Subject : Artificial Intelligence Lab

Project Title : Detection and recognition of fruits and vegetables using Deep Learning

Team Members :

1. Swapnil Chhatre - 33213
2. Krishiv Dakwala - 33216
3. Amey Godse - 33226
4. Akash Kulkarni - 33241

Implementation

Loading the dataset

```
!pip install kaggle
```

```
from google.colab import files
files.upload()
```

```
# Creating new folder
!mkdir -p ~/.kaggle

# Copy kaggle.json into kaggle folder
!cp kaggle.json ~/.kaggle

# Changing permissions so only I can read-write in the json file
'chmod 600 /root/.kaggle/kaggle.json'
```

```
    cp: cannot stat 'kaggle.json': No such file or directory
    'chmod 600 /root/.kaggle/kaggle.json'
```

```
'chmod 600 /root/.kaggle/kaggle.json'
!echo '{"username":"swapnilchhatre","key":"16805da928ed51e2d658878236f78ee4"}' > /
!kaggle datasets download -d kritikseth/fruit-and-vegetable-image-recognition
```

```
    Warning: Your Kaggle API key is readable by other users on this system! To fi
    Downloading fruit-and-vegetable-image-recognition.zip to /content
```

```
100% 1.98G/1.98G [00:20<00:00, 110MB/s]
100% 1.98G/1.98G [00:20<00:00, 104MB/s]
```

```
!ls
```

```
fruit-and-vegetable-image-recognition.zip  model_1_weights.h5
model_1.h5                                 sample_data
```

```
!unzip fruit-and-vegetable-image-recognition.zip
```

Creating Arrays for Training Model

```
import numpy as np
import pandas as pd
import os
import cv2
import matplotlib.pyplot as plt
```

```
train_path = "/content/train"
valid_path = "/content/validation"
test_path = "/content/test"

# Displaying total number of classes
categories = os.listdir(train_path)
categories.sort()

"Categories Count :", len(categories) # There are 36 categories (classes) of fruit
```

```
('Categories Count :', 36)
```

```
categories
```

```
['apple',
 'banana',
 'beetroot',
 'bell pepper',
 'cabbage',
 'capsicum',
 'carrot',
 'cauliflower',
 'chilli pepper',
 'corn',
 'cucumber',
 'eggplant',
 'garlic',
 'ginger',
      'grapes',
      'jalepeno',
      'kiwi',
      'lemon',
      'lettuce',
      'mango',
      'onion',
      'orange',
      'paprika',
      'pear',
      'peas',
      'pineapple',
      'pomegranate',
      'potato',
      'raddish',
      'soy beans',
      'spinach',
      'sweetcorn',
      'sweetpotato',
      'tomato',
      'turnip',
      'watermelon']


# Creating a 'train' array of all images and corresponding lables from all classes
# with images of size 100x100
from keras.preprocessing.image import ImageDataGenerator
from PIL import Image

datagen = ImageDataGenerator(
        rotation_range = 40,
        zoom_range = 0.2)

train = []

IMG_SIZE=100

for category in categories:
    folder = os.path.join(train_path, category)
    label = categories.index(category)

    for file in os.listdir(folder):
        file = os.path.join(folder,file)
        img = cv2.imread(file)

        try:
            img_arr = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            train.append([img_arr, label])

        except:
            pass
```

```
validation = []

for category in categories:
    folder = os.path.join(valid_path,category)
    label = categories.index(category)

    for file in os.listdir(folder):
        file = os.path.join(folder,file)
        img = cv2.imread(file)
        try:
            img_arr = cv2.resize(img,(IMG_SIZE,IMG_SIZE))
            validation.append([img_arr,label])
        except:
            pass

"Validation Image Count :", len(validation)
```

```
     ('Validation Image Count :', 351)
```

```
test = []

for category in categories:
    folder = os.path.join(test_path, category)
    label = categories.index(category)

    for file in os.listdir(folder):
        file = os.path.join(folder, file)
        img = cv2.imread(file)
        try:
            img_arr = cv2.resize(img, (IMG_SIZE, IMG_SIZE))
            test.append([img_arr, label])
        except:
            pass

"Testing Image Count :", len(test)
```

```
     ('Testing Image Count :', 359)
```

```
# Manually splitting the created arrays for the model
```

```
# Lists

x_train = []  # Image features
y_train = []  # Label of the image

x_validation = [] # Image features
y_validation = [] # Label of the image
```

```
x_test = [] # Image features
y_test = [] # Label of the image

for features , label  in train:
    x_train.append(features)
    y_train.append(label)
y_train = pd.get_dummies(y_train)

for features , label  in validation:
    x_validation.append(features)
    y_validation.append(label)
y_validation = pd.get_dummies(y_validation)

for features , label  in test:
    x_test.append(features)
    y_test.append(label)
y_test = pd.get_dummies(y_test)


# Convert the lists to numpy arrays
x_train = np.array(x_train)/255 # Normalizing features to range 0-1
y_train = np.array(y_train)

x_validation = np.array(x_validation)/255 # Normalizing features to range 0-1
y_validation = np.array(y_validation)

x_test = np.array(x_test)/255 # Normalizing features to range 0-1
y_test = np.array(y_test)

# Displaying the lengths of the lists
len(x_train), len(y_train), len(x_validation), len(y_validation), len(x_test), len
```

```
(3114, 3114, 351, 351, 359, 359)
```

