

Image Classification Assignment

Student Information Name: Marya mohammed bawazir.

Student ID:253000692

In this assignment, the Rock-Paper-Scissors image dataset from Kaggle is used to develop an image classification system. The dataset is first downloaded and prepared by splitting it into training, validation, and testing sets. Data augmentation techniques are then applied to improve the diversity of the training images and enhance the model's generalization performance.

Three pre-trained deep learning models are implemented and evaluated:

EfficientNetB0- ResNet50- MobileNetV2

Each model is trained and tested using the same dataset. Their performance is evaluated using Accuracy, Precision, Recall, F1-score, and Confusion Matrix. Finally, a comparison is performed to identify the model that achieves the best classification performance on the Rock-Paper-Scissors dataset.

```
from google.colab import drive
drive.mount('/content/drive', force_remount=True)

Mounted at /content/drive

!unzip "/content/drive/MyDrive/Rock-Paper-Scissors Images.zip" -d
"/content/dataset"

Archive: /content/drive/MyDrive/Rock-Paper-Scissors Images.zip
replace /content/dataset/README_rpc-cv-images.txt? [y]es, [n]o, [A]ll,
[N]one, [r]ename:

import os

os.listdir("/content/dataset")

['scissors', 'rock', 'rps-cv-images', 'README_rpc-cv-images.txt',
'paper']
```

Step 1: Install split-folders

```
!pip install split-folders

Requirement already satisfied: split-folders in
/usr/local/lib/python3.12/dist-packages (0.6.1)
```

Step 2: Split the dataset

```
import splitfolders

splitfolders.ratio(
```

```
    "/content/dataset/rps-cv-images",  
    output="/content/data",  
    seed=42,  
    ratio=(0.7, 0.15, 0.15)  
)  
Copying files: 2188 files [00:04, 459.33 files/s]
```

Step 3: Create the data generators

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator  
  
train_datagen = ImageDataGenerator(  
    rescale=1./255,  
    rotation_range=20,  
    zoom_range=0.2,  
    horizontal_flip=True  
)  
  
test_datagen = ImageDataGenerator(rescale=1./255)
```

Step 4: Load the images

```
train_data = train_datagen.flow_from_directory(  
    "/content/data/train",  
    target_size=(150,150),  
    batch_size=32,  
    class_mode="categorical"  
)  
  
val_data = test_datagen.flow_from_directory(  
    "/content/data/val",  
    target_size=(150,150),  
    batch_size=32,  
    class_mode="categorical"  
)  
  
test_data = test_datagen.flow_from_directory(  
    "/content/data/test",  
    target_size=(150,150),  
    batch_size=32,  
    class_mode="categorical",  
    shuffle=False  
)  
  
Found 1531 images belonging to 3 classes.  
Found 326 images belonging to 3 classes.  
Found 331 images belonging to 3 classes.
```

Step 5: Build the CNN model

```

from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten,
Dense

model = Sequential([
    Conv2D(32, (3,3), activation='relu', input_shape=(150,150,3)),
    MaxPooling2D(),

    Conv2D(64, (3,3), activation='relu'),
    MaxPooling2D(),

    Flatten(),

    Dense(128, activation='relu'),

    Dense(3, activation='softmax')
])

/usr/local/lib/python3.12/dist-packages/keras/src/layers/
convolutional/base_conv.py:113: UserWarning: Do not pass an
`input_shape`/`input_dim` argument to a layer. When using Sequential
models, prefer using an `Input(shape)` object as the first layer in
the model instead.
  super().__init__(activity_regularizer=activity_regularizer,
**kwargs)

```

Step 6: Compile the model

```

model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

```

Step 7: Train the model

```

history = model.fit(
    train_data,
    validation_data=val_data,
    epochs=5
)

```

Epoch 1/5
48/48 ————— 82s 2s/step - accuracy: 0.4775 - loss: 1.2715 - val_accuracy: 0.8252 - val_loss: 0.5955
Epoch 2/5
48/48 ————— 78s 2s/step - accuracy: 0.8027 - loss: 0.5345 - val_accuracy: 0.8988 - val_loss: 0.3508
Epoch 3/5
48/48 ————— 78s 2s/step - accuracy: 0.9027 - loss:

