where(class_names == target_label[0])[0][0])
print(yum_yuck_encoded[0].argmax())
print(yum_yuck_encoded[0].astype(int))
```

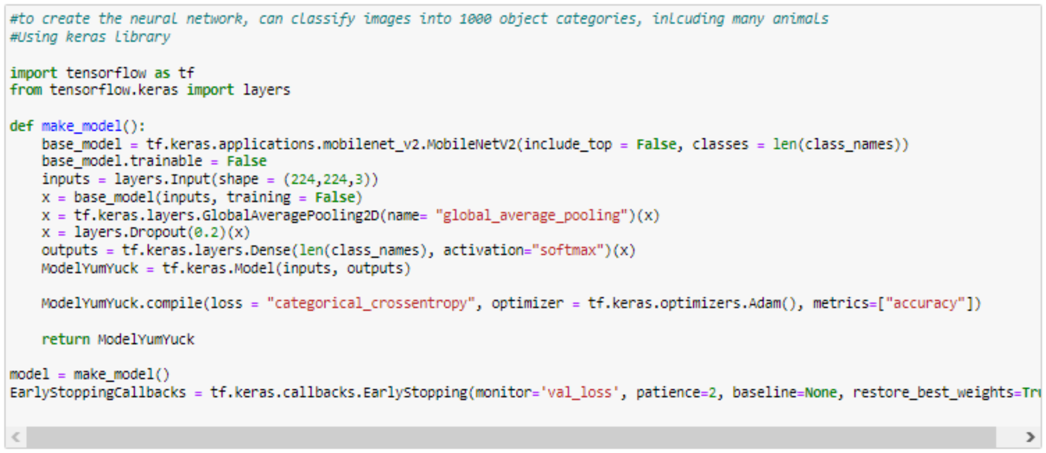
```
['yuck' 'yum']
yuck
yuck
0
0
[1 0]
```



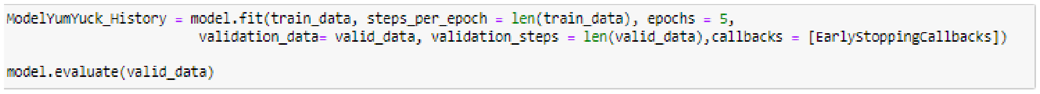
```
#Splitting train / test data
from sklearn.model_selection import train_test_split

#Experimenting with small data sample

img_num = 200

#Splitting Data
X_train, X_test, y_train, y_test = train_test_split(filenames[:img_num], yum_yuck_encoded[:img_num], test_size=0.25, random_stat

len(X_train), len(X_test), len(y_train), len(y_test)

#Checking the training data
X_train[0], y_train[0]
```

```
('../4661Final Project/archive/data/butterfly_mimics/images/mx4fe59321.jpg',
 array([ True, False]))
```

Using the tensorflow and keras libraries, we create a neural network to help classify the images with more accurate image processing handling. We fit the data into the neural network model and keep track of its accuracy and possible data loss to analyze our results which we will discuss in the Results section.



```python
#to create the neural network, can classify images into 1000 object categories, inlcuding many animals
#Using keras library

import tensorflow as tf
from tensorflow.keras import layers

def make_model():
    base_model = tf.keras.applications.mobilenet_v2.MobileNetV2(include_top = False, classes = len(class_names))
    base_model.trainable = False
    inputs = layers.Input(shape = (224,224,3))
    x = base_model(inputs, training = False)
    x = tf.keras.layers.GlobalAveragePooling2D(name= "global_average_pooling")(x)
    x = layers.Dropout(0.2)(x)
    outputs = tf.keras.layers.Dense(len(class_names), activation="softmax")(x)
    ModelYumYuck = tf.keras.Model(inputs, outputs)

    ModelYumYuck.compile(loss = "categorical_crossentropy", optimizer = tf.keras.optimizers.Adam(), metrics=["accuracy"])

    return ModelYumYuck

model = make_model()
EarlyStoppingCallbacks = tf.keras.callbacks.EarlyStopping(monitor='val_loss', patience=2, baseline=None, restore_best_weights=Tr
```



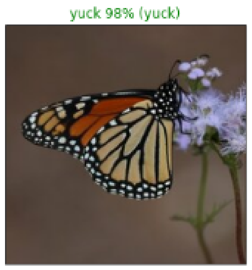
```python
ModelYumYuck_History = model.fit(train_data, steps_per_epoch = len(train_data), epochs = 5,
                        validation_data= valid_data, validation_steps = len(valid_data),callbacks = [EarlyStoppingCallbacks])

model.evaluate(valid_data)
```

Finally we create an array of predictions of the probabilities that each butterfly in the test set images is “yum” or “yuck.” Also, we display the images with the predicted label and its probability with “yum” in green text and “yuck” in red text for better user experience.

