

**NIT 3251**  
**Advanced Data Science**  
**(Block 3, Semester 4)**

**Group 3**

**Assignment 2**

**Image Classification Using Pre-Trained CNNs (Transfer Learning with Customization)**

**Fruits and Vegetables.**

**Hernandez Romero, Diego Alejandro, s8146903**

**Crossley, Daniel, s4681821**

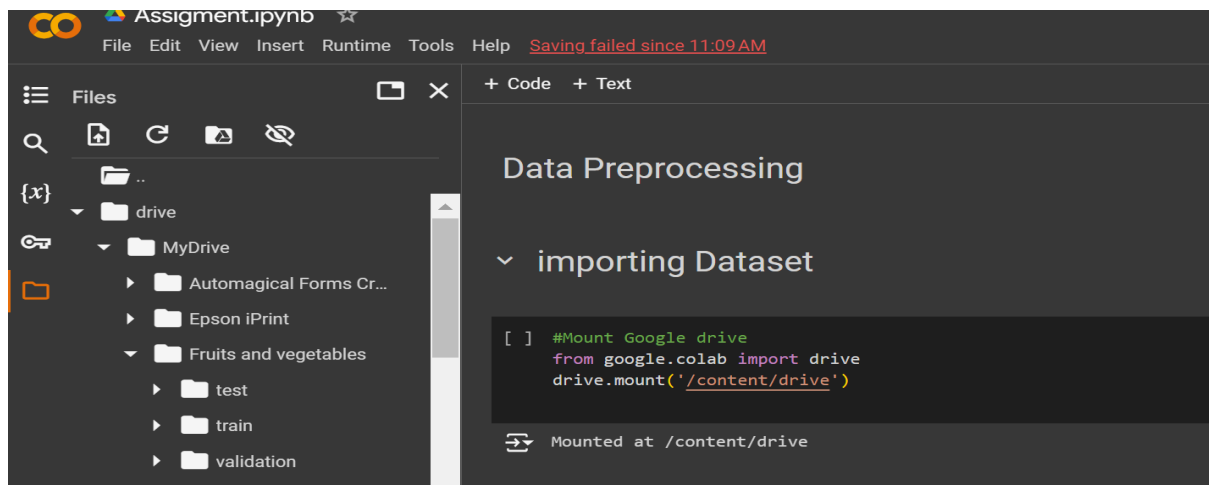
**Qasimi, Farzana, s8078482**

Accurate classification of fruits and vegetables using image recognition technologies is essential for a variety of industries, including agriculture, retail, and supply chain management. Traditional machine learning approaches often require large amounts of labeled data and significant computational resources to achieve high accuracy. However, with the growing complexity of deep learning models, training from scratch becomes impractical for many specific tasks due to time constraints and limited data availability. The objective is to classify various types of fruits and vegetables from the Fruits 360 dataset, which is publicly available on Kaggle.

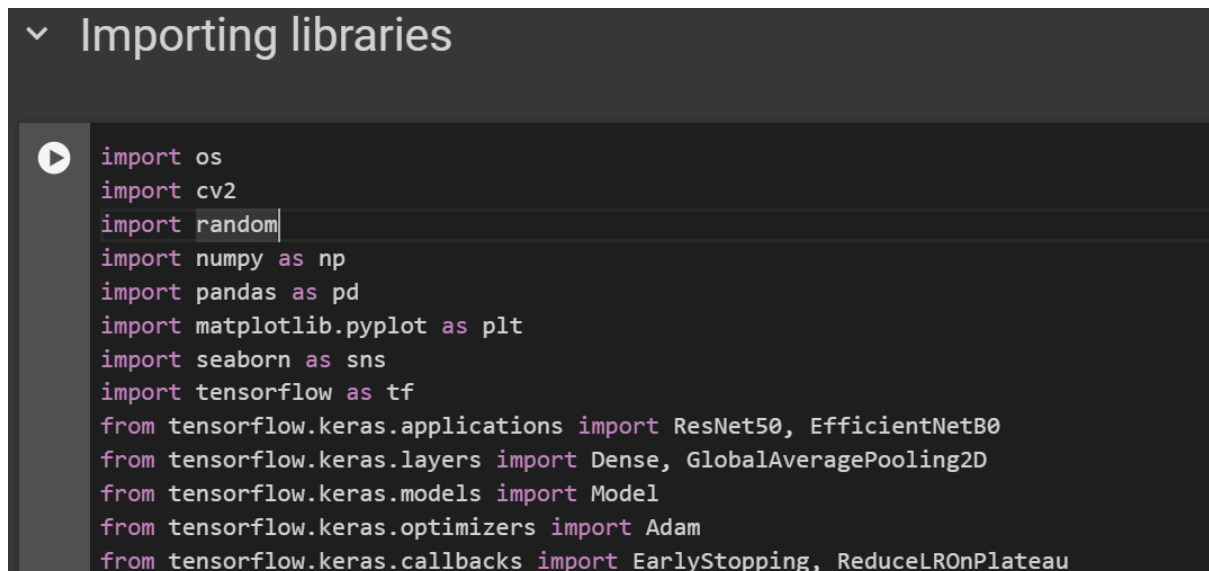
This project aims to address the challenge of fruit and vegetable classification by leveraging transfer learning through fine-tuning pre-trained convolutional neural network (CNN) models, specifically ResNet or EfficientNet, to enhance classification accuracy on the Fruits 360 Dataset. The main problem is how to best customize these models for this specific classification task while ensuring efficiency and high performance.

