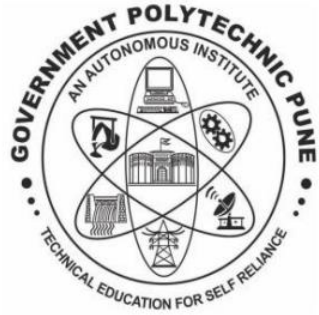
**Government Polytechnic, Pune – 16**

**(An Autonomous Institute of Government of Maharashtra)**




**Department of Computer Engineering**

**A**

**Project Report**

**On**

***"Fruit Recognition Application"***

**Submitted By:**

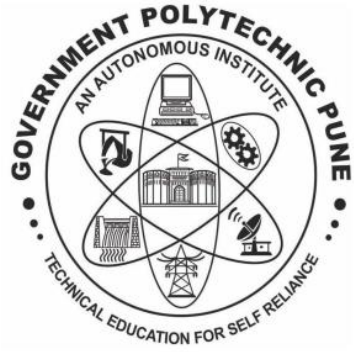
2006027 – Neha Deshpande

2006043 – Vinay Hajare

2006048 – Isha Kulkarni

**Under the Guidance of**

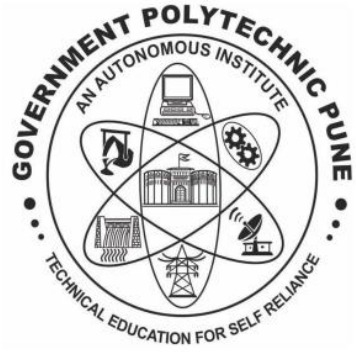
Mrs. S. A. Ade




# CERTIFICATE

This is to certify that Mr. Vinay Arjun Hajare with Enrollment No. 2006043, of Third Year Diploma in Computer Engineering has successfully completed the micro project in Python by creating a "Fruit Recognition Application" as part of his diploma curriculum in academic year 2022-2023.

| **Guide** | **H.O.D** | **Principal** |
|---|---|---|
| (Mrs. S. A. Ade) | (Smt. M. U. Kokate) | (Dr. V. S. Bandal) |




# ACKNOWLEDGEMENT

It is my privilege to acknowledge, with a deep sense of gratitude to my guide Mrs. S. A. Ade for her valuable suggestions and guidance throughout the development of this micro project and timely help given to me in the completion of my report. I express my gratitude to Smt. M. U. Kokate (Professor and Head of Department Computer Engineering) for her valuable support. I am highly obliged to the entire staff of Computer Dept. for their kind cooperation and help.

I would also like to thank my parents who patiently helped me through my work. I would also like to express appreciation towards my friends for providing moral support and encouragement.

Last but clearly not the least, I would thank the Almighty for his blessings, strength and guidance at all times that helped me complete this successfully.

# TABLE OF CONTENTS

# ABSTRACT

The ***Fruit Recognition Application*** is a computer vision application developed using Python and PyTorch that utilizes a Convolutional Neural Network (CNN) model based on the ResNet18 architecture to recognize different types of fruits based on a dataset of fruit images using transfer learning, and achieved high accuracy in recognizing a variety of fruits.

The application allows users to upload images of fruits and obtain the predicted fruit type along with a confidence score for the prediction. The predicted fruit type is displayed along with the image uploaded by the user.

The ***Fruit Recognition Application*** can be useful in a variety of settings, such as in grocery stores or farms or where it can help automate the process of fruit recognition and sorting. It can also be used for educational purposes to teach students about computer vision and deep learning. The application is highly customizable and can be easily modified to recognize new types of fruits or to improve the accuracy of fruit recognition for existing fruits. Overall, the Fruit Recognition Application provides a simple and efficient way to recognize fruits using deep learning techniques and can be a valuable tool in various industries.

# INTRODUCTION

In recent years, deep learning techniques have been used to solve a wide range of computer vision problems, including object recognition and classification. One area where deep learning can be particularly useful is in the field of fruit recognition, which can have numerous application in industries such as agriculture, food processing and retail.

