School of Computer Science and Engineering
Digital Assignment – I
Winter – SEMESTER 2022 - 23

Course Code: MCSE603L
Course-Title: – Deep Learning
Programme Name: M.Tech
Branch: CSE with Specialization in AI & ML
Class Number (s): VL2022230506176
SLOT: E2
Faculty: 18812 - BIJI C L - SCOPE

Name: Nidhi Singh
Reg. No: 22MAI0015

Flowers image classification model

Abstract:

This paper presents a flower classification model developed using Google Teachable Machine. Flower classification plays a vital role in various domains such as agriculture, ecology, and botanical research. With the advancements in machine learning techniques, creating accurate and efficient flower classification models has become achievable. The model was trained using a dataset consisting of various flower species with diverse visual characteristics. The training process involved uploading the dataset to Google Teachable Machine and leveraging transfer learning techniques on a pre-trained model. The model achieves accurate classification of various flower types based on their images. The study highlights the importance of dataset quality and diversity in achieving optimal performance. Strategies such as data augmentation, fine-tuning, and parameter adjustments are employed to improve the model's performance. The results demonstrate the potential of Google Teachable Machine as an accessible platform for training machine learning models. The findings contribute to the field of flower classification and provide insights for researchers and practitioners interested in similar models. This study showcases the democratization of machine learning and its applications in flower classification.

Keywords : Flowers, classification, Google Teachable machine, training, transfer learning

Introduction :-

Flower classification plays a significant role in various domains, including agriculture, ecology, and botanical research. Accurate identification and classification of different flower species can provide valuable insights into plant diversity, ecosystem health, and conservation efforts. With the advancements in machine learning techniques, creating robust flower classification models has become increasingly feasible. Google Teachable Machine offers a user-friendly and accessible platform that allows users to train machine learning models without extensive coding knowledge. The objective of this study is to develop a flower classification model using Google Teachable Machine and explore its potential for accurate and efficient flower identification. By leveraging transfer learning techniques on pre-trained models, we aim to build a model capable of recognizing and classifying diverse flower species based on their visual features. The availability of a high-quality and diverse dataset is crucial for training effective flower classification models. This study utilizes a carefully curated dataset comprising images of various flower species, ensuring representation across different botanical families and visual characteristics. The dataset is uploaded to Google Teachable Machine, which facilitates the training process and model development through a simple and intuitive interface. The findings of this study contribute to the growing body of research on flower classification using accessible machine learning platforms. By showcasing the capabilities of Google Teachable Machine, this research demonstrates the democratization of machine learning and its potential for expanding applications in flower classification and beyond. The insights and methodologies presented in this study can serve as a valuable resource for researchers, practitioners, and enthusiasts interested in developing flower classification models or exploring similar machine learning applications.

Method :-

To create a flower image classification model using Google Teachable Machine, follow these steps:

- Go to the Google Teachable Machine website.
- Click on "Get Started" to begin creating your project.
- Choose the "Image Project" option.
- Prepare your training data by collecting or finding images of different types of flowers. It's important to have a diverse set of images representing each category you want to classify. For example, if you want to classify roses, sunflowers, and tulips, gather images of each flower type.
- Click on the "Upload" button to add your training data. You can either drag and drop your files or choose files from your computer. Organize your images into folders for each category to make it easier to upload.
- Once your training data is uploaded, you'll see a preview of the examples. Make sure the images are correctly assigned to their respective categories.
- Click on the "Train Model" button to start training your network. The training process may take a few minutes, depending on the size of your dataset.

- After training, you can test your model using the webcam or by uploading test images. The model will attempt to classify the images based on what it has learned.
- If you're satisfied with the performance of your model, you can export it by clicking on the "Export Model" button. Choose the appropriate format based on your needs (TensorFlow, TensorFlow.js, or MobileNet).
- Use the exported model in your own applications or projects. You'll receive instructions on how to integrate the model into different programming environments.

Dataset:-

The data set used in this analysis contains 16 different kinds of images of flowers. I collected more than twenty different kinds of flowers available on the VIT campus and as well as other places and created my dataset. My dataset contains Adenium obesum, Autumn zephyrlily, Bougainvillea, daisy, dandelion, Ixoroideae, Plumeria alba, Red ginger, rose, Singapore graveyard, sunflower , tecoma capensis, Tecoma stans, tulip, West Indian Lantana and yellow West Indian Lantana flowers.

All images were taken with a resolution of 224x224 pixels color images. The flower images were captured by my mobile phone and other cameras from different angles and situations during the day and night. I come across a variety of challenges when acquiring images like light, sunshine, darkness, the camera capturing artifacts, shadows, pose variations and lighting changes. Same categories of images were taken at various times and days, which makes the scenario more realistic.

Serial no.	Flower name	Quantity
1	Adenium obesum	7
2	Autumn zephyrlily	7
3	Bougainvillea	13
4	Daisy	61
5	Dandelion	109
6	Ixoroideae	2
7	Plumeria alba	5
8	Red ginger	6
9	Rose	25
10	Singapore graveyard	3
11	Sunflower	42
12	tecoma capensis	5
13	Tecoma stans	6
14	Tulip	34
15	West Indian Lantana	10
16	yellow West Indian Lantana	6

Choose your project type: -

In Teachable Machine, you have three project types to choose from: Image Project, Audio Project, or Pose Project. For flower classification, select the Image Project. chose the image project because I wanted to make a model for image classification.

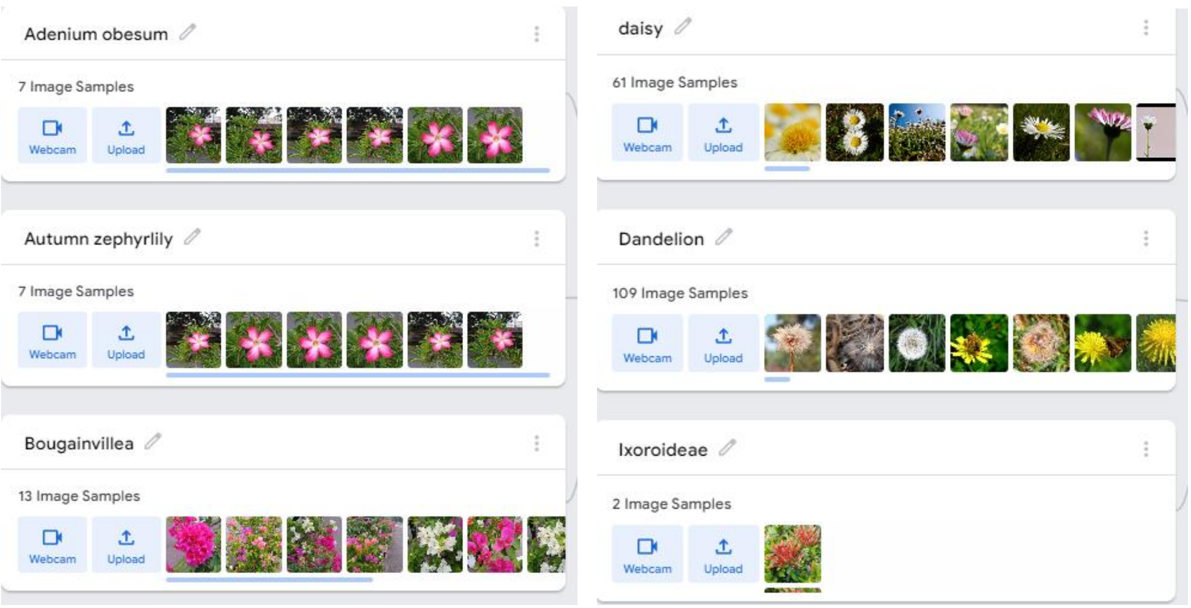
Set up your classes:

