Udacity Final Project
Banana Ripeness Through Image Classification

Definition

Project Overview

Bananas are one of the most consumed fruits around the world due to its accessibility, cost, and health benefits.

Bananas are also known for its fast ripening and most consumers can determine the ripeness level visually, but the people with color blindness have trouble determining when the banana is ripe enough to eat. The studies show that color blindness affects approximately 1 in 12 men and 1 in 200 women in the world [1].

There has been a lot of researches and applications of machine learning and deep learning in the food industry. One of the research papers uses various image classification architectures such as AlexNet, ResNet, GoogleNet, etc. to classify the different type of fruits from an image [2]. Another study resolves similar problem of classifying the ripeness level of the strawberry using CNN.[3]

In this project, we propose to develop a classification model that predicts the ripeness of the banana using the dataset collected through web scraping the Google Images. The labels of the images will be created through visual inspection of the gathered image.

Problem Statement

There are those that have color blindness which make them hard to determine whether the banana is ripe enough to eat. There are several devices and solutions offered for detection of the ripeness level, but these devices are mainly for agricultural applications and are very expensive. Also, a lot of the current solutions use a gas emission level of the banana to determine the ripeness level. This has a lot of limitation as it requires a physical device for ripeness detection.

This project proposes image classification model that determines the ripeness level of the banana while being easily accessible by everyone.

The flow of our product is going to be as follows:
1. The users input the image.
2. The model preprocesses the image. This includes re-sizing to 224x224 and apply normalization.
3. The initial pretrained model, ResNet101, analyzes if the input image is a banana. If the input image is not a banana, the model does not accept the image and output as an error.
4. After banana is detected, the model analyzes and predicts the ripeness class of green, ripe, and overripe.
5. The model outputs a message to the user on the ripeness of the banana or that the input image is not a banana.

To achieve the product as presented above, the steps of developing the product is as follows:
1. We collect data / images through web scraping and create labels for bananas through visual inspection. Afterwards, analysis is performed on the dataset and gather insights about the data. Also, we format our directory in the form that PyTorch, a deep learning framework, can accept and split the data into train, test, and validation
2. We gather additional data of images other than banana. This is to validate our pretrained model to make sure that the pretrained model does not predict a false positive or falsely predict banana when it is not.
3. Using the collected images of bananas and not-bananas, we try various pretrained architecture such as ResNet101 and VGG16 to determine the performance of the banana object prediction.
4. For classification of ripeness, baseline model of simple CNN with fully connected layers that achieved the accuracy of 80%.
5. For the development of the final model, we use transfer learning from pretrained models such as ResNet101 to train and develop our final model. For development of the model, we use PyTorch on SageMaker server to accelerate our training
6. For final refinement of the model, we perform hyperparameter tuning using Sage Maker and select the best performing model and its hyperparameter
7. The model is deployed through SageMaker and the script for outputting a message to the user is created.

Metrics

The collected dataset is not imbalanced, and all labels are deemed to be equally important. Thus, we will use accuracy as our main metric for our evaluation. The accuracy is formulated as follows:

$$Accuracy = \frac{True\ Postiive + True\ Negative}{(True\ Positive + False\ Positive + False\ Negative + False\ Negative)}$$

For our object detection model, it would require the model to predict the banana object when an image is provided. There are 1000 classes to predict from which make the model prediction extremely hard. We thus use Accuracy@5 where if the model predicts banana within its top 5 prediction, we accept as true positive. The reason for choosing top 5 is that having top 5 would minimize the false negative (predicting banana as not banana) which is deemed to be critical in our application. Even though it is best to minimize for across all prediction, having higher false positive is not as critical as false negative in our case, because we would like all banana objects to classified as banana.

Another candidate for evaluation metric could be F1 score which is the harmonic mean between precision and recall, but since we consider the true positive and the true negative rate to be more important than false negative and false positive, we ultimately chose accuracy.