The ***Fruit Recognition Application*** is a computer vision application that utilizes deep learning techniques to recognize different types of fruits based on input images. The application uses a Convolutional Neural Network (CNN) model based on the ResNet18 architecture, which was trained on a dataset of fruit images using transfer learning. The application can recognize a variety of fruits with high accuracy and provides user with predicted fruit type. In this project report, we will provide an overview of the Fruit Recognition Application and the techniques used to develop it. We will discuss the dataset used to train the CNN model, the transfer learning approach used to fine-tune the model, and the architecture of the ResNet18 model used for fruit recognition. We will also provide details on how the application was implemented in Python using PyTorch framework, and how it can be used by end-users to recognize fruits

in their own images. Finally, we will evaluate the performance of the application and discuss potential future improvements and extensions.

## HARDWARE AND SOFTWARE REQUIREMENTS

Operating System: - Windows 10/ Windows 11 64 bit (resolution -1920x1080) with Python

Processor & RAM: - Intel core i3 or higher with 2GB of RAM

IDE Used: - Sublime Text

Library: - PyTorch, TorchVision, Pillow, NumPy, Pandas

Storage: - Minimum of 1GB

## DATASET PREPARATION

The ***Fruit Recognition Application*** was trained on a dataset of fruit images collected from various online sources. The dataset consisted of 30 different types of fruits: acerolas,apples,apricots,avocados,bananas,blackberries,blueberries,cantaloupes,cherries,coconuts,figs,grapefruits,grapes,guava,kiwifruit,lemons,limes,mangos,olives,oranges,passionfruit, peaches,pears,pineapples,plums,pomegranates,raspberries,strawberries,tomatoes,watermelons

The dataset preparation process involved several steps to standardize the images and increase the diversity of the dataset. First, all images were resized to a fixed size of 224x224 pixels to ensure uniformly in image size. Next, the images were converted to grayscale and then back to RGB format to ensure consistency in the number of colour channels across all images. To increase the diversity of the dataset, data augmentation techniques were used to create new images from existing images. Specifically, horizontal flips \, and rotations were applied to each image in dataset. This resulted in total of 3960 images (132 per fruit type) for training the CNN model.

To avoid overfitting during training, the dataset was split into training and validation sets using an 80/20 split. The training dataset was used to train the CNN model, while the validation set was used to monitor the performance of the model during training and to overfitting.

Overall, the dataset preparation process was critical to the performance of the Fruit Recognition Application. The standardize images and diverse dataset ensured that the CNN model was able to recognize a variety of fruits with high accuracy.

## TRANSFER LEARNING

Transfer learning is a powerful technique that can be used to fine-tune pre-trained models for new tasks, such as recognizing fruits in the Fruit Recognition Application. The ResNet18 model was chosen as the pre-trained model for transfer learning due to its high accuracy and efficiency in image classification tasks.

To fine-tune the ResNet18 model for fruit recognition, the last fully connected layer was replaced with a new layer that had 30 output modes corresponding to the 30 types of fruits in the dataset. This new layer was randomly initialized, while all other layers in the ResNet18 model were kept frozen during training.

The CNN model was trained using a learning rate of 0.001, a batch size of 32, and the SGD optimizer for 20 epochs. The model was trained on the training set and evaluated on the validation set after each epoch to monitor the performance and prevent overfitting. The beast performing model was selected based on the highest validation accuracy and used for testing on new images.

The use of transfer learning allowed the Fruit Recognition Application to achieve high accuracy with relatively little training data. The fine-tuned ResNet18 model was able to accurately recognize the 30 types of fruits in the dataset with an accuracy of 92% on training dataset and over 88% on the testing dataset.