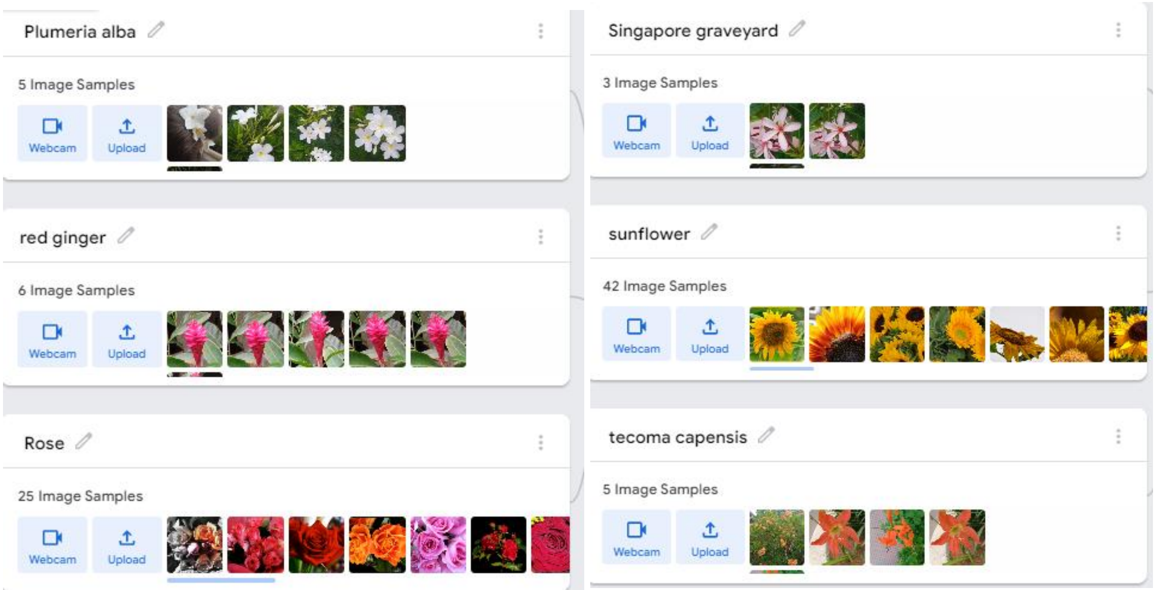
Click on the "Add a Class" button to create classes corresponding to the different types of flowers in your dataset. I creates 16 different classes with 16 different flowers name labels. The names of the classes are Adenium obesum, Autumn zephyrlily, Bougainvillea, daisy, dandelion, Ixoroideae, Plumeria alba, Red ginger, rose, Singapore graveyard, sunflower , tecoma capensis, Tecoma stans, tulip, West Indian Lantana and yellow West Indian Lantana.

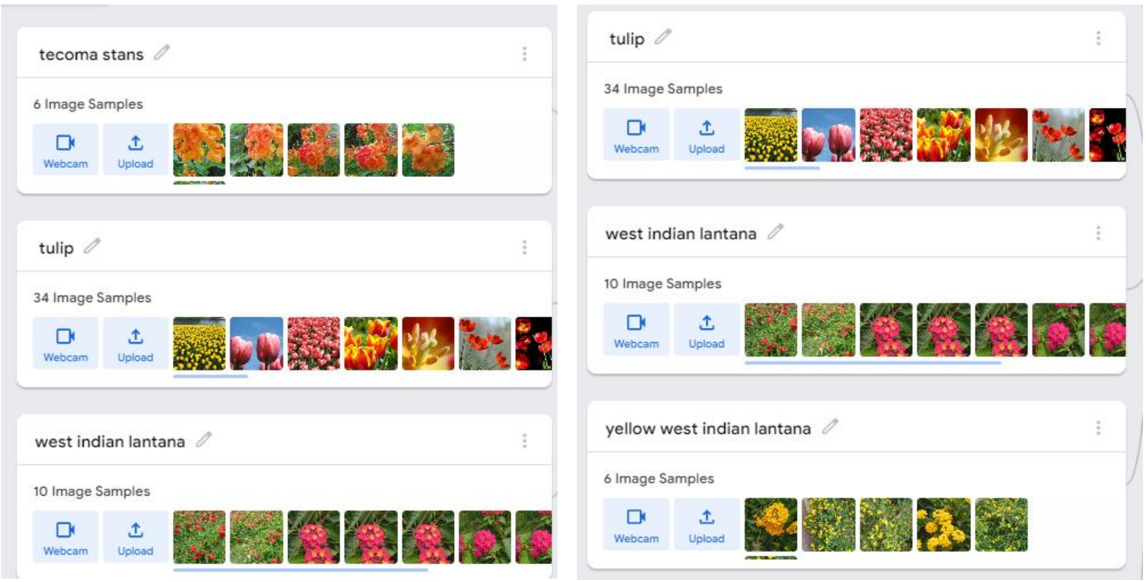
Model Training

Further, the dataset containing 16 different kinds of images of flowers is splitted into sixteen training classes. The class Adenium obesum, Autumn zephyrlily, Bougainvillea, daisy, dandelion, Ixoroideae, Plumeria alba, Red ginger, rose, Singapore graveyard, sunflower , tecoma capensis, Tecoma stans, tulip, West Indian Lantana and yellow West Indian Lantana are contains 7, 7,13,764,29,2,5,6,25,3,42,5,6,34,10 and 6 different images as training dataset , respectively. I trained my model with all there 16 different types of images.

After collecting and labelling my data, click on the "Train Model" button. Teachable Machine will start training a machine learning model using your labelled images. The training process may take a few moments or longer, depending on the size of my dataset and the complexity of my model. Snapshot of the training procedure is given bellow :