Analysis

Data Exploration and Visualization

In this section, we analyze the data that were collected for this problem. The total images collected is 338, where 129 images are ripe, 137 are not ripe, and 72 images are overripe.
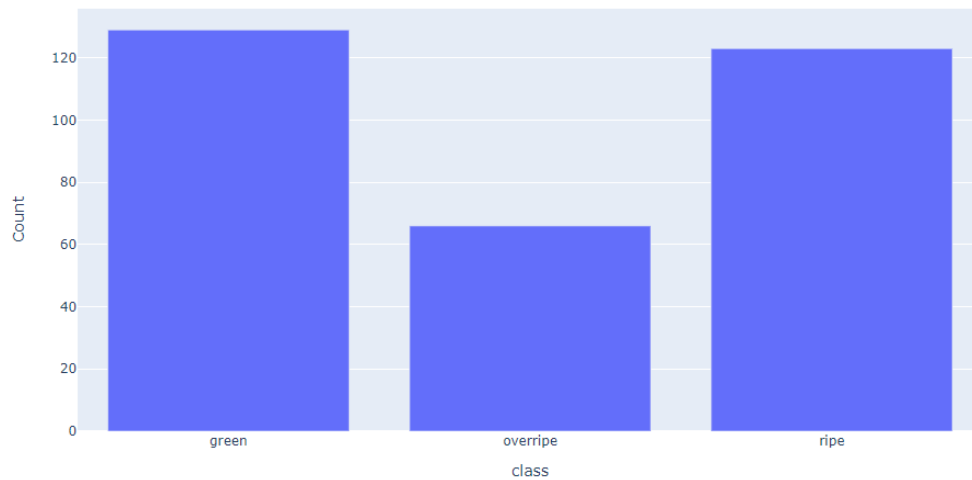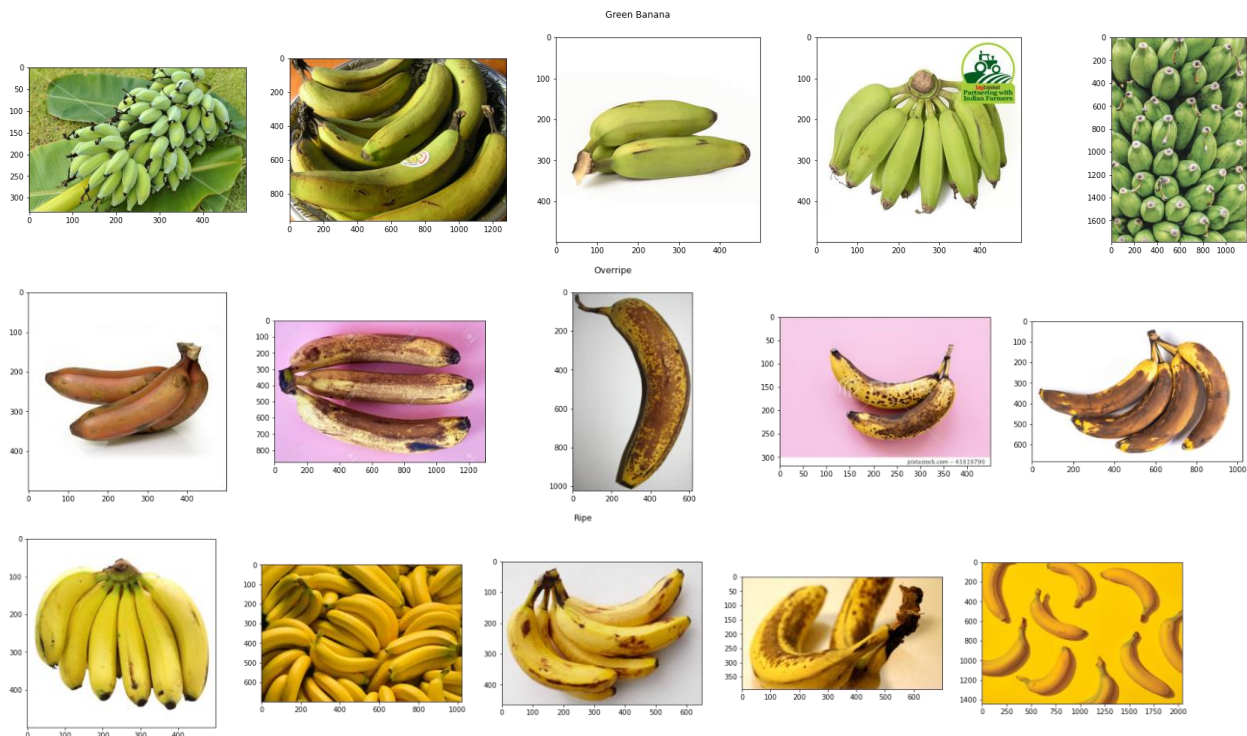

Figure 1 - Distribution of the classes

Figure 2 – Sample images of the data

From Figure 2, we observe that all images tend to have different sizes which would require a re-sizing for training. Additionally, we can observe that every image has different rotation, tilt, the number of bananas, different background, etc. This is good as the model can generalize well regardless of which images they receive as they have been trained on images with diverse settings.