```
0.2981 - val_accuracy: 0.9325 - val_loss: 0.2065
Epoch 4/5
48/48 ━━━━━━━━━━━ 78s 2s/step - accuracy: 0.9157 - loss:
0.2392 - val_accuracy: 0.9755 - val_loss: 0.1093
Epoch 5/5
48/48 ━━━━━━━━━━━ 77s 2s/step - accuracy: 0.9425 - loss:
0.1706 - val_accuracy: 0.9816 - val_loss: 0.0555
```

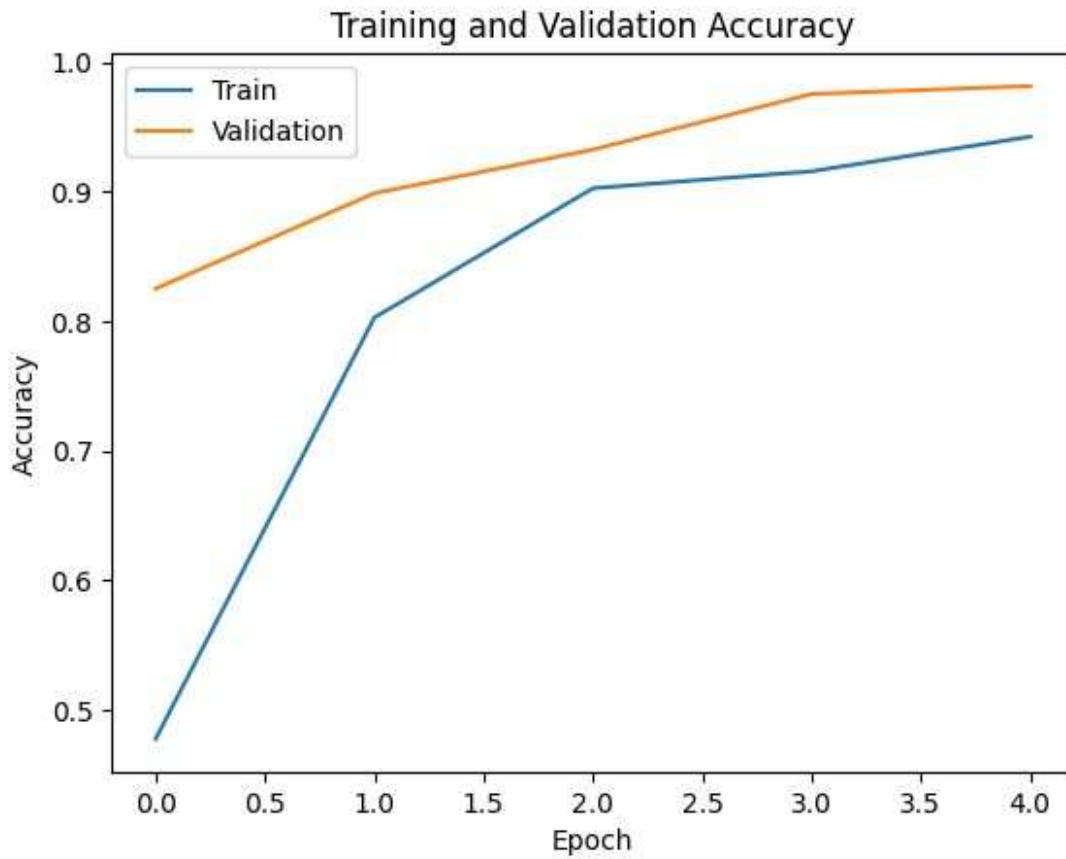
Step 8: Test the model

```
loss, accuracy = model.evaluate(test_data)
print("Test Accuracy:", accuracy)
11/11 ━━━━━━━━━━━ 5s 445ms/step - accuracy: 0.9517 - loss:
0.1227
Test Accuracy: 0.9516616463661194
```

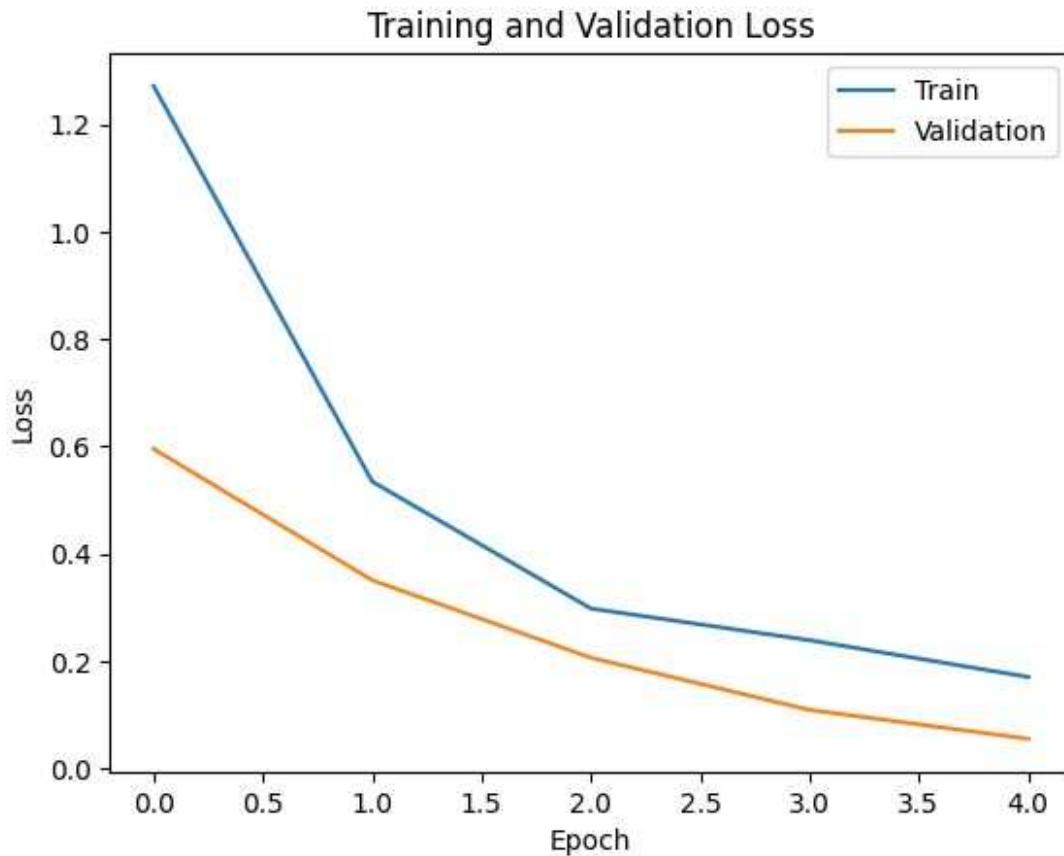
Step 9: Plot the results