The pre-train model that we used for this project was ResNet, the reason for choosing this is because ResNet is a top choice for image classification tasks due to its residual learning architecture, which prevents vanishing gradients and enables the training of deep networks. This helps capture intricate features in data like the Fruits 360 dataset. Pre-trained on large datasets like ImageNet, ResNet transfers useful knowledge, leading to faster fine-tuning and better accuracy. Its strong generalization capabilities and widespread use make it a reliable and efficient option for fine-tuning in fruit and vegetable classification tasks.

## Importing Dataset




## Importing libraries



## Training image preprocessing

### Training Image preprocessing

```
[ ] # Load and preprocess the training dataset
import tensorflow as tf
training_set = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/Fruits and vegetables',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(64, 64),
    shuffle=True
)
```


 Found 3830 files belonging to 3 classes.

## Validation image preprocessing

### Validation Image Preprocessing

```
[ ] #Load and preprocess the validation dataset
import tensorflow as tf

validation_set = tf.keras.utils.image_dataset_from_directory(
    '/content/drive/MyDrive/Fruits and vegetables/validation',
    labels="inferred",
    label_mode="categorical",
    color_mode="rgb",
    batch_size=32,
    image_size=(64, 64),
    shuffle=True
)
```

 Found 351 files belonging to 36 classes.

## Data augmentation

### ✓ Data Augmentation

```
[ ] from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Define the image data generator for data augmentation
datagen = ImageDataGenerator(
    rescale=1./255, # Normalize pixel values to [0, 1]
    rotation_range=20, # Randomly rotate images by up to 20 degrees
    width_shift_range=0.2, # Shift images horizontally by up to 20%
    height_shift_range=0.2, # Shift images vertically by up to 20%
    shear_range=0.2, # Shear images by up to 20%
    zoom_range=0.2, # Randomly zoom in on images by up to 20%
    horizontal_flip=True, # Randomly flip images horizontally
    fill_mode='nearest' # Fill in missing pixels after transformations
)

train_dir = '/content/drive/MyDrive/Fruits and vegetables'

# Generate the augmented training set
augmented_training_set = datagen.flow_from_directory(
    train_dir, # Directory containing the training images
    target_size=(64, 64), # Resize images to 64x64 pixels
    batch_size=32, # Load images in batches of 32
    class_mode='categorical' # Use categorical labels (for multi-class classification)
)

# Print the summary
print("Data augmentation applied. Training set is ready.")
```

Found 3830 images belonging to 3 classes.  
Data augmentation applied. Training set is ready.

## Data normalization

```
[3] # Importing necessary modules including ImageDataGenerator
from tensorflow.keras.preprocessing.image import ImageDataGenerator

# Data normalization for the validation set
validation_datagen = ImageDataGenerator(rescale=1./255)

augmented_validation_set = validation_datagen.flow_from_directory(
    '/content/drive/MyDrive/Fruits and vegetables/validation',
    target_size=(64, 64),
    batch_size=32,
    class_mode='categorical'
)

print("Validation set with normalization is ready.")
```

Found 351 images belonging to 36 classes.  
Validation set with normalization is ready.

## Visualization augmented images

### Visualizing Augmented Images

```
[21] # Visualizing Augmented Images
def visualize_augmented_images(datagen, directory):
    batch = datagen.flow_from_directory(
        directory,
        target_size=(124, 124),
        batch_size=4,
        class_mode='categorical'
    )

    images, labels = next(batch)

    fig, axes = plt.subplots(1, 4, figsize=(12, 12))
    axes = axes.flatten()
    for img, ax in zip(images, axes):
        ax.imshow(img)
        ax.axis('off')
    plt.tight_layout()
    plt.show()

# Call the visualization function
visualize_augmented_images(datagen, train_dir)

print("Data augmentation and normalization applied. Training set is ready.")
```

Found 3830 images belonging to 3 classes.



