# Student Name : Vibek Rana Magar

# Student Number : 2358119

```
from google.colab import drive
drive.mount('/content/drive')

Mounted at /content/drive

import os
import random
import matplotlib.pyplot as plt
from PIL import Image

train_path = '/content/drive/MyDrive/AI &
ML/Week_5/FruitinAmazon/train'
test_path = '/content/drive/MyDrive/AI & ML/Week_5/FruitinAmazon/test'
```

**3. Task - 1:**

**Repeat all the task from worksheet - 5 but, try to improve the model from last week with same dataset.**

**• Use Data Augmentation to increase the number of training image.**

**• Use deeper model with BN and DropOut layer as presented above.**

**• Understand the Model Summary and Training Behavior.**

```
import tensorflow as tf
from tensorflow.keras import layers, models, regularizers
import matplotlib.pyplot as plt

train_dir = '/content/drive/MyDrive/AI &
ML/Week_5/FruitinAmazon/train'
img_height, img_width = 128, 128
batch_size = 32

train_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=(img_height, img_width),
    batch_size=batch_size,
    label_mode='int',
    validation_split=0.2,
    subset='training',
    seed=123
)
```

```
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    image_size=(img_height, img_width),
    batch_size=batch_size,
    label_mode='int',
    validation_split=0.2,
    subset='validation',
    seed=123
)

class_names = train_ds.class_names
print("Class names:", class_names)

data_augmentation = tf.keras.Sequential([
    layers.RandomFlip("horizontal"),
    layers.RandomRotation(0.1),
    layers.RandomZoom(0.1),
    layers.RandomContrast(0.1),
])

def create_improved_model(input_shape=(128, 128, 3), num_classes=6):
    model = models.Sequential([
        layers.Input(shape=input_shape),
        data_augmentation,

        layers.Rescaling(1./255),

        layers.Conv2D(32, (3, 3), padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(64, (3, 3), padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(128, (3, 3), padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.MaxPooling2D((2, 2)),
        layers.Dropout(0.25),

        layers.Conv2D(256, (3, 3), padding='same'),
        layers.BatchNormalization(),
        layers.Activation('relu'),
        layers.GlobalAveragePooling2D(),
        layers.Dropout(0.5),
        layers.Dense(512, activation='relu',
kernel_regularizer=regularizers.l2(0.01)),
        layers.BatchNormalization(),
        layers.Dropout(0.5),

        layers.Dense(num_classes, activation='softmax')
    ])
    return model

improved_model = create_improved_model(
    input_shape=(img_height, img_width, 3),
    num_classes=len(class_names)
)

improved_model.compile(
    optimizer=tf.keras.optimizers.Adam(learning_rate=0.0001),
    loss='sparse_categorical_crossentropy',
    metrics=['accuracy']
)

improved_model.summary()

history = improved_model.fit(
    train_ds,
    validation_data=val_ds,
    epochs=100,
    callbacks=[
        tf.keras.callbacks.ModelCheckpoint('best_model.h5',
save_best_only=True),
        tf.keras.callbacks.EarlyStopping(patience=15)
    ]
)

plt.figure(figsize=(12, 4))
plt.subplot(1, 2, 1)
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.legend()
plt.subplot(1, 2, 2)
plt.plot(history.history['loss'], label='Train Loss')
plt.plot(history.history['val_loss'], label='Val Loss')
plt.legend()
plt.show()
```

```
Found 90 files belonging to 6 classes.
Using 72 files for training.
Found 90 files belonging to 6 classes.
Using 18 files for validation.
```

```
Class names: ['acai', 'cupuacu', 'graviola', 'guarana', 'pupunha',
'tucuma']

Model: "sequential_4"

┏━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━━━━━━━━━━━━━━━━━━━━━━━━━━━━━┳━
━━━━━━━━━━━━━━━━━┓
┃ Layer (type)                         ┃ Output Shape                ┃
Param # ┃
┡━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━━━━━━━━━━━━━━━━━━━━━━━━━━━━━╇━
━━━━━━━━━━━━━━━━━┩
│ sequential_3 (Sequential)            │ (None, 128, 128, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ rescaling_1 (Rescaling)              │ (None, 128, 128, 3)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ conv2d_4 (Conv2D)                    │ (None, 128, 128, 32)        │
896 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ batch_normalization_5                │ (None, 128, 128, 32)        │
128 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ activation_4 (Activation)            │ (None, 128, 128, 32)        │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ max_pooling2d_3 (MaxPooling2D)       │ (None, 64, 64, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ dropout_5 (Dropout)                  │ (None, 64, 64, 32)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ conv2d_5 (Conv2D)                    │ (None, 64, 64, 64)          │
18,496 │
├──────────────────────────────────────┼─────────────────────────────┼─
─────────────────┤
│ batch_normalization_6                │ (None, 64, 64, 64)          │
256 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼─
```