```
import matplotlib.pyplot as plt

plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title("Training and Validation Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend(["Train", "Validation"])
plt.show()
```



```
!ls /content/dataset  
  
plt.plot(history.history['loss'])  
plt.plot(history.history['val_loss'])  
plt.title("Training and Validation Loss")  
plt.xlabel("Epoch")  
plt.ylabel("Loss")  
plt.legend(["Train", "Validation"])  
plt.show()
```



1. Reload images with 224×224

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

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    "/content/data/train",
    target_size=(224,224),
    batch_size=32,
    class_mode="categorical"
)

val_data = test_datagen.flow_from_directory(
    "/content/data/val",
    target_size=(224,224),
    batch_size=32,
```

```

        class_mode="categorical"
    )

test_data = test_datagen.flow_from_directory(
    "/content/data/test",
    target_size=(224,224),
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)

Found 1531 images belonging to 3 classes.
Found 326 images belonging to 3 classes.
Found 331 images belonging to 3 classes.

import os

for folder in ['train', 'val', 'test']:
    total = 0
    path = f'/content/data/{folder}'
    for cls in os.listdir(path):
        total += len(os.listdir(os.path.join(path, cls)))
    print(f"{folder}: {total} images")

train: 1531 images
val: 326 images
test: 331 images

```

1. Function to build transfer learning models

```

import tensorflow as tf
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
Dropout
from tensorflow.keras.applications import EfficientNetB0, ResNet50,
DenseNet121

num_classes = train_data.num_classes

def build_model(base_model):
    base_model.trainable = False

    x = base_model.output
    x = GlobalAveragePooling2D()(x)
    x = Dense(128, activation='relu')(x)
    x = Dropout(0.3)(x)
    output = Dense(num_classes, activation='softmax')(x)

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

    model.compile(

```

```

        optimizer='adam',
        loss='categorical_crossentropy',
        metrics=['accuracy']
    )

    return model

```

1. EfficientNetB0

```

efficient_base = EfficientNetB0(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3)
)

efficient_model = build_model(efficient_base)

history_efficient = efficient_model.fit(
    train_data,
    validation_data=val_data,
    epochs=3
)

efficient_loss, efficient_acc = efficient_model.evaluate(test_data)
print("EfficientNetB0 Test Accuracy:", efficient_acc)

```

Epoch 1/3
48/48 ————— 179s 3s/step - accuracy: 0.3325 - loss: 1.1294 - val_accuracy: 0.3313 - val_loss: 1.1004
Epoch 2/3
48/48 ————— 164s 3s/step - accuracy: 0.3455 - loss: 1.1048 - val_accuracy: 0.3313 - val_loss: 1.1030
Epoch 3/3
48/48 ————— 178s 4s/step - accuracy: 0.3455 - loss: 1.1022 - val_accuracy: 0.3252 - val_loss: 1.1022
11/11 ————— 24s 2s/step - accuracy: 0.3263 - loss: 1.1021
EfficientNetB0 Test Accuracy: 0.3262839913368225

1. ResNet50

```

resnet_base = ResNet50(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3)
)

resnet_model = build_model(resnet_base)

history_resnet = resnet_model.fit(
    train_data,

```

```

        validation_data=val_data,
        epochs=3
    )

resnet_loss, resnet_acc = resnet_model.evaluate(test_data)
print("ResNet50 Test Accuracy:", resnet_acc)

Epoch 1/3
48/48 _____ 398s 8s/step - accuracy: 0.3370 - loss:
1.1333 - val_accuracy: 0.3466 - val_loss: 1.0932
Epoch 2/3
48/48 _____ 365s 8s/step - accuracy: 0.3305 - loss:
1.1028 - val_accuracy: 0.3436 - val_loss: 1.0921
Epoch 3/3
48/48 _____ 366s 8s/step - accuracy: 0.3671 - loss:
1.0933 - val_accuracy: 0.3436 - val_loss: 1.0891
11/11 _____ 62s 6s/step - accuracy: 0.3414 - loss:
1.0899
ResNet50 Test Accuracy: 0.3413897156715393