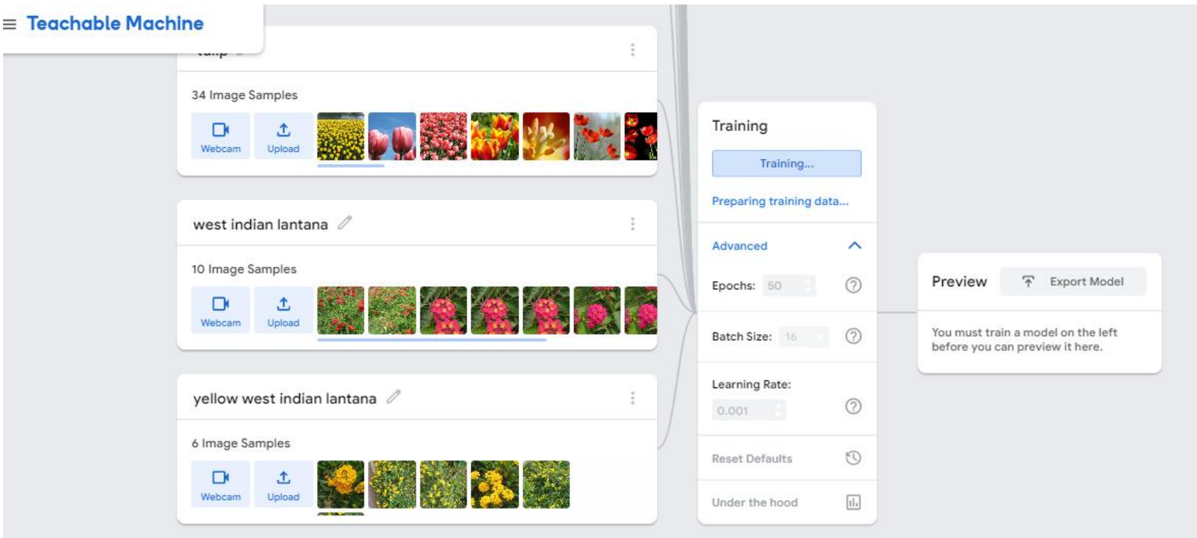


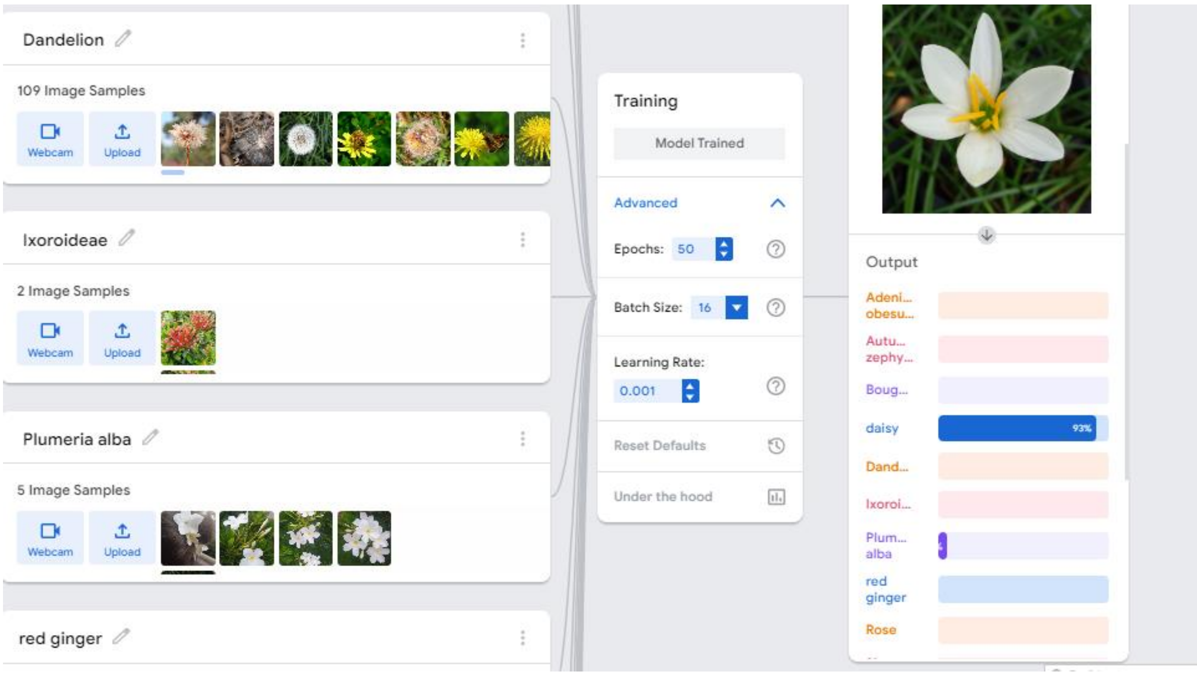
To test a flower classification model in Google Teachable Machine, you can follow these steps:

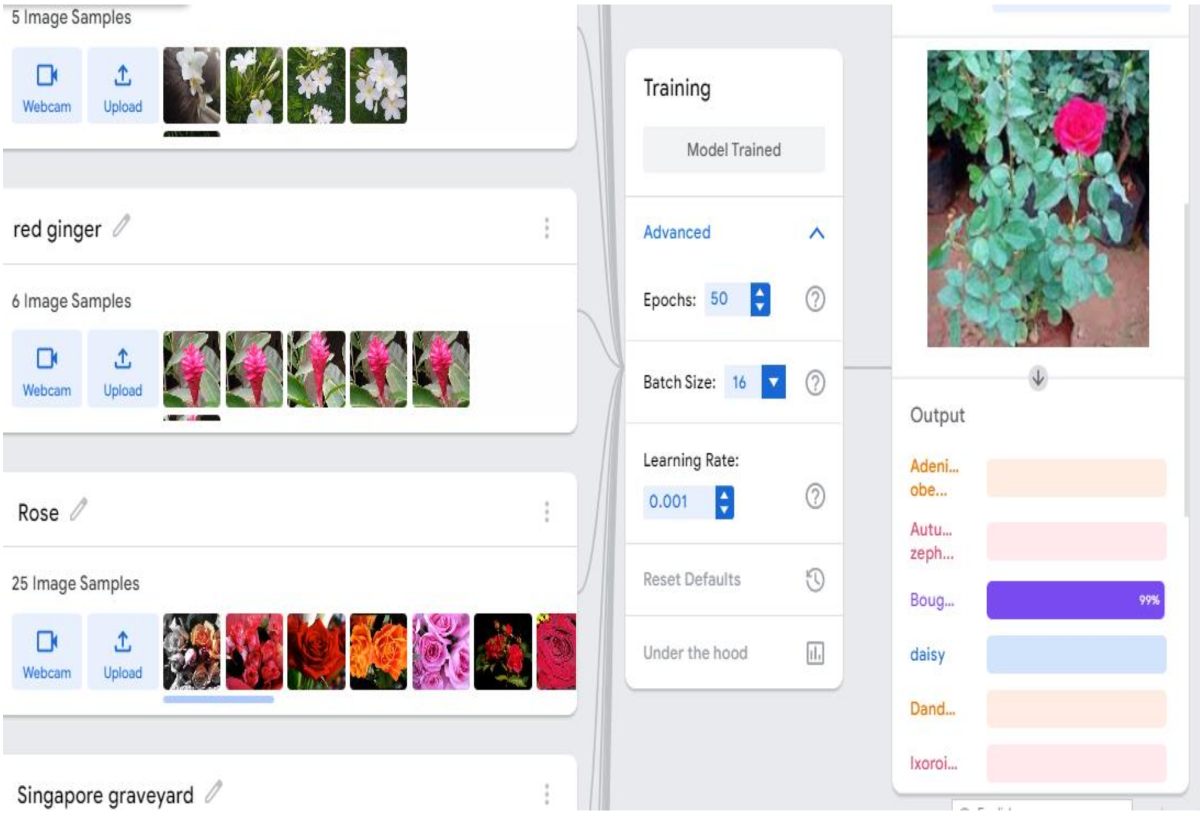
Once the training is complete, I can test my model by clicking on the "Test" tab. I upload a flower image from my computer. The model will classify the image based on the flower classes it has been trained on.