```
#Making prediction on the validation data

predictions = model.predict(valid_data)
predictions
```

```
2/2 [==============================] - 2s 361ms/step

array([[0.9727411 , 0.02725888],
       [0.9834114 , 0.01658867],
       [0.04412269, 0.9558773 ],
       [0.58371925, 0.4162808 ],
       [0.33470437, 0.66529554],
       [0.20002973, 0.79997027],
       [0.983878  , 0.01612194],
       [0.9699346 , 0.0300655 ],
       [0.9975333 , 0.00246667],
       [0.9862918 , 0.01370821],
       [0.02786494, 0.97213507],
       [0.990725  , 0.00927505],
       [0.977457  , 0.02254303],
       [0.96568066, 0.0343193 ],
       [0.09854527, 0.90145475],
       [0.96732104, 0.03267892],
       [0.98685294, 0.01314703],
       [0.9511133 , 0.04888667],
       [0.01176454, 0.9882355 ],
       [0.9515672 , 0.04843284],
       [0.93055314, 0.06944691],
       [0.96755713, 0.03244289],
       [0.9920536 , 0.00794635],
       [0.9368869 , 0.06311307],
       [0.95497346, 0.04502653],
       [0.9845083 , 0.01549175],
       [0.9920265 , 0.00797354],
       [0.9606406 , 0.03935931],
       [0.6832692 , 0.31673086],
       [0.09063257, 0.90936744],
       [0.99020255, 0.00979742],
       [0.97349054, 0.02650948],
       [0.9766311 , 0.0233689 ],
       [0.9577141 , 0.04228598],
       [0.6712662 , 0.32873386],
       [0.8951421 , 0.10485793],
       [0.020955  , 0.979045  ],
       [0.9904128 , 0.00958721],
       [0.9836113 , 0.01638866],
       [0.94153607, 0.05846398],
       [0.9854897 , 0.01451021],
       [0.92925173, 0.07074834],
       [0.980339  , 0.01966103],
       [0.95276535, 0.04723462],
       [0.9930775 , 0.00692245],
       [0.06465631, 0.9353436 ],
       [0.24793746, 0.75206256],
       [0.75312144, 0.24687853],
       [0.9771359 , 0.02286416],
       [0.8631143 , 0.13688569]], dtype=float32)
```

```
#function to view the prediction, ground truth label and image for sample n.

def plot_pred(prediction_probabilities, labels, images, n=1):

  prediction_prob, actual_label, image = prediction_probabilities[n], labels[n], images[n]

  # Get the prediction label
  prediction_label = get_pred_label(prediction_prob)

  # Plot image & remove ticks
  plt.imshow(image)
  plt.xticks([])
  plt.yticks([])

  # Change the color of the title depending on if the prediction is right or wrong
  if prediction_label == actual_label:
    color = "green"
  else:
    color = "red"

  plt.title("{} {:2.0f}% ({})".format(prediction_label, np.max(prediction_prob)*100, actual_label), color=color)