```

1. MobileNetV2

```

from tensorflow.keras.preprocessing.image import ImageDataGenerator

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=20,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_data = train_datagen.flow_from_directory(
    "/content/data/train",
    target_size=(160,160),
    batch_size=32,
    class_mode="categorical"
)

val_data = test_datagen.flow_from_directory(
    "/content/data/val",
    target_size=(160,160),
    batch_size=32,
    class_mode="categorical"
)

test_data = test_datagen.flow_from_directory(
    "/content/data/test",
    target_size=(160,160),

```

```

        batch_size=32,
        class_mode="categorical",
        shuffle=False
    )

    Found 1531 images belonging to 3 classes.
    Found 326 images belonging to 3 classes.
    Found 331 images belonging to 3 classes.

import tensorflow as tf
from tensorflow.keras.applications import MobileNetV2
from tensorflow.keras.models import Model
from tensorflow.keras.layers import GlobalAveragePooling2D, Dense,
Dropout

num_classes = train_data.num_classes

mobilenet_base = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(160,160,3)
)

mobilenet_base.trainable = False

x = mobilenet_base.output
x = GlobalAveragePooling2D()(x)
x = Dense(128, activation='relu')(x)
x = Dropout(0.3)(x)
output = Dense(num_classes, activation='softmax')(x)

mobilenet_model = Model(inputs=mobilenet_base.input, outputs=output)

mobilenet_model.compile(
    optimizer='adam',
    loss='categorical_crossentropy',
    metrics=['accuracy']
)

history_mobilenet = mobilenet_model.fit(
    train_data,
    validation_data=val_data,
    epochs=2
)

mobilenet_loss, mobilenet_acc = mobilenet_model.evaluate(test_data)

print("MobileNetV2 Test Accuracy:", mobilenet_acc)
print("MobileNetV2 Test Loss:", mobilenet_loss)