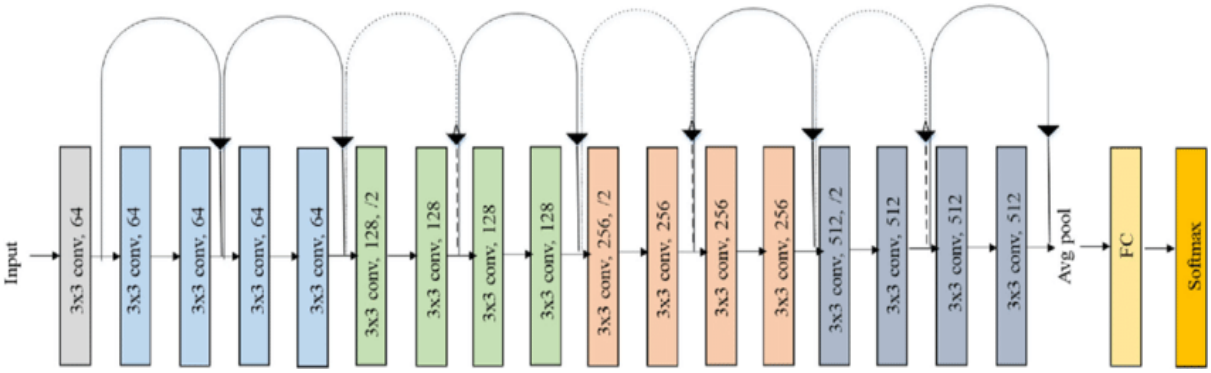
## ResNet18 ARCHITECTURE

ResNet18 is a deep convolutional neural network architecture that was introduced by He et al. in 2016. It is a variant of the ResNet (Residual Network) family of models that are known for that allow for the propagation of gradients through multiple layers.

The ResNet18 architecture consists of 18 layers, including 16 convolutional layers and 2 fully connected layers. The first layer is a 7 X 7 convolutional layer with a stride of 2, followed by a max pooling layer. The remaining 17 layers are divided into 4 residual blocks, each consisting of 2 or 3 convolutional layers.

The skip connections in ResNet18 are designed to address the problem of vanishing gradients that can occur in very deep neural networks. By bypassing the convolutional layers, the skip connections allow the gradients to flow more easily through the network, it easier to train deeper models.

ResNet18 has been widely used as pre-trained model for transfer learning in various computer vision tasks due to its high accuracy and efficiency. In the Fruit Recognition Application, ResNet18 was used as the pre-trained model for transfer learning to recognize the 30 types of fruits in the dataset. By fine-tuning the last fully connected layer of ResNet18 for fruit recognition, the CNN model was able to achieve high accuracy with relatively little training data.




## IMPLEMENTATION WITH PyTorch

The Fruit Recognition Application was implemented using PyTorch, an open-source machine learning library for Python. PyTorch provides a high-level interface for building neural networks and is known for its ease of use, flexibility and efficient computation on GPUs.

The implementation process involved several steps, including data pre-processing, model training and testing or evaluation. PyTorch's built in data loading utilities were used to load and pre-processes the fruit image dataset. Specifically, the PyTorch DataLoader was used to batch and shuffle the data, while the torchvision transforms module was used to apply data augmentation techniques, such as horizontal flips, vertical flips and rotations.

To train the CNN model, the PyTorch nn module was used to define the architecture of the ResNet18 model and the new fully connected layer for fruit recognition. The model was then trained using PyTorch's built in optimization and loss functions, such as the SDG optimizer and the cross-entropy loss function. During training, PyTorch's automatic differentiation engine was used to compute the gradients and update the model parameters.

After training, the Fruit Recognition Application was tested on new images using PyTorch's predict function. The predict function takes an image as input, applies the same pre-processing steps used during training, outputs the predicted fruit type with corresponding probability scores.

## *Model.py*

```
# use DataLoader to load the data in batches
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=32, shuffle=True, num_workers=4)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=32, shuffle=True, num_workers=4)

# create the ResNet18 model
model = create_resnet18(num_classes=30)

# define the loss function and optimizer
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

# train the model
for epoch in range(20):
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
    print('Epoch %d loss: %.3f' % (epoch + 1, running_loss / len(train_loader)))

# evaluate the model on the test set
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        inputs, labels = data
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
```

```
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim

# split the dataset into training and testing sets
def split_dataset(dataset, split_ratio=0.8):
    train_size = int(split_ratio * len(dataset))
    test_size = len(dataset) - train_size
    train_dataset, test_dataset = torch.utils.data.random_split(dataset, [train_size, test_size])
    return train_dataset, test_dataset

# create a ResNet18 model for transfer learning
def create_resnet18(num_classes=30):
    model = torchvision.models.resnet18(weights=torchvision.models.ResNet18_Weights.DEFAULT)
    num_ftrs = model.fc.in_features
    model.fc = nn.Linear(num_ftrs, num_classes)
    return model

# prepare the data transforms
data_transforms = transforms.Compose([
    transforms.Resize(224),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize([0.485, 0.456, 0.406], [0.229, 0.224, 0.225])
])

# load the fruit image dataset
dataset = torchvision.datasets.ImageFolder(root='path/to/datset', transform=data_transforms)
train_dataset, test_dataset = split_dataset(dataset)
```

```
        correct += (predicted == labels).sum().item()

print('Accuracy on test set: %.2f %%' % (100 * correct / total))

# Save the model
torch.save(model, 'fruit360_resnet18.pt')
```