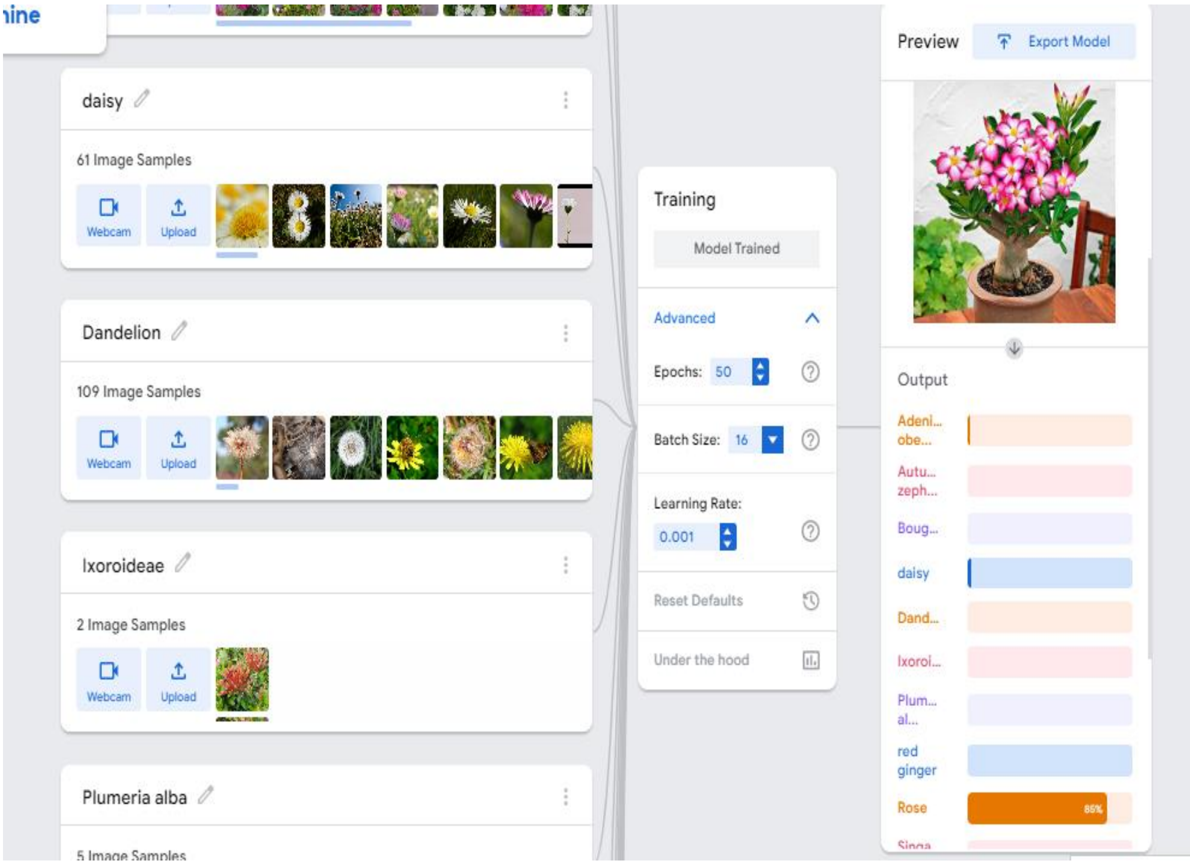
Classification Accuracy:-

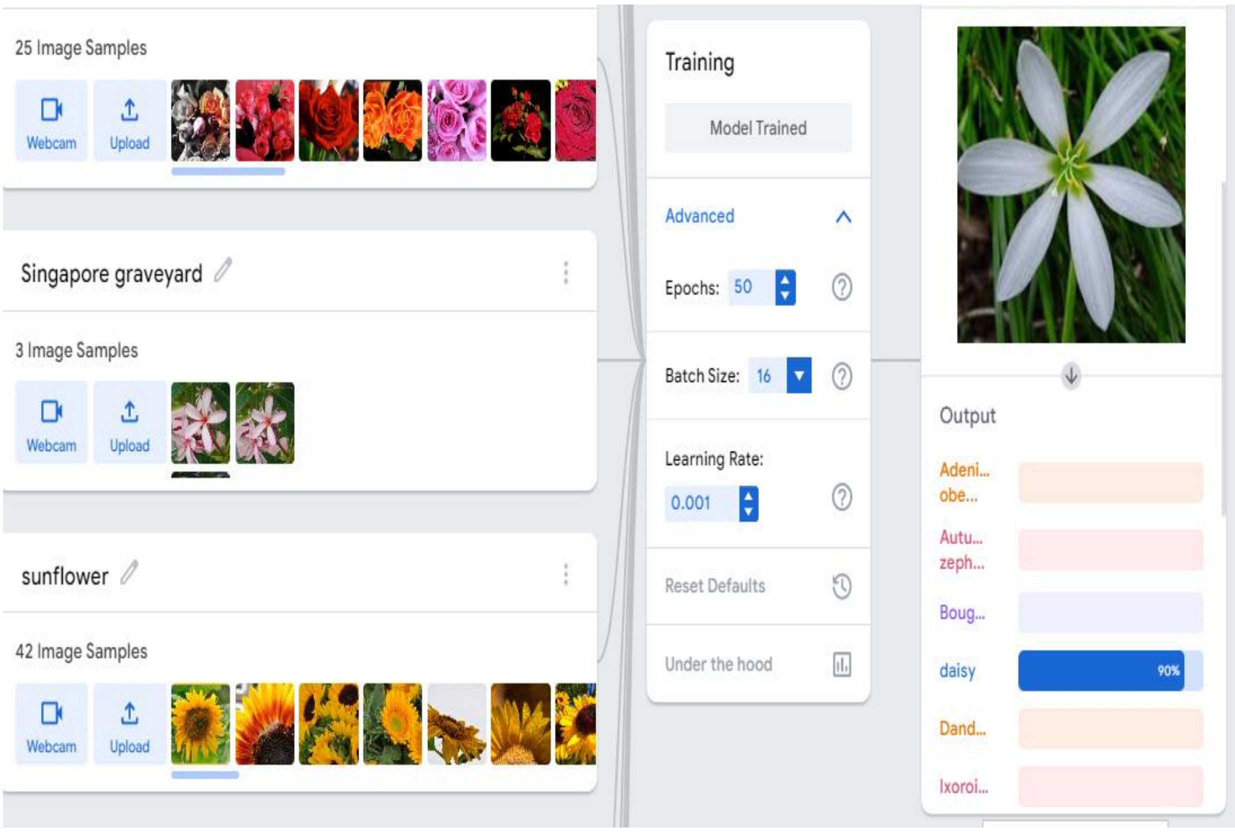
After testing my model, I reviewed the classifications to determine how accurate my model is. Flower classification model gives 100 % accuracy for daisy flower but perform poorly with other flowers. The model out perform with daisy flower only. The classification model gives 29% accuracy with Autumn zephyr lily flowers. The classification model most of the classify Autumn zephyrlily flower as daisy flower class which is not gave good classification result. The classification model perform average with rose flower classification. The model's performance is not satisfactory, i may need to iterate and refine my dataset or experiment with different training settings. I can add more images, improve the quality of my labels, or adjust the training duration to improve the model's accuracy. The bellow picture shows the classification result for our model.