```

```
Epoch 1/2
48/48 _____ 60s 1s/step - accuracy: 0.9236 - loss:
0.1861 - val_accuracy: 0.9724 - val_loss: 0.0730
Epoch 2/2
48/48 _____ 53s 1s/step - accuracy: 0.9824 - loss:
0.0576 - val_accuracy: 0.9908 - val_loss: 0.0204
11/11 _____ 6s 564ms/step - accuracy: 0.9970 - loss:
0.0211
MobileNetV2 Test Accuracy: 0.9969788789749146
MobileNetV2 Test Loss: 0.021093038842082024
```

```
import pandas as pd

results = pd.DataFrame({
    "Model": ["EfficientNetB0", "ResNet50", "MobileNetV2"],
    "Test Accuracy": [efficient_acc, resnet_acc, mobilenet_acc],
    "Test Loss": [efficient_loss, resnet_loss, mobilenet_loss]
})

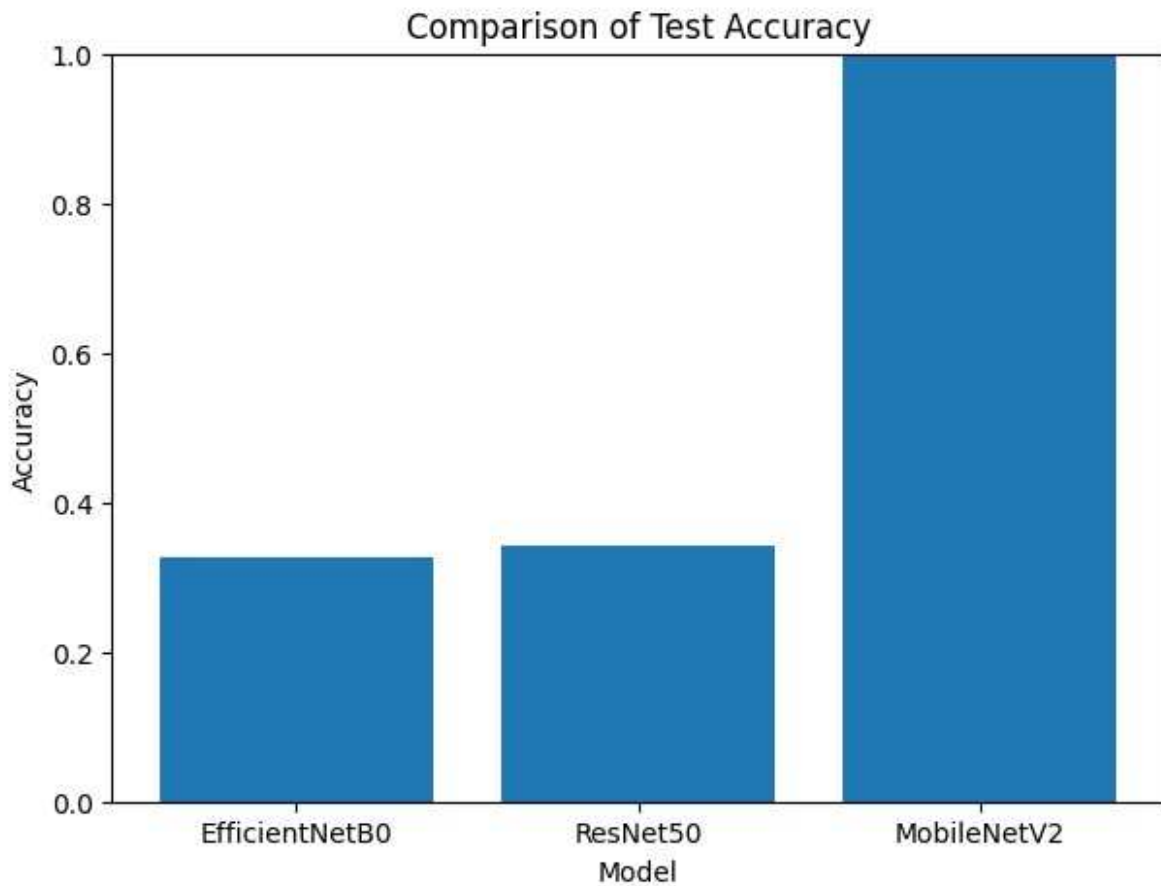
results

{"summary": "{\n  \"name\": \"results\",\n  \"rows\": 3,\n  \"fields\":\n  [\n    {\n      \"column\": \"Model\",\n      \"properties\": {\n        \"dtype\": \"string\",\n        \"num_unique_values\": 3,\n        \"samples\": [\n          \"EfficientNetB0\",\n          \"ResNet50\",\n          \"MobileNetV2\"\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Test Accuracy\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.38293971811334504,\n        \"min\": 0.3262839913368225,\n        \"max\": 0.9969788789749146,\n        \"num_unique_values\": 3,\n        \"samples\": [\n          0.3262839913368225,\n          0.3413897156715393,\n          0.9969788789749146\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    },\n    {\n      \"column\": \"Test Loss\",\n      \"properties\": {\n        \"dtype\": \"number\",\n        \"std\": 0.6206147571209156,\n        \"min\": 0.021093038842082024,\n        \"max\": 1.1021002531051636,\n        \"num_unique_values\": 3,\n        \"samples\": [\n          1.1021002531051636,\n          1.0898537635803223,\n          0.021093038842082024\n        ],\n        \"semantic_type\": \"\",\n        \"description\": \"\"\n      }\n    }\n  ]\n}", "type": "dataframe", "variable_name": "results"}

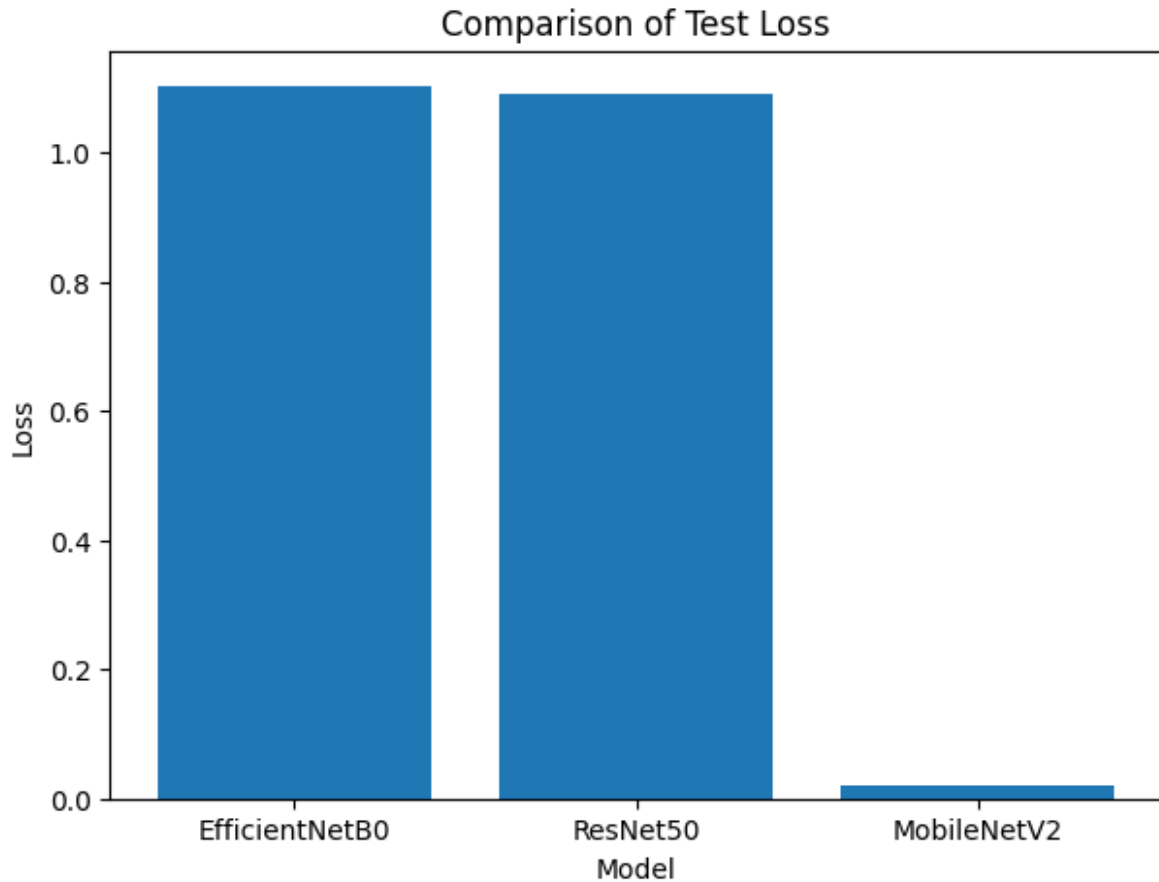
import matplotlib.pyplot as plt

plt.figure(figsize=(7,5))
plt.bar(results["Model"], results["Test Accuracy"])
plt.title("Comparison of Test Accuracy")
plt.xlabel("Model")
plt.ylabel("Accuracy")
```

```
plt.ylim(0,1)
plt.show()
```



```
plt.figure(figsize=(7,5))
plt.bar(results["Model"], results["Test Loss"])
plt.title("Comparison of Test Loss")
plt.xlabel("Model")
plt.ylabel("Loss")
plt.show()
```



```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
import matplotlib.pyplot as plt
import numpy as np

from tensorflow.keras.preprocessing.image import ImageDataGenerator

test_datagen = ImageDataGenerator(rescale=1./255)

test_data_224 = test_datagen.flow_from_directory(
    "/content/data/test",
    target_size=(224,224),
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)

Found 331 images belonging to 3 classes.
```