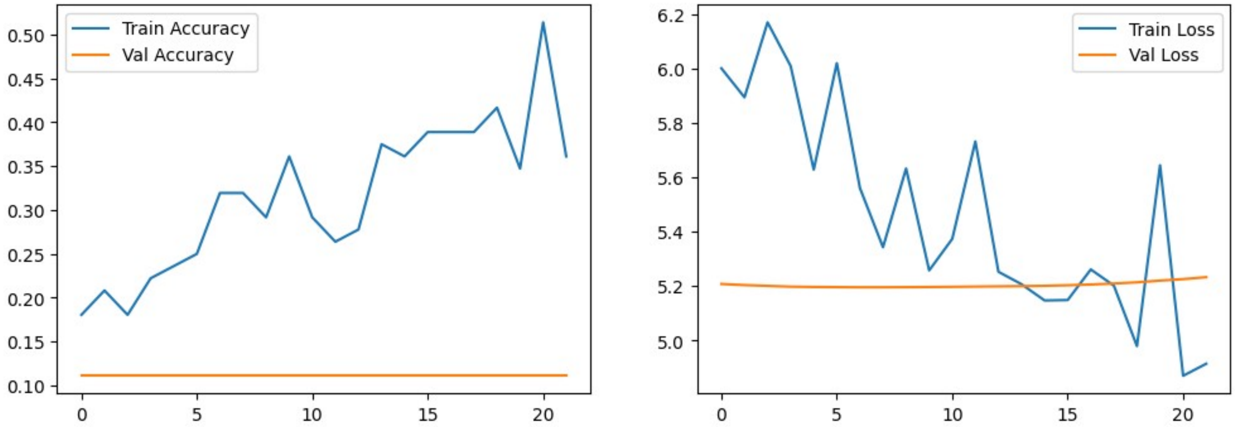
```
─────────────────┤
│ activation_5 (Activation)            │ (None, 64, 64, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_4 (MaxPooling2D)       │ (None, 32, 32, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_6 (Dropout)                  │ (None, 32, 32, 64)          │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_6 (Conv2D)                    │ (None, 32, 32, 128)         │
73,856 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_7                │ (None, 32, 32, 128)         │
512 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ activation_6 (Activation)            │ (None, 32, 32, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ max_pooling2d_5 (MaxPooling2D)       │ (None, 16, 16, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_7 (Dropout)                  │ (None, 16, 16, 128)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ conv2d_7 (Conv2D)                    │ (None, 16, 16, 256)         │
295,168 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_8                │ (None, 16, 16, 256)         │
1,024 │
│ (BatchNormalization)                 │                             │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ activation_7 (Activation)            │ (None, 16, 16, 256)         │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
```

```
│ global_average_pooling2d_1           │ (None, 256)                  │
0 │
│ (GlobalAveragePooling2D)             │                              │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_8 (Dropout)                  │ (None, 256)                  │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_2 (Dense)                      │ (None, 512)                  │
131,584 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ batch_normalization_9                │ (None, 512)                  │
2,048 │
│ (BatchNormalization)                 │                              │
│
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dropout_9 (Dropout)                  │ (None, 512)                  │
0 │
├──────────────────────────────────────┼─────────────────────────────┼
─────────────────┤
│ dense_3 (Dense)                      │ (None, 6)                    │
3,078 │
└──────────────────────────────────────┴─────────────────────────────┴
─────────────────┘

 Total params: 527,046 (2.01 MB)

 Trainable params: 525,062 (2.00 MB)

 Non-trainable params: 1,984 (7.75 KB)

Epoch 1/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.1279 - loss: 6.2653

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 14s 2s/step - accuracy: 0.1411 - loss:
6.1991 - val_accuracy: 0.1111 - val_loss: 5.2077
Epoch 2/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.1892 - loss: 5.9016

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
```

```
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.1940 - loss:
5.8996 - val_accuracy: 0.1111 - val_loss: 5.2036
Epoch 3/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.1852 - loss: 6.1824

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 6s 2s/step - accuracy: 0.1840 - loss: 6.1791
- val_accuracy: 0.1111 - val_loss: 5.2006