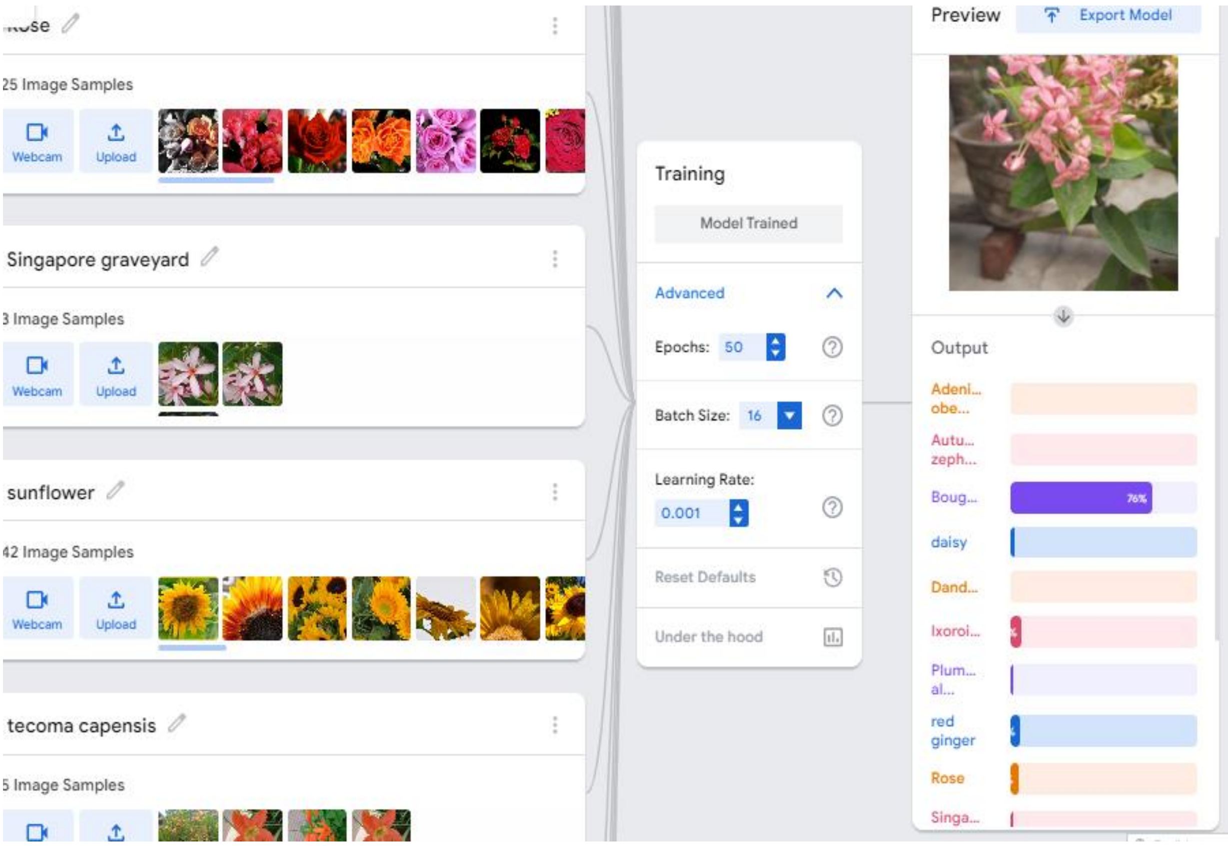


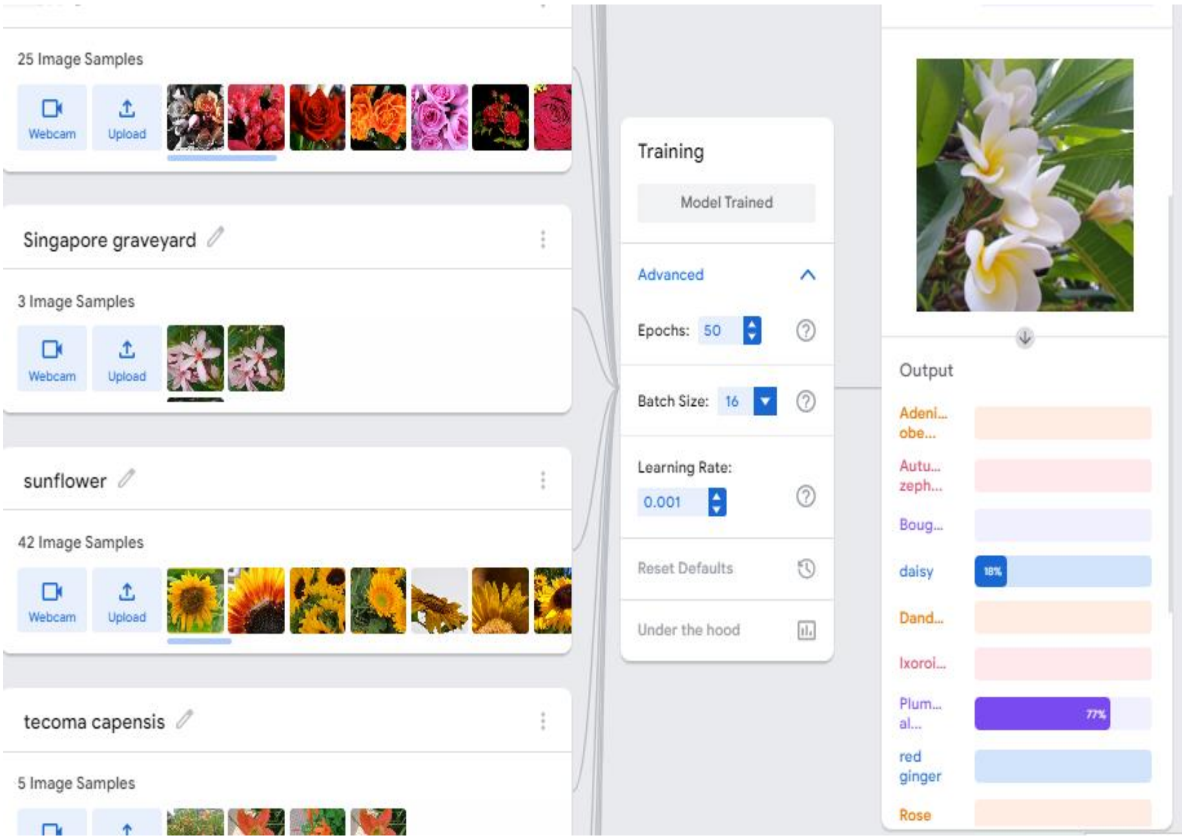


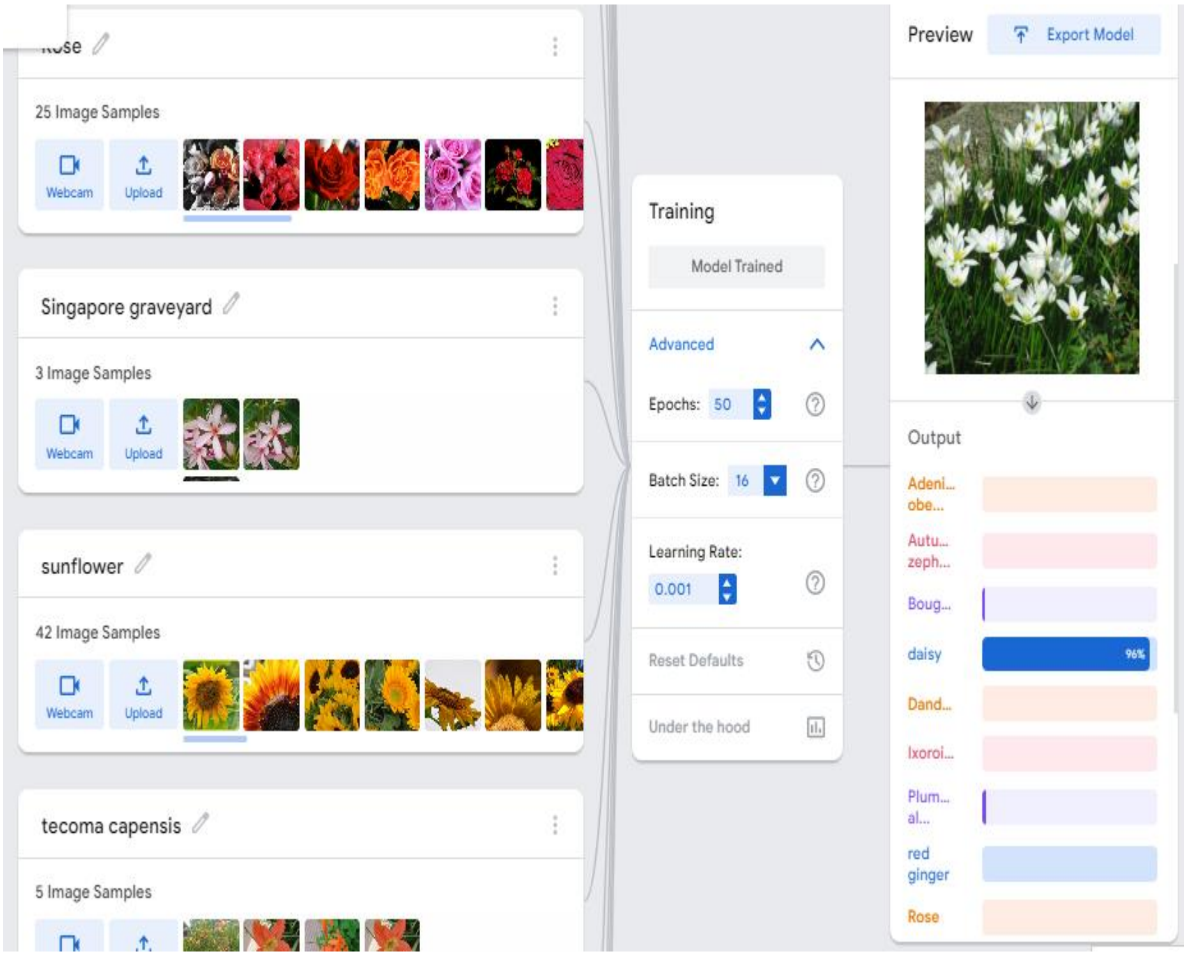

5 Image Samples
Webcam
Upload
red ginger
6 Image Samples
Webcam
Upload
Rose
25 Image Samples
Webcam
Upload
Singapore graveyard
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Output
Adeni... obe...
Autu... zeph...
Boug...
99%
daisy
Dand...
Ixoroi...

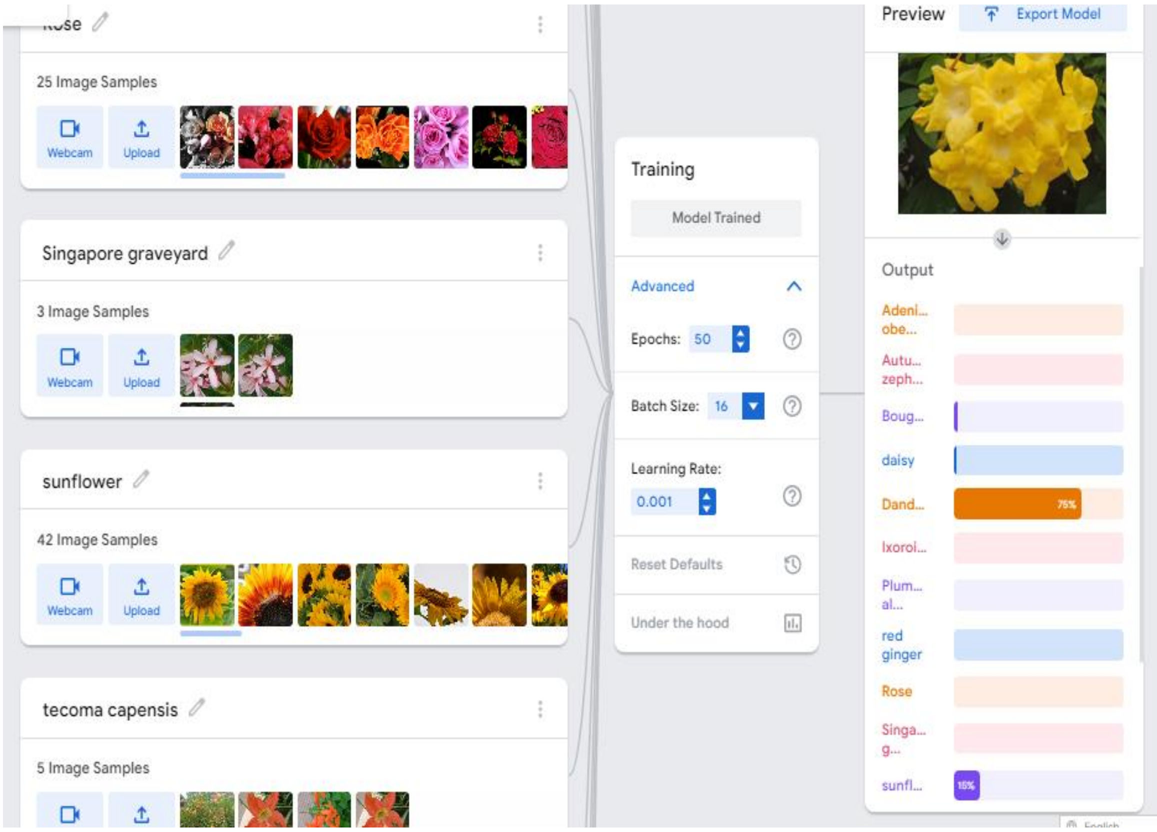

daisy
61 Image Samples
Webcam
Upload
Dandelion
109 Image Samples
Webcam
Upload