EfficientNetB0 Confusion Matrix

```
import numpy as np
from sklearn.metrics import confusion_matrix, ConfusionMatrixDisplay
import matplotlib.pyplot as plt

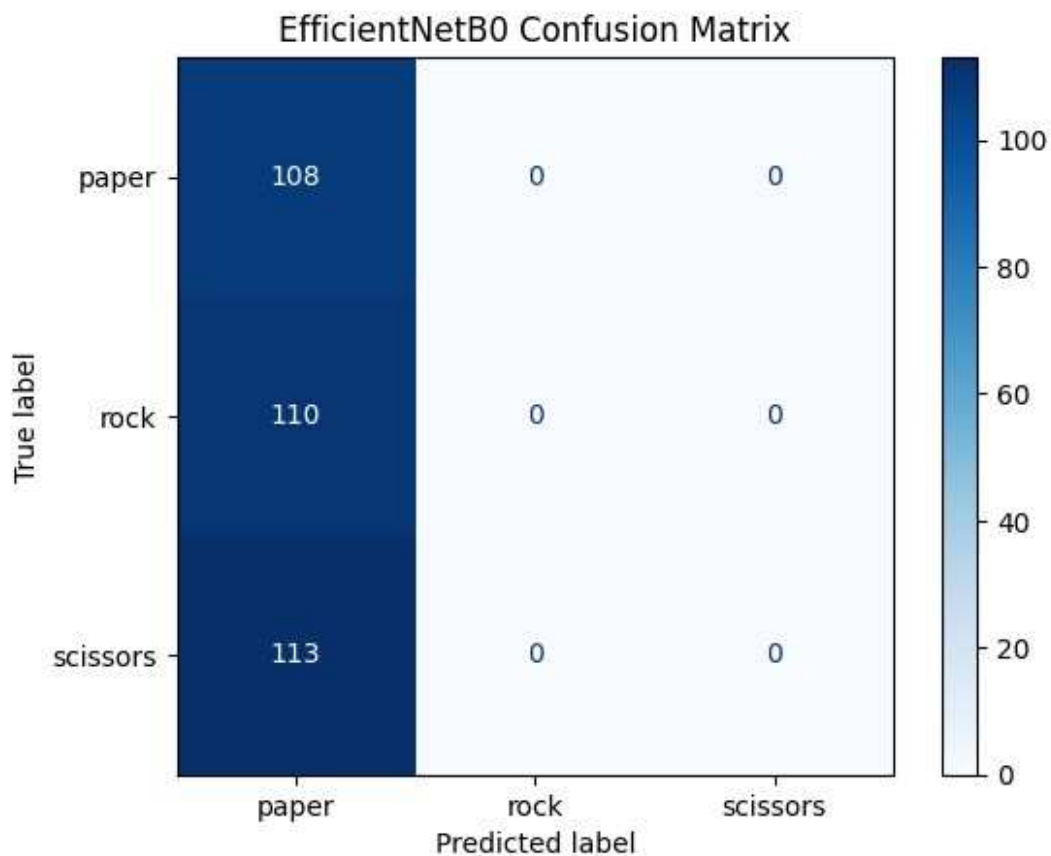
pred = efficient_model.predict(test_data_224)
pred = np.argmax(pred, axis=1)

cm = confusion_matrix(test_data_224.classes, pred)

disp = ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=list(test_data_224.class_indices.keys())
)

disp.plot(cmap="Blues")
plt.title("EfficientNetB0 Confusion Matrix")
plt.show()
```