Data augmentation and normalization applied. Training set is ready.

## Model Building - Diego

### Model building

```
#load model
base_model = ResNet50(weights='imagenet', include_top=False, input_shape=(64, 64, 3))

base_model.trainable = False

# Add new classification layers
x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation='relu')(x)
output = Dense(training_set.num_classes, activation='softmax')(x)

# Define the model
model = Model(inputs=base_model.input, outputs=output)

# Compile the model
model.compile(optimizer=Adam(learning_rate=1e-4), loss='categorical_crossentropy', metrics=['accuracy'])

# Print model summary
model.summary()
```

## Training the model

```
# Define callbacks
early_stopping = EarlyStopping(monitor='val_loss', patience=5, restore_best_weights=True)
reduce_lr = ReduceLROnPlateau(monitor='val_loss', factor=0.2, patience=3, min_lr=1e-6)

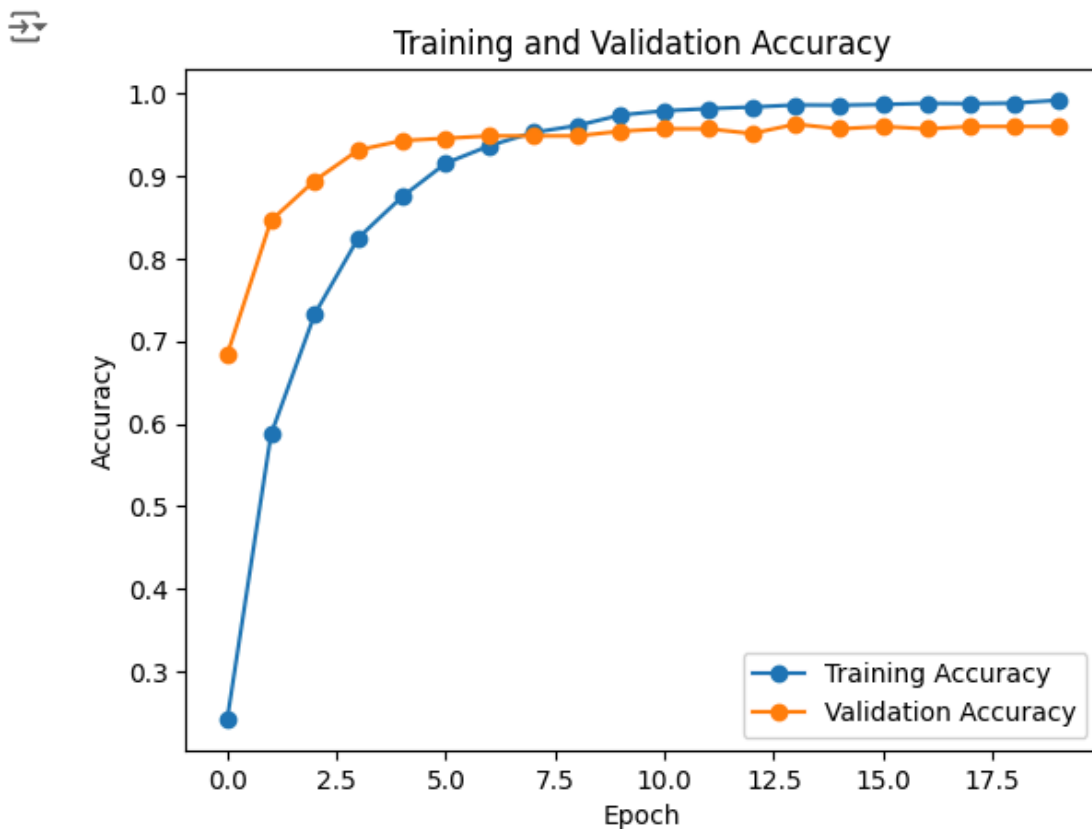
# Train the model
history = model.fit(
    training_set,
    validation_data=validation_set,
    epochs=20,
    callbacks=[early_stopping, reduce_lr]
)
```

## Model Eval - Daniel

### Evaluating the model

### Plot accuracy of model over multiple epochs

```
plt.plot(history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.title('Training and Validation Accuracy')
plt.legend()
plt.show()
```



In the final epochs of training, both the training and validation accuracy curves have leveled off, indicating that the model has converged and is no longer improving with additional epochs. The training accuracy approaches 100%, while the validation accuracy stabilizes around 90%, with only a small gap between the two. This suggests that the model is not overfitting and is generalizing well to unseen data.