As mentioned previously, we seek to implement a pretrained model for detecting banana in the image. To validate our pretrain model, we would need to also validate on both bananas and objects other than bananas. For collection of the dataset for objects other than bananas, we use the test set of Kaggle Fruits 360 dataset, which contains 22204 images to validate from [7].

Figure 3 – Sample Images from Kaggle Fruits 360 dataset
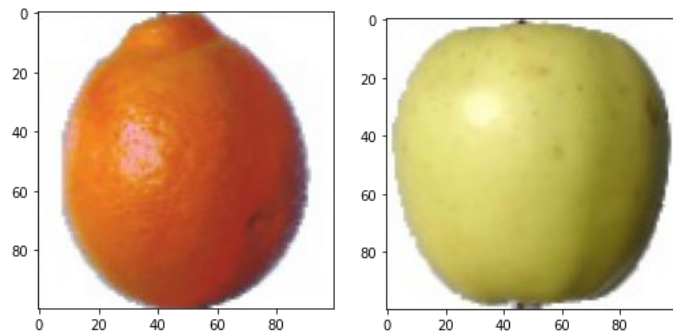
Prior to training the model, the structuring the images and its directory are required for almost any image training frameworks. The images were firstly split into train, test, and validation where the dataset is split between train and test by 80-20 split. Afterwards, the train set were split again for validation set using 80-20 split. The split images were placed in the folders of validation, test, and train for training of the model.

Algorithm and Techniques

There are many algorithm and techniques applied for the development of the project. The initial algorithm we utilize is the pretrained models from ImageNet project for detection of the banana object. For us to develop an object classification model ourselves, we would require a large dataset with long hours of training. Instead we use the pretrained model from the ImageNet project. The ImageNet project aims to train a model that can correctly classify an input image into 1000 separate object categories using millions of images and the trained model is readily available to the public. Utilizing the pre-trained models would allow us to develop more robust banana detection model while not requiring training the model. Instead of using one, we try various pretrained models such as VGG-16 and ResNet to select the architecture with highest performance for prediction of the banana object. The performance evaluation is done through use of our collected data, and the Fruit360 dataset from Kaggle.

Another algorithm or technique we utilize is transfer learning. Transfer learning is a method of using the representations / information from a trained model for another application or model that need to be trained on different data for similar task. Transfer learnings directly use the pre-trained model which are the models we utilize in our object detection model. With transfer learning, the deep learning application that solve for complex applications such as

vision-related tasks could be built and implemented quicker. Like our object detection model, we will validate on various architectures and select the best performing model.

The next techniques we use is training with Amazon SageMaker and PyTorch, especially for hyperparameter tuning for ripeness classification model. There are many advantages of using SageMaker for training. There are various hyperparameters that would require tuning. With SageMaker, we can train high-quality models fast which utilizes parallel processing with GPU all through the cloud base. This means that regardless of the hardware we use, the training time will be similar throughout different systems. This is especially required for our application because image training with size 224x224 means that our input is in the dimension of 50176. This would require large computational power to train the model. Additionally, the built-in log system makes it extremely easy to debug the issues and the deployment of the model can be done with ease if we plan on deploying the model to a web application.
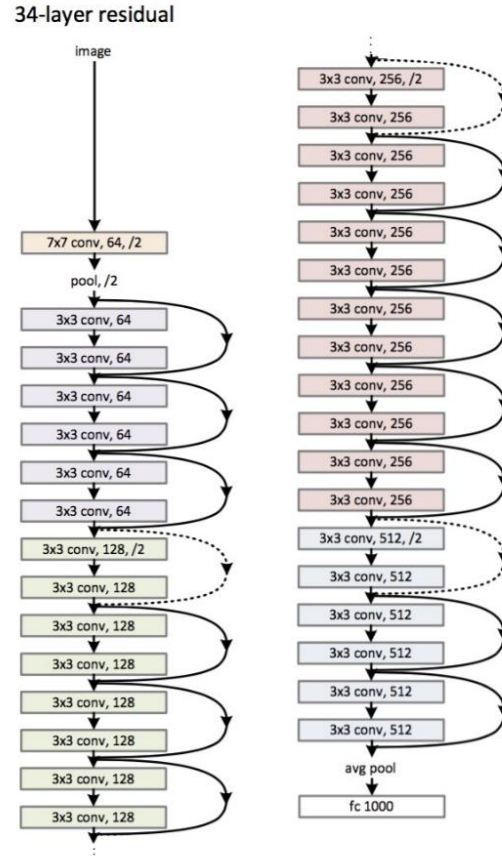