Epoch 4/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.2043 - loss: 6.0135

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.2088 - loss:
6.0123 - val_accuracy: 0.1111 - val_loss: 5.1978
Epoch 5/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.2818 - loss: 5.5803

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.2704 - loss:
5.5922 - val_accuracy: 0.1111 - val_loss: 5.1967
Epoch 6/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 2s/step - accuracy: 0.2396 - loss: 6.0203

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.2422 - loss:
6.0200 - val_accuracy: 0.1111 - val_loss: 5.1961
```

```
Epoch 7/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 0s 1s/step - accuracy: 0.3409 - loss: 5.5447

WARNING:absl:You are saving your model as an HDF5 file via
`model.save()` or `keras.saving.save_model(model)`. This file format
is considered legacy. We recommend using instead the native Keras
format, e.g. `model.save('my_model.keras')` or
`keras.saving.save_model(model, 'my_model.keras')`.

 3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 2s/step - accuracy: 0.3355 - loss: 5.5487
- val_accuracy: 0.1111 - val_loss: 5.1956
Epoch 8/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.2886 - loss: 5.3100
- val_accuracy: 0.1111 - val_loss: 5.1957
Epoch 9/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.2943 - loss: 5.6821
- val_accuracy: 0.1111 - val_loss: 5.1960
Epoch 10/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 6s 2s/step - accuracy: 0.3720 - loss: 5.2126
- val_accuracy: 0.1111 - val_loss: 5.1965
Epoch 11/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.3099 - loss: 5.3117
- val_accuracy: 0.1111 - val_loss: 5.1972
Epoch 12/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.2726 - loss: 5.6470
- val_accuracy: 0.1111 - val_loss: 5.1981
Epoch 13/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 12s 2s/step - accuracy: 0.2483 - loss: 5.2958
- val_accuracy: 0.1111 - val_loss: 5.1989
Epoch 14/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 9s 2s/step - accuracy: 0.3633 - loss: 5.2877
- val_accuracy: 0.1111 - val_loss: 5.1997
Epoch 15/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.3446 - loss: 5.1824
- val_accuracy: 0.1111 - val_loss: 5.2008
Epoch 16/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.4132 - loss: 5.0935
- val_accuracy: 0.1111 - val_loss: 5.2029
Epoch 17/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.3780 - loss: 5.2387
- val_accuracy: 0.1111 - val_loss: 5.2058
Epoch 18/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.3976 - loss: 5.1454
- val_accuracy: 0.1111 - val_loss: 5.2093
Epoch 19/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.4427 - loss: 4.8793
- val_accuracy: 0.1111 - val_loss: 5.2138
Epoch 20/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 11s 2s/step - accuracy: 0.3611 - loss: 5.5412
- val_accuracy: 0.1111 - val_loss: 5.2200
```

```
Epoch 21/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 8s 2s/step - accuracy: 0.5069 - loss: 4.9175
- val_accuracy: 0.1111 - val_loss: 5.2255
Epoch 22/100
3/3 ━━━━━━━━━━━━━━━━━━━━ 10s 2s/step - accuracy: 0.3407 - loss: 4.9590
- val_accuracy: 0.1111 - val_loss: 5.2327
```


Train Accuracy
Val Accuracy
Train Loss
Val Loss