Ixoroideae
2 Image Samples
Webcam
Upload
Plumeria alba
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Preview
Export Model
Output
Adeni... obe...
Autu... zeph...
Boug...
daisy
Dand...
Ixoroi...
Plum... al...
red ginger
Rose
86%

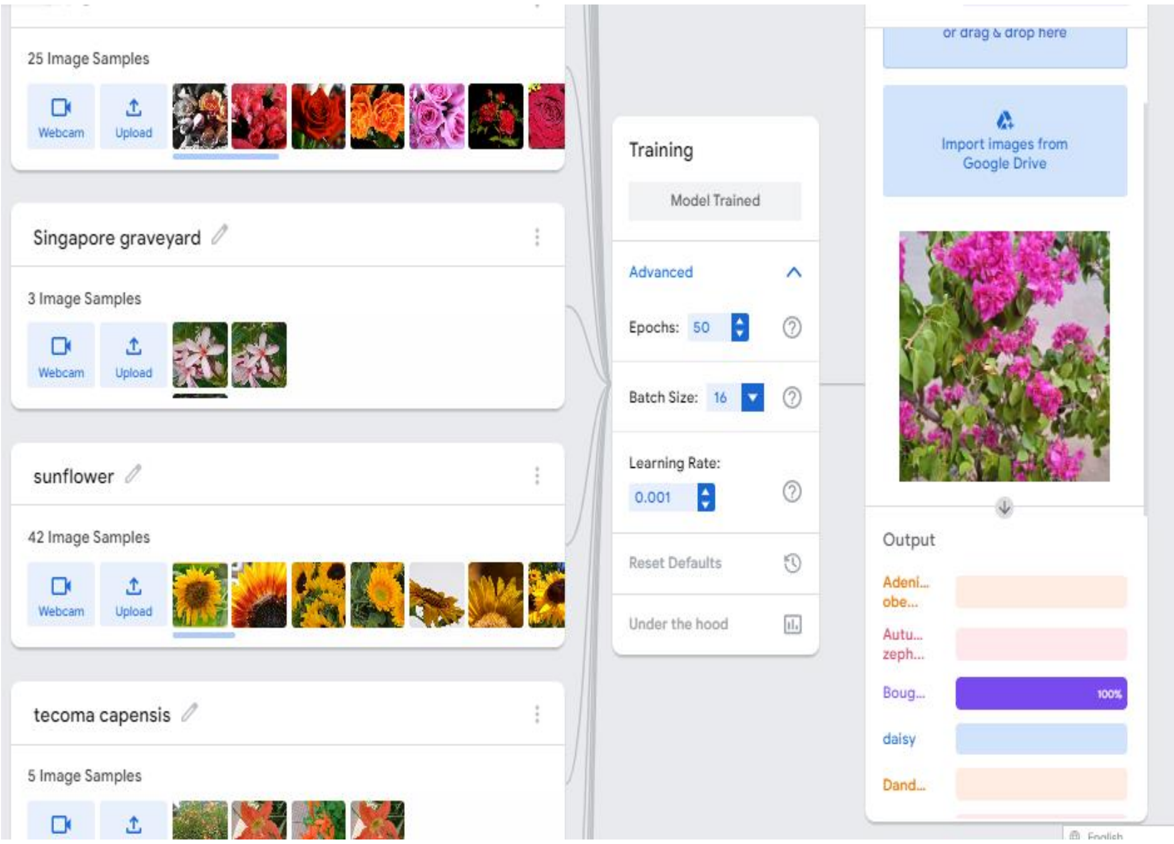

25 Image Samples
Webcam
Upload
Singapore graveyard
3 Image Samples
sunflower
42 Image Samples
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Output
Adeni... obe...
Autu... zeph...
Boug...
daisy
90%
Dand...
Ixoroi...