# Example prediction, original image and truth label
plot_pred(prediction_probabilities=predictions, labels=val_labels,images=val_images)
```



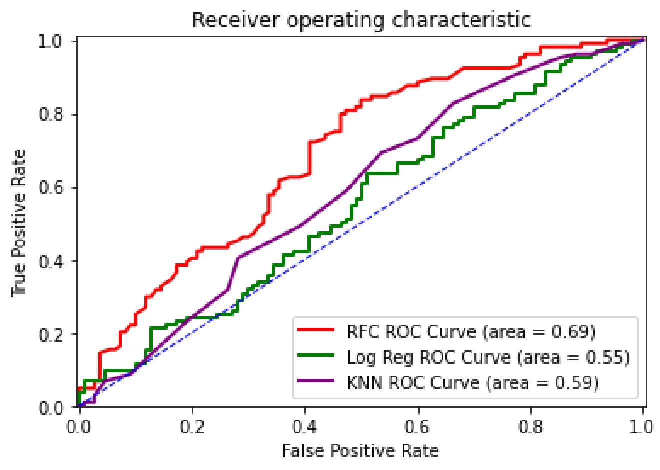
Final Results:

Out of Random Forest Classifier, Logistic Regression Classifier, and K Nearest Neighbors Classifier, the Random Forest Classifier had the highest accuracy with about 61%, followed by K Nearest Neighbors with about 57%, and lastly Logistic Regression Classifier with about 51%. In this case, logistic regression performed worse than KNN. The reason is most likely the large number of features and that the features, basically individual pixels and the side of the image, don't provide a lot of information about whether the butterfly is edible or not. Knn performed better because this classifier doesn't need a lot of information gain from the features, it just looks around at its "neighbors" to decide.

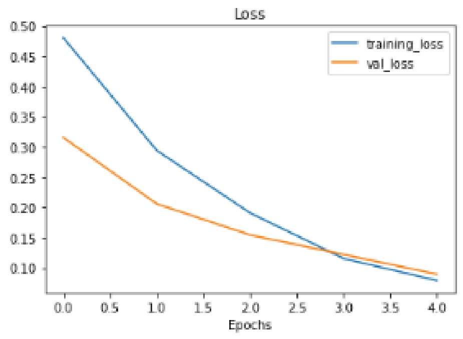
Using tensorflow, we get accurate predictions on the probability of whether or not the butterflies in the images are edible ("yum") or inedible ("yuck"). At 98% certainty we get probabilities of above 90% of being either "yum" or "yuck" from our training methods.

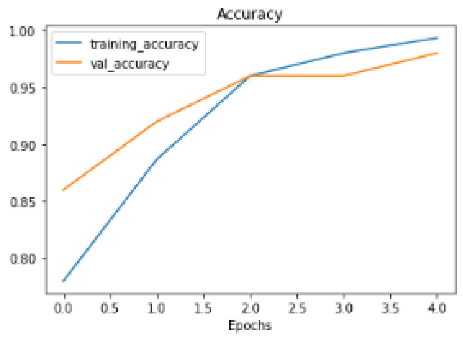
Performance(s): AUC Curve, ROC Curve to visualize the comparison numbers between the various ML algorithms listed above.

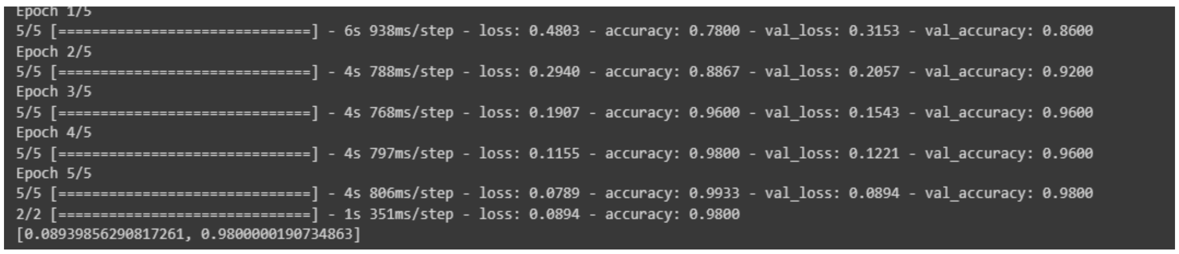
Roc Curve and AUC for Random Forest Classifier (Red Line), Logistic Regression Classifier (Green Line), and K Nearest Neighbors Classifier (Purple Line).



In using tensorflow and keras, we see that our model gets more accurate as we fit our data more times (higher number of epochs). At five epochs, we achieve 99.3% training accuracy and 98% value accuracy.







```
Epoch 1/5
5/5 [==============================] - 6s 938ms/step - loss: 0.4803 - accuracy: 0.7800 - val_loss: 0.3153 - val_accuracy: 0.8600
Epoch 2/5
5/5 [==============================] - 4s 788ms/step - loss: 0.2940 - accuracy: 0.8867 - val_loss: 0.2057 - val_accuracy: 0.9200
Epoch 3/5
5/5 [==============================] - 4s 768ms/step - loss: 0.1907 - accuracy: 0.9600 - val_loss: 0.1543 - val_accuracy: 0.9600
Epoch 4/5
5/5 [==============================] - 4s 797ms/step - loss: 0.1155 - accuracy: 0.9800 - val_loss: 0.1221 - val_accuracy: 0.9600
Epoch 5/5
5/5 [==============================] - 4s 806ms/step - loss: 0.0789 - accuracy: 0.9933 - val_loss: 0.0894 - val_accuracy: 0.9800
2/2 [==============================] - 1s 351ms/step - loss: 0.0894 - accuracy: 0.9800
[0.08939856290817261, 0.9800000190734863]
```

Team Member Responsibility:

- Hugo Izquierdo
    - Image Processing and classification algorithm (classifying the butterflies as Yum or Yuck)
- Karen Quan
    - Data Classification, Written Report
- Kenneth Lieu
    - Data Classification, Written Report
- Minsu Lee
    - Data Organization and Classification, Written Report, Powerpoint Slides
- Peter Han
    - Image processing algorithm, Classification Algorithm, Written Report