Figure 4 – Example of pretrained model architecture. The shown image is ResNet34 [4]

Benchmark Model

In the project, the benchmark model is the following:

1. For the ripeness level detection, the simple CNN with fully connected layer architecture is served as our baseline model. We chose this model as it is a baseline CNN model presented in the PyTorch documentation [5].

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1          [-1, 6, 112, 112]              168
         MaxPool2d-2            [-1, 6, 56, 56]                0
            Conv2d-3           [-1, 16, 28, 28]              880
         MaxPool2d-4           [-1, 16, 14, 14]                0
            Linear-5                  [-1, 120]          376,440
            Linear-6                   [-1, 84]           10,164
            Linear-7                    [-1, 3]              255
================================================================
Total params: 387,907
Trainable params: 387,907
Non-trainable params: 0
----------------------------------------------------------------
```

Figure 5 - The summary of the Baseline Model

The baseline model with the parameters in Figure 5 achieved 85% accuracy which will be the target accuracy we aim to outperform.

2. For banana object detection, when we input the banana image, the 94.5% accuracy is chosen as the benchmark accuracy since this accuracy is the highest achieved using the ImageNet. We expect the model to outperform the performance in the ImageNet since the ImageNet models are trained with 1000 classes whereas for our application, we only involve banana class.

Methodology

Data Preprocessing

The first step of the data preprocessing is structuring the image directory. PyTorch and SageMaker requires the train, test, and validation to have its own directory with the folders of each class containing the images.

The raw images require a preprocessing for training. The preprocessing steps are as the following:

1. We use PyTorch transforms function to preprocess the image
2. The image is firstly resized to 256x256 pixels and center cropped to 224 x 224 pixels. The pre-trained models require a consistent image size and 224 x 224 pixels are common image size used for training the model. The subject of interest is most likely placed in the center of the image; hence we resize to 256x256 then center crop to 224 x224 pixels.