25 Image Samples
Webcam
Upload
Singapore graveyard
3 Image Samples
sunflower
42 Image Samples
tecoma capensis
5 Image Samples
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Preview
Export Model
Output
Adeni... obe...
Autu... zeph...
Boug...
76%
daisy
Dand...
Ixoroi...
Plum... al...
red ginger
Rose
Singa...


25 Image Samples
Webcam
Upload
Singapore graveyard
3 Image Samples
Webcam
Upload
sunflower
42 Image Samples
Webcam
Upload
tecoma capensis
5 Image Samples
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Output
Adeni... obe...
Autu... zeph...
Boug...
daisy
18%
Dand...
Ixoroi...
Plum... al...
77%
red ginger
Rose

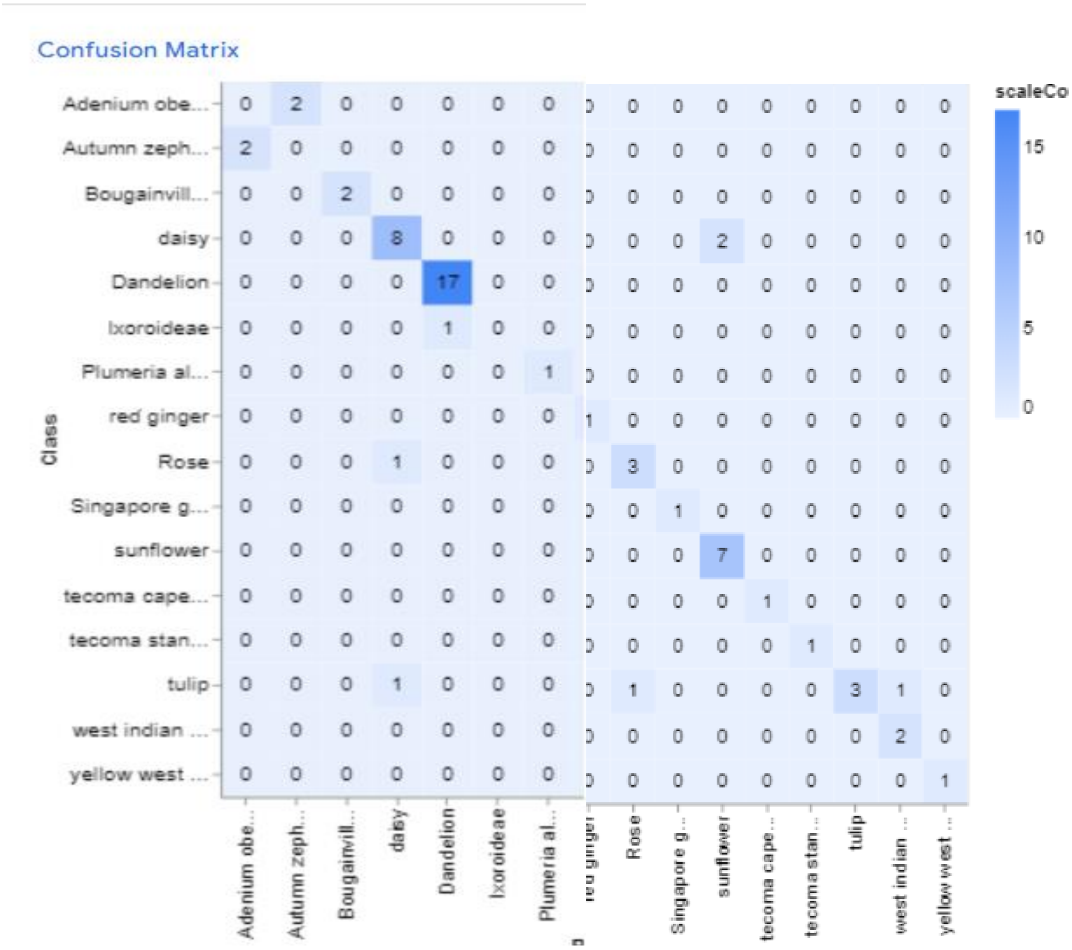

25 Image Samples
Webcam
Upload
Singapore graveyard
3 Image Samples
Webcam
Upload
sunflower
42 Image Samples
Webcam
Upload
tecoma capensis
5 Image Samples
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Preview
Export Model
Output
Adeni... obe...
Autu... zeph...
Boug...
daisy
96%
Dand...
Ixoroi...
Plum... al...
red ginger
Rose


Preview
Export Model
25 Image Samples
Webcam
Upload
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Singapore graveyard
3 Image Samples
sunflower
42 Image Samples
tecoma capensis
5 Image Samples
Output
Adeni... obe...
Autu... zeph...
Boug...
daisy
Dand...
75%
Ixoroi...
Plum... al...
red ginger
Rose
Singa... g...
sunfl...
15%


25 Image Samples
Webcam
Upload
or drag & drop here
Import images from Google Drive
Training
Model Trained
Advanced
Epochs: 50
Batch Size: 16
Learning Rate:
0.001
Reset Defaults
Under the hood
Singapore graveyard
3 Image Samples
sunflower
42 Image Samples
tecoma capensis
5 Image Samples
Output
Adeni... obe...
Autu... zeph...
Boug...
100%
daisy
Dand...

Accuracy per class

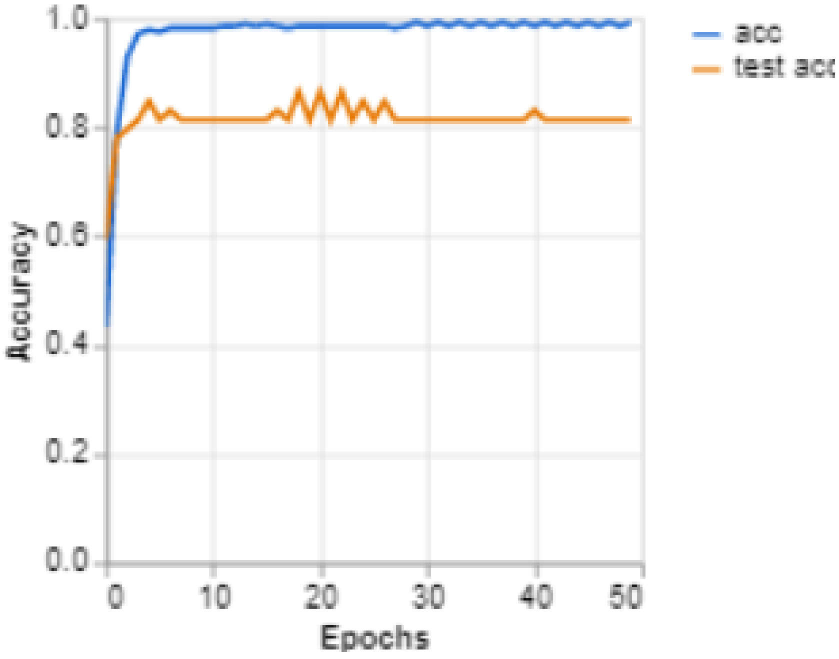
CLASS	ACCURACY	# SAMPLES
Adenium obesum	0.00	2
Autumn zephyrlily	0.00	2
Bougainvillea	1.00	2
daisy	0.80	10
Dandelion	1.00	17
Ixoroideae	0.00	1
Plumeria alba	1.00	1
red ginger	1.00	1
Rose	0.75	4
Singapore graveya...	1.00	1
sunflower	1.00	7
tecoma capensis	1.00	1
tecoma stans	1.00	1
tulip	0.50	6
west indian lantan...	1.00	2
yellow west indian...	1.00	1