11/11 ————— 33s 3s/step



ResNet50 Confusion Matrix

```

from sklearn.metrics import (
    confusion_matrix,
    ConfusionMatrixDisplay,
    classification_report
)

test_datagen = ImageDataGenerator(rescale=1./255)

test_data_224 = test_datagen.flow_from_directory(
    "/content/data/test",
    target_size=(224,224),
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)

resnet_pred = resnet_model.predict(test_data_224)
resnet_pred = np.argmax(resnet_pred, axis=1)

resnet_cm = confusion_matrix(test_data_224.classes, resnet_pred)

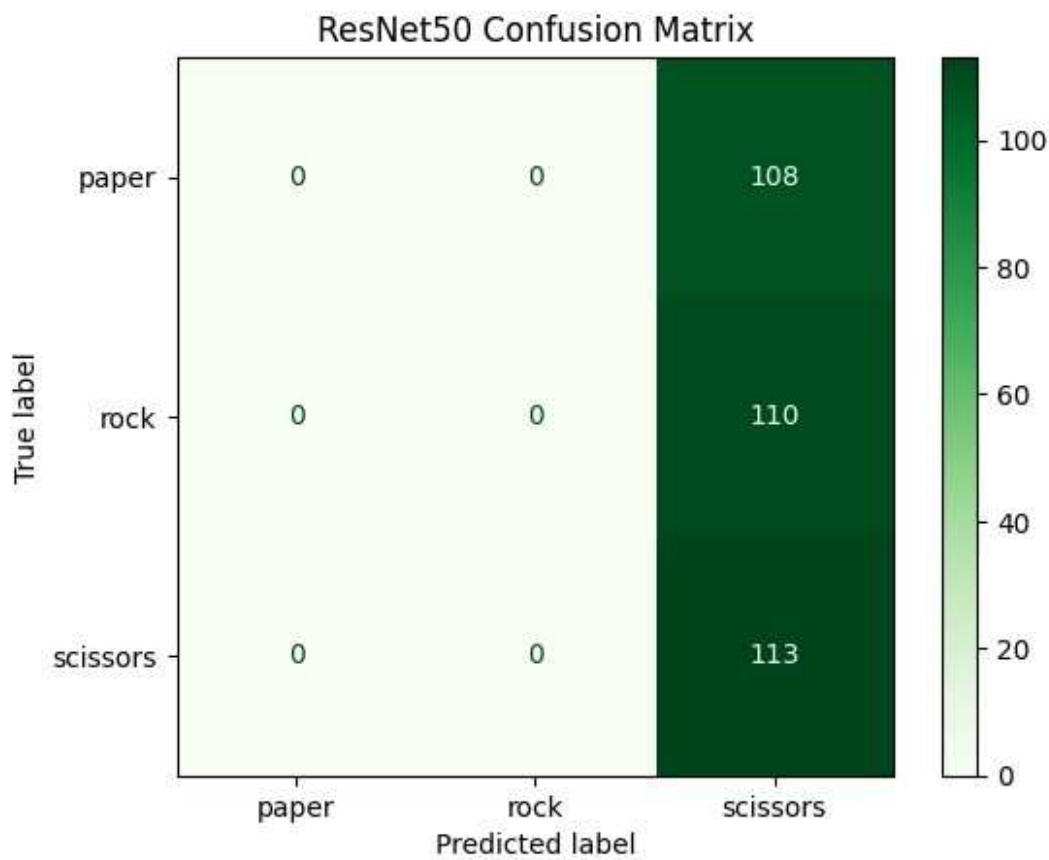
disp = ConfusionMatrixDisplay(
    confusion_matrix=resnet_cm,
    display_labels=list(test_data_224.class_indices.keys())
)

disp.plot(cmap="Greens")
plt.title("ResNet50 Confusion Matrix")
plt.show()

print("ResNet50 Classification Report")
print(classification_report(
    test_data_224.classes,
    resnet_pred,
    target_names=list(test_data_224.class_indices.keys())
))

```

Found 331 images belonging to 3 classes.
 11/11 69s 6s/step



ResNet50 Classification Report

	precision	recall	f1-score	support
paper	0.00	0.00	0.00	108
rock	0.00	0.00	0.00	110
scissors	0.34	1.00	0.51	113
accuracy			0.34	331
macro avg	0.11	0.33	0.17	331
weighted avg	0.12	0.34	0.17	331

```

/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is", len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",

```

```
len(result))
/usr/local/lib/python3.12/dist-packages/sklearn/metrics/_classification.py:1565: UndefinedMetricWarning: Precision is ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` parameter to control this behavior.
  _warn_prf(average, modifier, f"{metric.capitalize()} is",
len(result))
```