3. The images are converted to Tensor which is the data type for PyTorch model
4. The images are normalized using mean of [0.485, 0.456, 0.406] and the standard deviation of [0.229, 0.224, 0.225]. The reason for the mean and the standard deviation scale is because of the pre-trained models from ImageNet uses the same mean and standard deviation and have become a common practice in the industry. The purpose of the normalization is to accelerate the convergence of the loss function [6].

The reason for not applying image augmentation is that the collected images are considered to have diverse setup and environment that it would not require a special augmentation. If the performance is the not satisfied with the performance using the current approach, we will decide to augment the images in the future.

Implementation

In this section, we describe in detail the process for which the metrics, algorithms and techniques were implemented for each model.

The banana object detection model is developed through use of pretrained models from PyTorch. In our implementation, we evaluate different architectures or pretrained models such as VGG16 and ResNet101 using our collected dataset. The evaluation metric chosen for the evaluation is accuracy at top 5. The reasoning is that we expect the model to find challenges in predicting an object class amongst the 1000 classes. We thus choose to use accuracy at top 5 instead of conventional accuracy. Once we decide on the architecture, we use the model to also

validate on the images that are not banana using the accuracy at top 5 as well for our evaluation metric for the sake of ensuring that the model also predicts well for the false positives. As previously mentioned, the false positives is not considered as important as false negative. We thereby focus on the accuracy when having banana images as input.

For the implementation of banana ripeness classification, we utilize transfer-learning which uses the pretrained models to train on top of. SageMaker and PyTorch is used for training the model to accelerate the process. For our initial model, we build an architecture using ResNet101 as our basis and add two fully connected layers with a dropout for the model to learn specific to our application. After successful implementation, we explore different architecture as well as hyperparameter tuning to refine the model for final deployment. The metric used for evaluation of the model is accuracy.