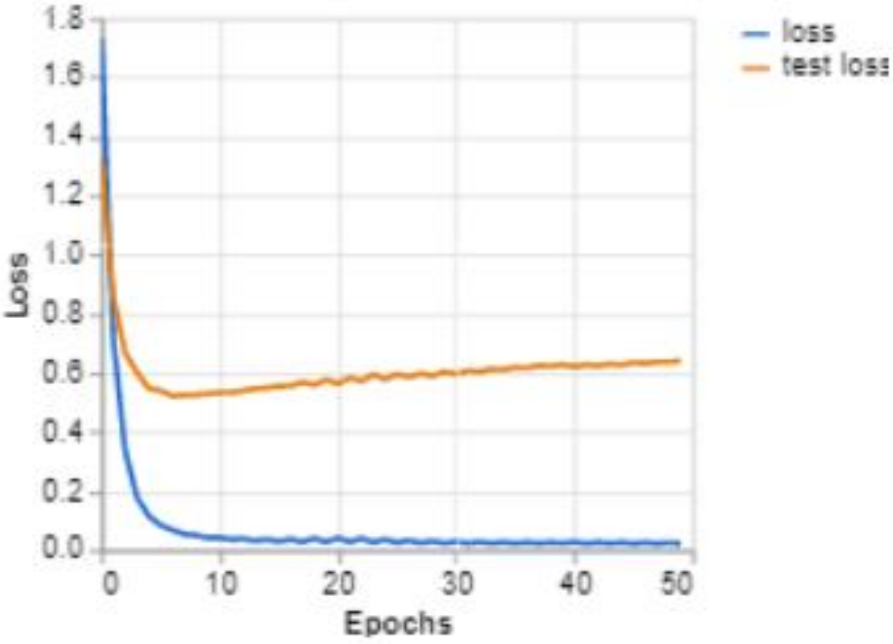


Result and discussion:-

To evaluate the performance of the model, various metrics such as accuracy, precision, recall, and F1 score were used. The results showed promising performance, with an overall satisfactory classification accuracy. However, it was observed that the model performed exceptionally well for certain flower classes, while struggling with others. This discrepancy highlighted the importance of dataset quality and diversity in achieving optimal performance. The developed flower classification model demonstrates the potential of Google Teachable Machine as a powerful tool for training machine learning models without extensive coding knowledge. The case study provides insights into the challenges and strategies involved in building effective flower classification models, emphasizing the significance of dataset quality, augmentation, and fine-tuning.

Accuracy per epoch





Export the model:

Once you are satisfied with your model's performance, you can export it for use in your own applications. Teachable Machine provides options to export the model as a TensorFlow project or as a model that can be integrated with other platforms or frameworks.

Import the necessary libraries: load_model from keras.models, Image and ImageOps from PIL, and numpy as np.

```
from keras.models import load_model
 # TensorFlow is required for Keras to work
from PIL import Image, ImageOps
  # Install pillow instead of PIL
import numpy as np
```

Set the option to disable scientific notation for clarity when printing numpy arrays.

```
# Disable scientific notation for clarity
np.set_printoptions(suppress=True)
```

- Load the pre-trained model using load_model function from Keras. The model file should be saved as "keras_Model.h5".
- The argument compile=False is used to prevent the model from being compiled, as you only need it for prediction.

```
# Load the model
model = load_model("keras_Model.h5", compile=False)
```

Load the class names or labels for the flower classes. The labels are assumed to be stored in a text file named "labels.txt".

```
# Load the labels
class_names = open("labels.txt", "r").readlines()
```

Create an array data with the shape (1, 224, 224, 3) to match the input shape expected by the model. The first dimension of 1 represents the number of images in the array.

```
# Create the array of the right shape to feed into the keras model
# The 'length' or number of images you can put into the array is
# determined by the first position in the shape tuple, in this case 1
data = np.ndarray(shape=(1, 224, 224, 3), dtype=np.float32)
```

Replace <IMAGE_PATH> with the actual path to the image you want to classify.

Open the image you want to classify using Image.open and convert it to RGB color space using convert("RGB").

```
# Replace this with the path to your image
image = Image.open("<IMAGE_PATH>").convert("RGB")
```

Resize the image to be at least 224x224 pixels using ImageOps.fit. The image is resampled using Lanczos resampling for better quality.

```
# Resizing the image to be at least 224x224 and then cropping from the center
size = (224, 224)
image = ImageOps.fit(image, size, Image.Resampling.LANCZOS)
```

Convert the image to a numpy array using np.asarray.

```
# Turn the image into a numpy array
image_array = np.asarray(image)
```

Normalize the image array by dividing it by 127.5 and subtracting 1, resulting in values ranging from -1 to 1. This step aligns with the preprocessing applied during model training.

```
# Normalize the image
normalized_image_array = (image_array.astype(np.float32) / 127.5) - 1
```

Load the normalized image array into the data array for prediction. Since you have only one image, it is placed at index 0 of the data array.

```
# Load the image into the array
data[0] = normalized_image_array
```

- Make a prediction using the loaded model by calling model.predict and passing the data array.
- Find the index of the predicted class with the highest probability using np.argmax on the prediction array.
- Retrieve

```
# Predict the model
prediction = model.predict(data)
index = np.argmax(prediction)
```

```
class_name = class_names[index]
confidence_score = prediction[0][index]
```

This part of the code is responsible for printing the predicted class and its corresponding confidence score.

```
# Print prediction and confidence score
print("Class:", class_name[2:], end="")
print("Confidence Score:", confidence_score)
```

The line print("Class:", class_name[2:], end="") prints the predicted class. It uses string slicing (class_name[2:]) to remove any leading characters, such as newline characters, from the class name. The end="" argument ensures that the next print statement will continue on the same line.

The line print("Confidence Score:", confidence_score) prints the confidence score of the predicted class.

These print statements provide the final output of the code, displaying the predicted class and the associated confidence score.

Conclusion :-

I created a model for flower classification on Google teachable machine. The model performs well with some flowers but perform poorly, with few flowers. The model gives 100% accuracy with daisy flower classification. The overall classification result is satisfactory. The accuracy of flower classification model depends on the quality and diversity of your dataset, as well as the training duration and other training settings. Experimenting with different approaches and iterations can help improve the model's performance. The performance is low for certain flowers, it could indicate that the dataset lacks diversity or has an insufficient number of examples for those specific classes. Consider gathering more images of the poorly performing flowers to ensure a balanced representation across all classes. Including different angles, lighting conditions, and variations within each class can also help improve the model's ability to generalize. The overall performance of the model is good for some flower classification. Overall, this paper contributes to the growing body of research on flower classification using accessible machine learning platforms. It highlights the potential of Google Teachable Machine in democratizing machine learning and fostering innovation in various domains that rely on flower classification. The findings and methodologies presented in this study can serve as a valuable resource for researchers and practitioners interested in developing similar models or exploring flower classification applications.