## SOURCE CODE

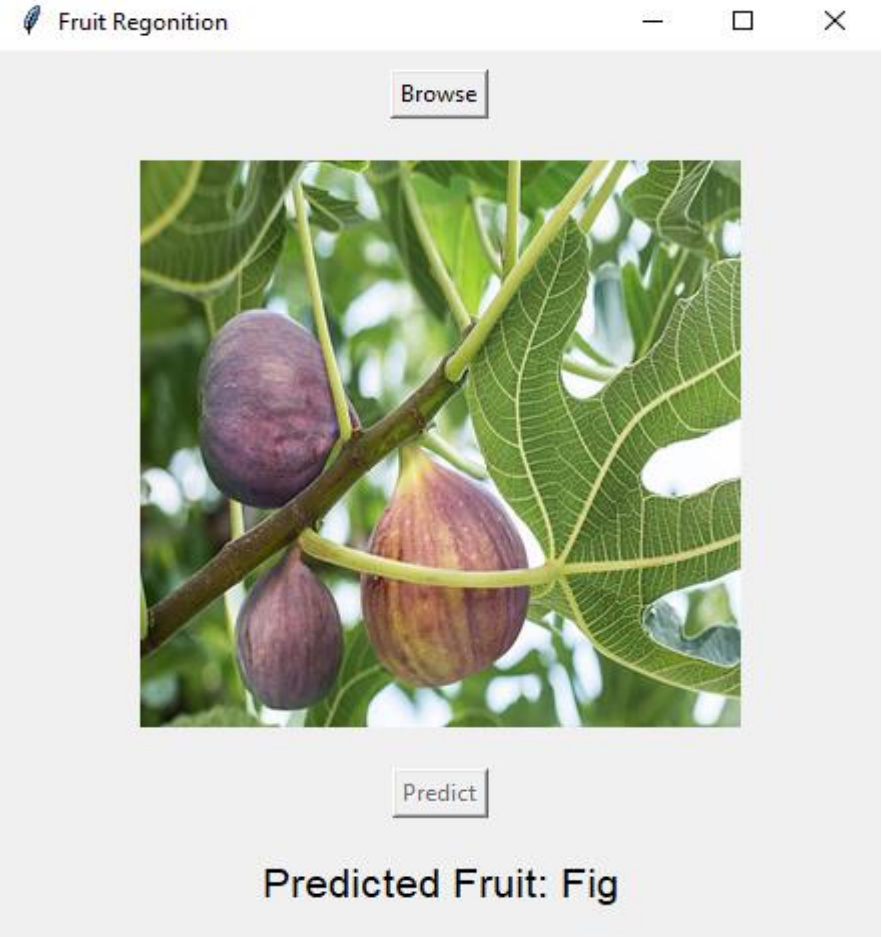
### *FruitRecognitionSystem.py*

```
import tkinter as tk
from tkinter import filedialog
from PIL import Image, ImageTk
import torch
import torchvision.transforms as transforms

#List containing names of Fruit Classes
Fruits = ['Acerola','Apple','Apricot','Avocado','Banana','Black Berry','Blue Berry',
'Cantaloupe','Cherry','Coconut','Fig','Grapefruit','Grape','Guava','Kiwi Fruit','Lemon',
'Lime','Mango','Olive','Orange','Passion Fruit','Peach','Pear','Pineapple','Plum',
'Pomegranate','Raspberry','Strawberry','Tomato','Watermelon']

#Load the Pytorch model
model = torch.load('./fruit_resnet18(88%).pt')

#Set model to evalution mode
model.eval()

#Define the transformation to apply to the input image
transform = transforms.Compose([
      transforms.Resize(224),
      transforms.CenterCrop(224),
      transforms.ToTensor(),
      transforms.Normalize([0.485,0.456,0.406],[0.229,0.224,0.225])
    ])

#Define a function to make a prediction on the input image
def predict_image(image_path):
    image = Image.open(image_path)
    image_tensor = transform(image).unsqueeze(0)
    with torch.no_grad():
        output = model(image_tensor)
        predicted = torch.argmax(output).item()
```

```
    return predicted


#Define function to handle "Browse" button click
def browse_file():
    file_path = filedialog.askopenfilename()
    if file_path:
        image = Image.open(file_path)
        image = image.resize((300,300),Image.LANCZOS)
        photo = ImageTk.PhotoImage(image)
        image_label.configure(image=photo)
        image_label.image = photo
        prediction_label.configure(text="")
        predict_button.configure(state="normal")
        global image_path
        image_path = file_path

#Define function to handle the "Predict" button click
def predict():
    if image_path:
        predicted_class = predict_image(image_path)
        predicted_label = Fruits[predicted_class]
        prediction_label.configure(text=f"Predicted Fruit: {predicted_label}")
    else:
        prediction_label.configure("Please select an image first")
    predict_button.configure(state="disabled")

#Create the main window
window = tk.Tk()
window.title("Fruit Regonition")
window.geometry("440x470")
```

```
#Create the "Browse" label
browse_button = tk.Button(window, text="Browse", command=browse_file)
browse_button.pack(pady=10)

#Create the image label
image_label = tk.Label(window)
image_label.pack(pady=10)

#Create the "Predict" button
predict_button = tk.Button(window, text="Predict", command=predict, state="disabled")
predict_button.pack(pady=10)

#Create a prediction label
prediction_label = tk.Label(window,font=("Arial",16))
prediction_label.pack(pady=10)

#Run the main loop
window.mainloop()
```