When training the model, we use train set to train the model, and use validation set to determine how well the model has been trained. Finally, the test set is used as the final evaluation method as it is the dataset the model has never seen. This allows for generalization and robustness of the model.

For our final implementation, we will use SageMaker to create an end point of the developed model to preprocess the input image and output the final prediction along with a message.

Refinement

For banana detection model, no additional refinement is required other than selection of the architecture. For refinement of the banana detection model, we will select the best performing model amongst the evaluated models. The result of our models is shown as follows:

Table 1 - Banana Object Detection Model Performance

Pretrained ImageNet Models	Banana Detection Acc@5	Other Object Detection Acc@5
VGG16	0.921	0.996
ResNet101	0.978	0.9955

The banana ripeness classification model has many hyperparameters that could be tuned. In our refinement step, we utilize SageMaker Hyperparameter tuning to create a custom tuning job for selection of the best performing model. The hyperparameters that are trained are the following:
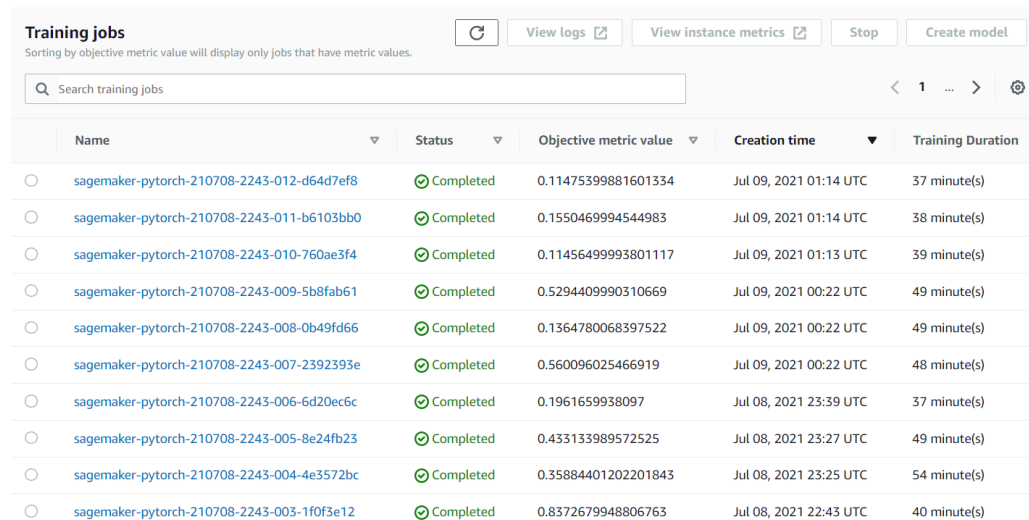
1. Train Parameters
   a. Learning Rate: The rate for updating the gradients (0.001, 0.1)
   b. Batch-Size: The amount of data to train before updating the gradients (8, 16, 32)
2. Neural Network
   a. Hidden Layer Neurons: The number of neurons in the fully connected layer (128, 256, 512)
   b. Model Transfer: The pretrained models to train from (ResNet101, ResNet152)

The performance of the initial model and the final model is given in the Table X.

Table 2 - Performance of the trained classification model

	Accuracy
Initial Model	0.87
Final Model	**0.93**
Benchmark Model	0.80

The initial model slightly outperforms the benchmark model, but with hyperparameter tuning, we were able to achieve 93% which is 13% increase from our baseline model.

Training jobs
Sorting by objective metric value will display only jobs that have metric values.

Name	Status	Objective metric value	Creation time	Training Duration
sagemaker-pytorch-210708-2243-012-d64d7ef8	⊘ Completed	0.11475399881601334	Jul 09, 2021 01:14 UTC	37 minute(s)