## ✓ Use the model for prediction

Show the confidence of the prediction, as well as the actual classification.

```
[49] # Find all the possible class names and display them.
      class_names = sorted(os.listdir(train_dir))
      print("Class Names:", class_names)
```

➡ Class Names: ['apple', 'banana', 'beetroot', 'bell pepper', 'cabbage', 'c  
Number of Classes: 36

✓  
0s

```
▶ def predict(model, img):
    img_array = tf.keras.utils.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)

    predictions = model.predict(img_array)

    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100*(np.max(predictions[0])), 0)
    return predicted_class, confidence
```

✓  
0s

```
▶ import matplotlib.pyplot as plt

# Assuming val_ds is your validation dataset
plt.figure(figsize=(15, 15))
for images, labels in validation_set.take(1):
    for i in range(25):
        ax = plt.subplot(5, 5, i+1) # Adjust the subplot layout as per your preference
        plt.imshow(images[i].numpy().astype("uint8"))

        predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[np.argmax(labels[i])]

        plt.title(f"Actual: {actual_class}, \n Predicted: {predicted_class}.\n Confidence: {confidence}%")

    plt.axis('off')

plt.tight_layout()
plt.show()
```

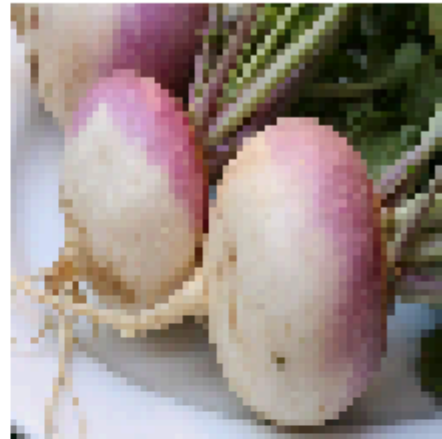


For the sake of length, the predictions shown have been reduced.

Actual: jalepeno,  
Predicted: jalepeno.  
Confidence: 99.0%



Actual: turnip,  
Predicted: turnip.  
Confidence: 100.0%



Actual: bell pepper,  
Predicted: bell pepper.  
Confidence: 99.0%



Actual: watermelon,  
Predicted: watermelon.  
Confidence: 95.0%



Actual: sweetpotato,  
Predicted: sweetpotato.  
Confidence: 100.0%



Actual: apple,  
Predicted: carrot.  
Confidence: 53.0%



As shown above, the predictions are very accurate. However, for images that lie outside the scope of realistic images (as shown by the Apple Logo image), the accuracy falls massively. To solve this problem we could add some more images in the dataset and not just realistic images like the ones we used but also more abstract ones like the Apple Logo, we could also improve the model by fine-tuning more layers to capture more differences between classes.

In this project, we focused on improving the classification of fruits and vegetables using pre-trained models like ResNet. By building on existing knowledge from these models and applying techniques like data augmentation, we were able to enhance the accuracy of the model. Although there were some initial misclassifications, refining our approach with better training and adjustments helped improve the results. Overall, this project demonstrated how using pre-trained models can effectively solve specific image classification tasks with a little fine-tuning.

## References

Pourmoradi, N. (2024, Feb 10).  
<https://www.kaggle.com/code/nimapourmoradi/fruits-and-vegetables-image-mobilenetv2>

Spotless Tech. *Fruits and Vegetables Recognition System Part-2* | *Data Preprocessing using Keras API*.<https://www.youtube.com/watch?v=NJX8FQZbA6Q>

## Appendix A – Group meeting minute

<b>Meeting/Project Name:</b>	<b>Group Meeting Image Classification (fruits and vegetables)</b>		
<b>Date of Meeting:</b> (MM/DD/YYYY)	14/10/2024	<b>Time:</b>	<b>2.30PM</b>
<b>Minutes Prepared By:</b>	Farzana	<b>Location:</b>	VU Footscray
<b>1. Meeting Objective</b>			
Discussing on which part should each member do for stage 1			
<b>2. Attendance at Meeting</b>			
<b>Name</b>	<b>Department/Division</b>	<b>E-mail</b>	<b>Phone</b>
Farzana	IT	Farzana.qasimi@live.vu.edu	
Diego	IT	s8146903	
Daniel	IT	daniel.crossley1@live.vu.edu	
<b>3. Agenda and Notes, Decisions, Issues</b>			
<b>Topic</b>	<b>Owner</b>	<b>Time</b>	
Deciding on who to do Pre Processing	Farzana	2.30PM up to 3PM	
Deciding on who to do Model Building	Diego	2.30PM up to 3PM	

Deciding on who to do Model Evalvet	Daniel	2.30PM up to 3PM
4. Action Items		
<b>Action</b>	<b>Owner</b>	<b>Due Date</b>
I did data selection, Pre Processing and data exploration	Farzana	
I did the model building and Training	Diego	
I did model evaluation and testing	Daniel	