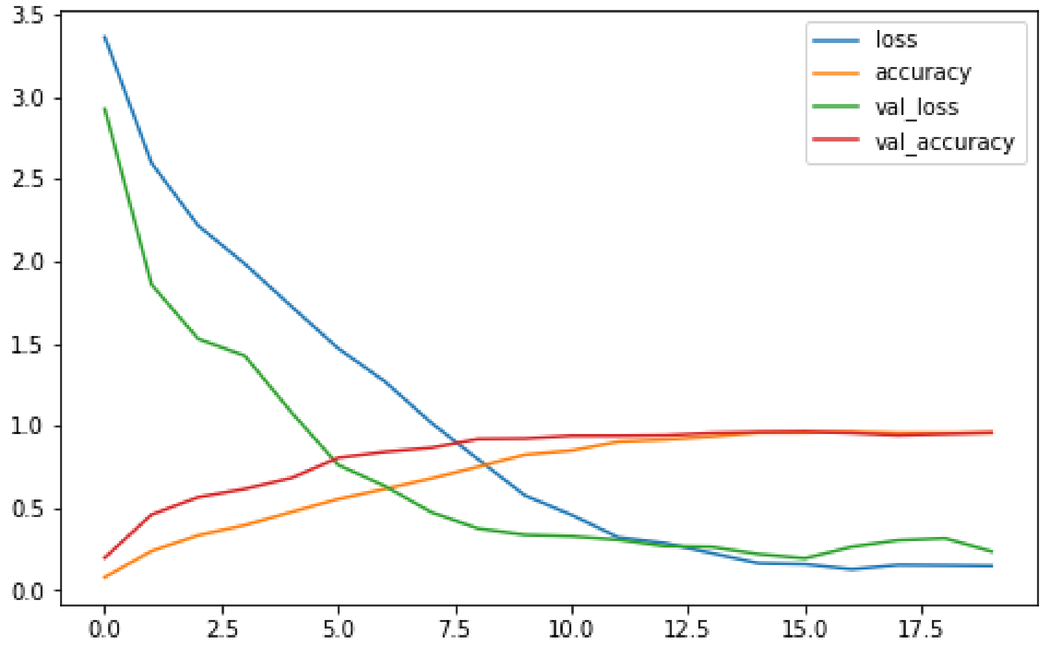
Creating Model

```
import tensorflow as tf
import keras
from keras.models import Sequential
from keras.layers import Dense, Conv2D, MaxPooling2D, Flatten, Activation, Dropout
from keras.utils.vis_utils import plot_model


model = Sequential()

model.add(Conv2D(32,(3,3),activation="relu"))
model.add(MaxPooling2D(2,2))
model.add(Conv2D(64,(3,3),activation="relu"))
model.add(MaxPooling2D(2, 2))

model.add(Conv2D(128,(3,3),activation="relu"))
model.add(MaxPooling2D(2,2))
model.add(Dropout(0.2))
model.add(Flatten())

# Output layer
model.add(Dense(128,input_shape = x_train.shape[1:],activation="relu"))
model.add(Dense(36,activation="softmax"))
input_shape = (None, 100, 100, 3)
model.build(input_shape)
plot_model(model, to_file='model_plot.png', show_shapes=True, show_layer_names=True


model.compile(optimizer="adam",loss = "categorical_crossentropy",metrics=["accurac
history = model.fit(x_train, y_train, epochs=17, validation_data=(x_test, y_test))
```

```
Epoch 1/17
98/98 [==============================] - 56s 556ms/step - loss: 3.2528 - accu
Epoch 2/17
98/98 [==============================] - 54s 551ms/step - loss: 2.6106 - accu
Epoch 3/17
98/98 [==============================] - 56s 568ms/step - loss: 2.2549 - accu
Epoch 4/17
98/98 [==============================] - 54s 553ms/step - loss: 2.0452 - accu
Epoch 5/17
98/98 [==============================] - 54s 552ms/step - loss: 1.8466 - accu
Epoch 6/17
98/98 [==============================] - 54s 551ms/step - loss: 1.7189 - accu
Epoch 7/17
98/98 [==============================] - 54s 552ms/step - loss: 1.4774 - accu
Epoch 8/17
98/98 [==============================] - 54s 552ms/step - loss: 1.3152 - accu
Epoch 9/17
98/98 [==============================] - 54s 553ms/step - loss: 1.1250 - accu
Epoch 10/17
98/98 [==============================] - 54s 554ms/step - loss: 0.9696 - accu
Epoch 11/17
98/98 [==============================] - 54s 555ms/step - loss: 0.7673 - accu
Epoch 12/17
98/98 [==============================] - 54s 554ms/step - loss: 0.6748 - accu
Epoch 13/17
98/98 [==============================] - 54s 553ms/step - loss: 0.5154 - accu
Epoch 14/17
98/98 [==============================] - 56s 570ms/step - loss: 0.3874 - accu
Epoch 15/17
98/98 [==============================] - 54s 555ms/step - loss: 0.3328 - accu
Epoch 16/17
98/98 [==============================] - 54s 554ms/step - loss: 0.3228 - accu
Epoch 17/17
98/98 [==============================] - 54s 554ms/step - loss: 0.2168 - accu
```

```
model.save("model_1.h5")
model.save_weights("model_1_weights.h5")
```

```
from matplotlib import pyplot as plt
pd.DataFrame(history.history).plot(figsize=(8,5))
plt.show()
```



```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_augmentation = ImageDataGenerator(
    rescale = 1./255,
    rotation_range = 40,
    zoom_range = 0.2,
    horzintal_flip = True
)

train_generator = train_augmentation.flow_from_directory(
    directory="/content/train",
    target_size = (100,100),
)
```