sagemaker-pytorch-210708-2243-011-b6103bb0	⊘ Completed	0.1550469994544983	Jul 09, 2021 01:14 UTC	38 minute(s)
sagemaker-pytorch-210708-2243-010-760ae3f4	⊘ Completed	0.11456499993801117	Jul 09, 2021 01:13 UTC	39 minute(s)
sagemaker-pytorch-210708-2243-009-5b8fab61	⊘ Completed	0.5294409990310669	Jul 09, 2021 00:22 UTC	49 minute(s)
sagemaker-pytorch-210708-2243-008-0b49fd66	⊘ Completed	0.1364780068397522	Jul 09, 2021 00:22 UTC	49 minute(s)
sagemaker-pytorch-210708-2243-007-2392393e	⊘ Completed	0.560096025466919	Jul 09, 2021 00:22 UTC	48 minute(s)
sagemaker-pytorch-210708-2243-006-6d20ec6c	⊘ Completed	0.1961659938097	Jul 08, 2021 23:39 UTC	37 minute(s)
sagemaker-pytorch-210708-2243-005-8e24fb23	⊘ Completed	0.433133989572525	Jul 08, 2021 23:27 UTC	49 minute(s)
sagemaker-pytorch-210708-2243-004-4e3572bc	⊘ Completed	0.35884401202201843	Jul 08, 2021 23:25 UTC	54 minute(s)
sagemaker-pytorch-210708-2243-003-1f0f3e12	⊘ Completed	0.8372679948806763	Jul 08, 2021 22:43 UTC	40 minute(s)

Figure 6 - SageMaker Hyperparameter Tuning job

Results

Model Evaluation and Validation

The final object detection model uses the ResNet101 as it yielded the best accuracy@5 for both banana detection and other object detection. To validate the data, we have tested on all our banana images which were 338 images, and on Kaggle Fruit 360 images which contained over 20,000 images. The final model achieved accuracy of 97.8% for detection of bananas and 99.6% for detection of objects other than bananas.

For the ripeness classification, the final model achieved 93% accuracy which shows 8% improvement from our benchmark model and 6% increase from our initial model. The parameters of the final model are given in the Table 3.

Table 3 - Final Model Parameters

Description	Value
Learning Rate	0.02942
Batch Size	8
Number of Neurons in FC Layer	256
Number of Fully FC Layer	2
Activation Function of FC Layer	ReLU
Pretrained ImageNet Model	ResNET152
Epochs	15

The evaluation of the model always used the test dataset that the model has never seen. This ensures the robustness and generalization of the model.

Justification

The final model has accuracy of 97.8% for detecting the banana and ripeness classification accuracy of 93% on the test set which shows a great improvement from our benchmark model. With visual inspection, we can determine whether the object is banana and its ripeness based on its color. The developed model also provides high accuracy which reflects with the thesis. Also, providing the ripeness classification with 90%+ accuracy for those that have color blindness would be useful in their daily lives.