MobileNetV2 Confusion Matrix

```
test_data_160 = test_datagen.flow_from_directory(
    "/content/data/test",
    target_size=(160,160),
    batch_size=32,
    class_mode="categorical",
    shuffle=False
)

mobilenet_pred = mobilenet_model.predict(test_data_160)
mobilenet_pred = np.argmax(mobilenet_pred, axis=1)

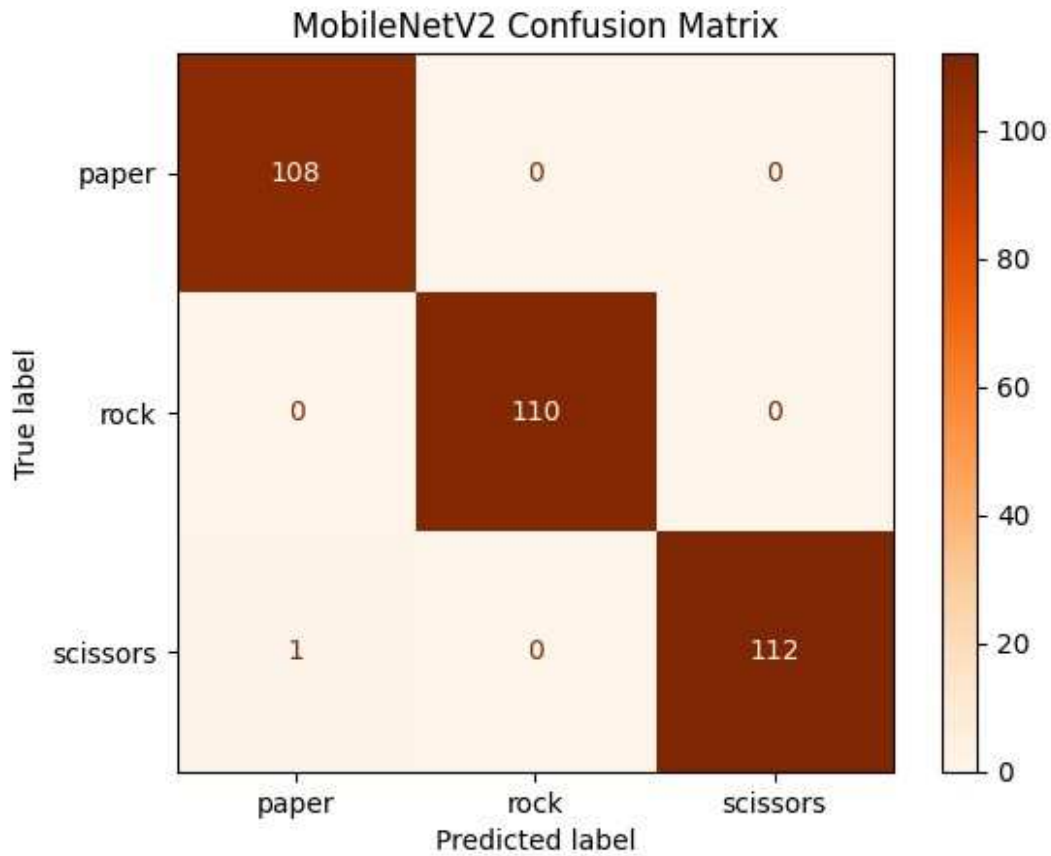
mobilenet_cm = confusion_matrix(test_data_160.classes, mobilenet_pred)

disp = ConfusionMatrixDisplay(
    confusion_matrix=mobilenet_cm,
    display_labels=list(test_data_160.class_indices.keys())
)

disp.plot(cmap="Oranges")
plt.title("MobileNetV2 Confusion Matrix")
plt.show()

print("MobileNetV2 Classification Report")
print(classification_report(
    test_data_160.classes,
    mobilenet_pred,
    target_names=list(test_data_160.class_indices.keys())
))

Found 331 images belonging to 3 classes.
11/11                      11s 704ms/step
```



MobileNetV2 Classification Report

	precision	recall	f1-score	support
paper	0.99	1.00	1.00	108
rock	1.00	1.00	1.00	110
scissors	1.00	0.99	1.00	113
accuracy			1.00	331
macro avg	1.00	1.00	1.00	331
weighted avg	1.00	1.00	1.00	331

```
import numpy as np
import matplotlib.pyplot as plt

models = ['EfficientNetB0', 'ResNet50', 'MobileNetV2']

accuracy = [efficient_acc, resnet_acc, mobilenet_acc]
loss = [efficient_loss, resnet_loss, mobilenet_loss]

x = np.arange(len(models))
width = 0.35

plt.figure(figsize=(8,5))
```

```
plt.bar(x - width/2, accuracy, width, label='Accuracy')
plt.bar(x + width/2, loss, width, label='Loss')

plt.xticks(x, models)
plt.ylabel('Score')
plt.title('Model Performance Comparison')
plt.legend()

plt.show()
```