Creating a Web Application using Streamlit

```
!pip install streamlit
```

```
!pip install pyngrok
```

```
%%writefile app.py
import streamlit as st
import cv2
import tensorflow
from tensorflow import keras
from PIL import Image
from keras.preprocessing.image import load_img, img_to_array
import numpy as np
from keras.models import load_model

model = load_model("/content/model_1.h5")
categories = ['apple', 'banana', 'beetroot', 'bell pepper', 'cabbage', 'capsicum',
              'chilli pepper', 'corn', 'cucumber', 'eggplant', 'garlic', 'ginger',
              'lemon', 'lettuce', 'mango', 'onion', 'orange', 'paprika', 'pear', 'p
              'potato', 'raddish', 'soy beans', 'spinach', 'sweetcorn', 'sweetpotat


def processed_img(img_path):
    img = load_img(img_path, target_size=(100, 100, 3))
    img = img_to_array(img)

    answer = model.predict(img)
    y_class = answer.argmax(axis=-1)

    print(y_class)

    y = " ".join(str(x) for x in y_class)
    y = int(y)

    res = categories[y]
    print(res)

    return res.capitalize()


def run():
    st.title("Fruits and Vegetable Recognition")

    img_file = st.file_uploader("Choose an Image", type = ["jpg", "png"])

    if img_file is not None:
        img = Image.open(img_file).resize((100, 100))
        st.image(img, use_column_width=False)
        save_image_path = './uploaded_images/' + img_file.name

        with open(save_image_path, "wb") as f:
            f.write(img_file.getbuffer())

        if img_file is not None:
            result = processed_img(save_image_path)
            print(result)
```

```
run()
```

```
     Overwriting app.py
```

```
!ls
```

```
     app.py  model_1.h5  model_1_weights.h5  sample_data
```

```
!ngrok authtoken 28O8OY3n53cbFP8Huk8RiQpMXRM_3qDLwEbbaPkvgVnmDTfJ1
```

```
     Authtoken saved to configuration file: /root/.ngrok2/ngrok.yml
```

```
from pyngrok import ngrok
```

```
!streamlit run --server.port 80 app.py&>/dev/null&
```

```
!pgrep streamlit
```

```
     227
     449
```

```
publ_url = ngrok.connect(port='80')
```

```
publ_url
```

```
     <NgrokTunnel: "http://9e63-34-125-198-45.ngrok.io" -> "http://localhost:80">
```