MODEL COMPARISON

BRIEF OVERVIEW AND PERFORMANCE

Optimizer used- Adam

Loss Function- Categorical Crossentropy

Model	Accuracy	Precision	Recall	F1 Score
1	0.897260	0.904018	0.897260	0.898359
2	0.890411	0.896222	0.890411	0.889731
3	0.893836	0.906641	0.893836	0.895713
4	0.750000	0.792586	0.750000	0.756804
5	0.907534	0.910789	0.907534	0.907227
6	0.708904	0.728991	0.708904	0.713543

MODEL- 1(BASE)

Architecture- 2 Convolution-Pooling layers

```
model1 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(128, activation="relu"),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```

MODEL- 2(DEEPER CNN MODEL)

Architecture- 3 Convolution-Pooling layers

```
model2 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(256, activation="relu"),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```

MODEL- 3(CNN + BATCH DROPOUT)

Architecture- 2 Convolution-Pooling layers with a Dense layer with 50% dropout

```
model3 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(256, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```

MODEL- 4(CNN + BATCH NORMALISATION)

Architecture- 2 Convolution-Normalisation-Pooling layers

```
model4 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.BatchNormalization(),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(256, activation="relu"),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```

MODEL- 5(EVEN DEEPER CNN)(BEST MODEL)

Architecture- 4 Convolution-Pooling layers

```
model5 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(256, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(512, activation="relu"),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```

MODEL- 6(DEEPER CNN + DROPOUT)

Architecture- 3 Convolution-Pooling layers with Dense layer and 50% dropout

```
model6 = models.Sequential([
    layers.Conv2D(32, (3,3), activation="relu", input_shape=(224,224,3)),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(64, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Conv2D(128, (3,3), activation="relu"),
    layers.MaxPooling2D(2,2),

    layers.Flatten(),
    layers.Dense(512, activation="relu"),
    layers.Dropout(0.5),
    layers.Dense(NUM_CLASSES, activation="softmax")
])
```