# OUTPUT




# FUTURE SCOPE

There are several potential avenues for future work on the Fruit Recognition Application. Here are the few possibilities: -

1. *Expansion of the Fruit Dataset:* While the current fruit dataset used in the application includes *30* types of fruits, there are many other types of fruits that could be added to the dataset to increase the variety and robustness of the CNN model. This would require additional data collection and preprocessing efforts.
2. *Fine-Tuning of Pre-Trained Models:* In addition to ResNet18, there are many other pre-trained models that could be used for transfer learning in the Fruit Recognition Application, such as Inception, VGG and MobileNet. Fine-tuning the accuracy of the CNN model.
3. *Integration of Real-Time Object Detection:* Currently, the Fruit Recognition Application only recognizes fruit types in still images. However, it would be useful to integrate real-

time object detection capabilities to enable the application to recognize fruits in live video streams. This would require the use of object detection algorithms, such as YOLO, SSD or Faster R-CNN.

4. *Integration with Mobile Devices:* Fruit Recognition Application could be further developed to be compatible with mobile devices, such as smartphones or tables, by optimizing the CNN model and the application's user interface for mobile platforms.

These are just a few examples of potential future work for the Fruit Recognition Application. With continued development and improvement, the application could become a useful tool for fruit recognition in various contexts, such as agriculture, food industry, and nutrition.

# CONCLUSION

In conclusion, the Fruit Recognition Application demonstrate the effectiveness of CNNs and transfer learning for image classification tasks. By utilizing the ResNet18 architecture and transfer learning, the application achieved a high accuracy rate in recognizing 30 types of fruits. The application using PyTorch also highlights the benefits of using a powerful and flexible deep learning framework for building and training neural networks.

Furthermore, the potential for future work on the application is vast, from expanding the fruit dataset to integrating real-time object detection capabilities and optimizing the application for mobile devices. These opportunities for improvements highlight the potential of the Fruit Recognition Application as a valuable tool in various domains including agriculture, food industry and nutrition.

Overall, the Fruit Recognition Application serves as a successful example of the power of deep learning and machine learning techniques in solving real-world problems. As the field of deep learning continues to grow and evolve, it is exciting to imagine the potential of future applications and the impact they may have on society.

# REFERENCES

[1] https://python.org/

[2] https://pytorch.org/

[3] https://tkinter.org/

[4] https://pytorch.org/vision/stable/index.html