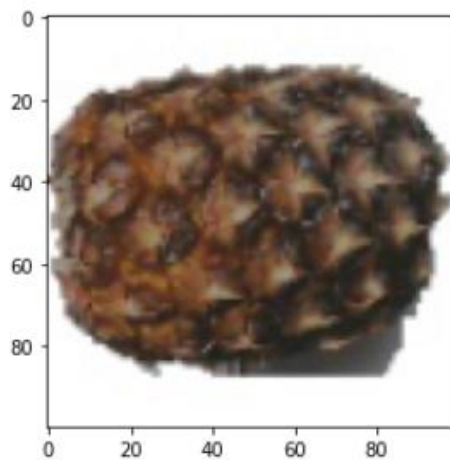The examples of the output are shown in the figures below.

Figure 7 - Example output with pineapple as an input

This is not banana! This looks like strawberry 🫤🫤
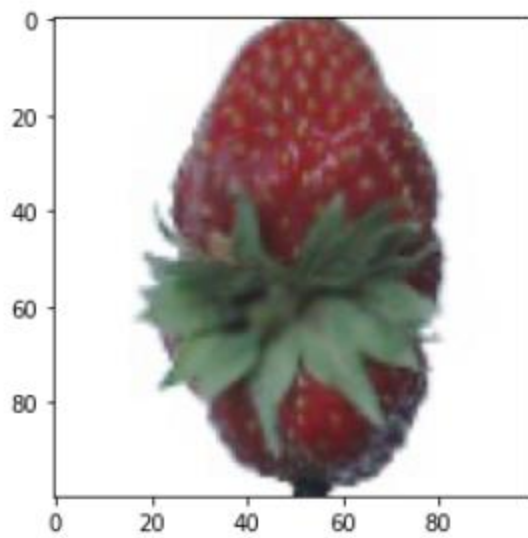Please upload banana for classification

Figure 8 - Example output with strawberry as an input
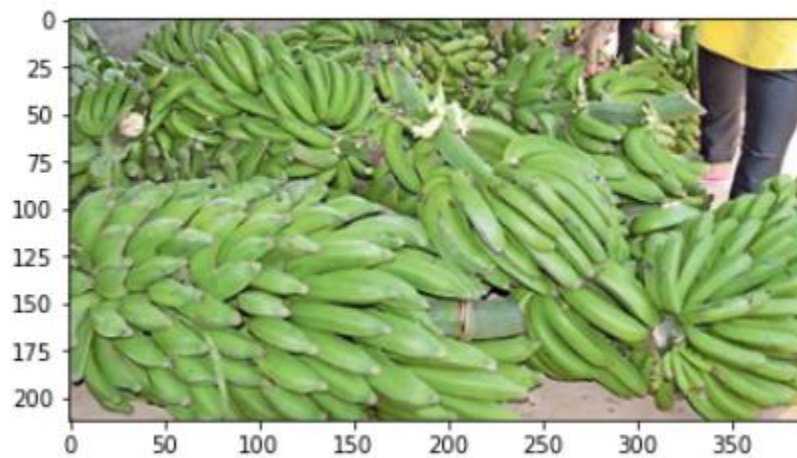
This banana is Unripe!! 🤮🤮

Figure 9 - Example output with unripe banana as an input
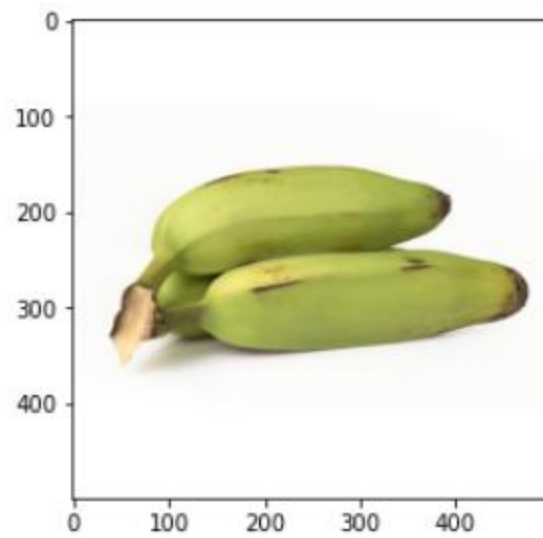
This banana is Unripe!! 🤮🤮

Figure 10 - Example output with unripe banana as an input
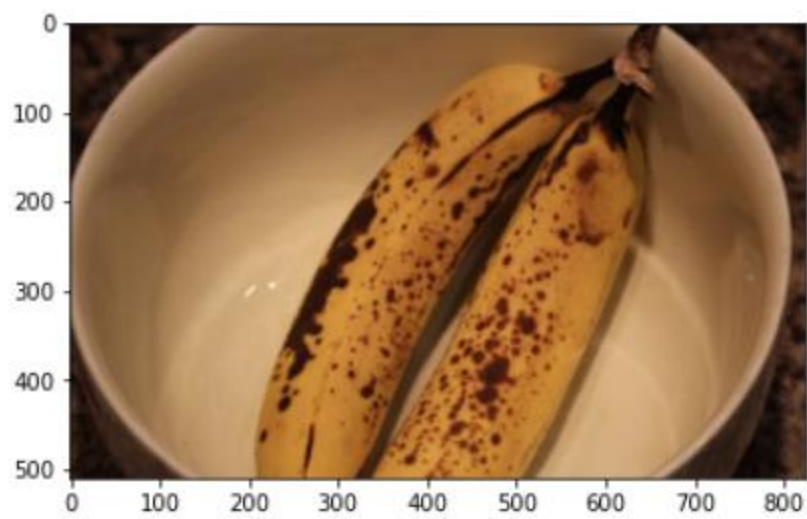
This banana is Overripe!! 🤮🤮

Figure 11- Example output with overripe banana as an input
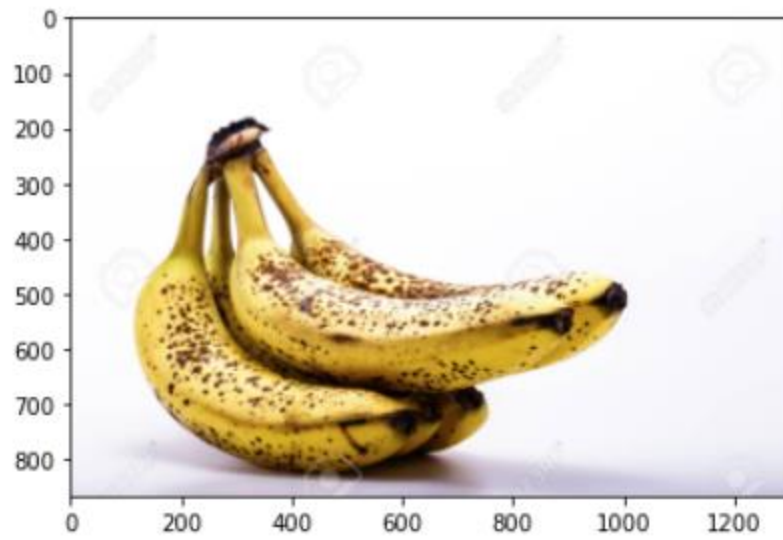
This banana is Overripe!! 🤮🤮

Figure 12 - Figure 11- Example output with overripe banana as an input

This banana is Ripe!! 🚀🚀

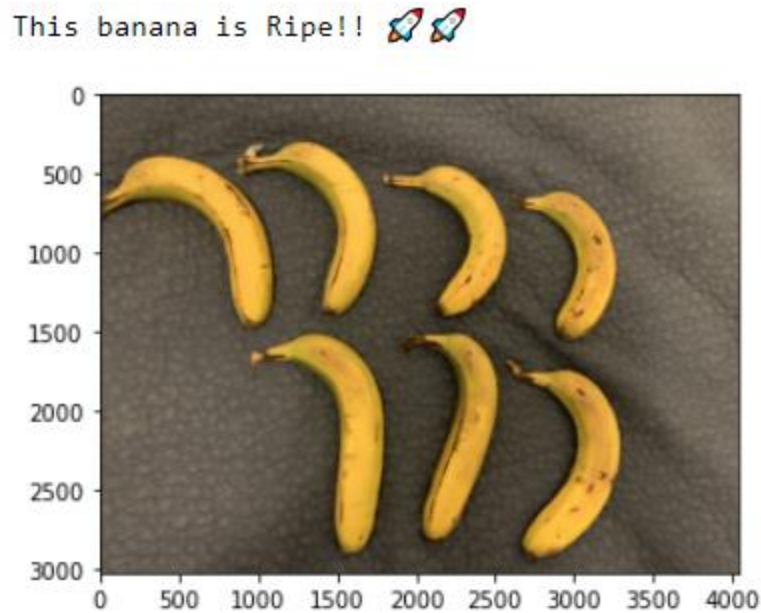*Figure 13 - Figure 11- Example output with ripe banana as an input*

Figure 14 - Figure 11- Example output with ripe banana as an input

References

[1] *Colour Blindness.* Colour Blind Awareness. (n.d.). https://www.colourblindawareness.org/colour-blindness/. Accessed July 2021

[2] Hameed, Khurram & Chai, Douglas & Rassau, Alexander. (2018). A comprehensive review of fruit and vegetable classification techniques. Image and Vision Computing. 80. 10.1016/j.imavis.2018.09.016. Accessed July 2021

[3] R. Thakur, G. Suryawanshi, H. Patel and J. Sangoi, "An Innovative Approach For Fruit Ripeness Classification," *2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS)*, 2020, pp. 550-554, doi: 10.1109/ICICCS48265.2020.9121045. Accessed July 2021

[4] V.Gupta, A.Murzova "Keras Tutorial: Using pre-trained Imagenet Models" https://learnopencv.com/keras-tutorial-using-pre-trained-imagenet-models/, Accessed July 2021

[5] Defining a Neural Network in PyTorch. https://pytorch.org/tutorials/recipes/recipes/defining_a_neural_network.html, Accessed July 2021

[6] Torchvision Models. https://pytorch.org/vision/stable/models.html, Accessed July 2021

[7] Kaggle Fruits Dataset. https://www.kaggle.com/moltean/fruits, Accessed July 2021