

Dataset link: <https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia>

```
!pip install -q tensorflow seaborn scikit-learn
```

```
import os
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
from google.colab import drive
```

```
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import MobileNetV2, EfficientNetB0
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

```
from sklearn.metrics import (
    accuracy_score,
    precision_score,
    recall_score,
    f1_score,
    confusion_matrix,
    classification_report
)
```

```
drive.mount('/content/drive')
```

```
# DATASET PATH
```

```
dataset_path = "/content/drive/MyDrive/archive/chest_xray/chest_xray"
```

```
print("Dataset Exists:", os.path.exists(dataset_path))
print(os.listdir(dataset_path))
```

```
IMG_SIZE = (224,224)
BATCH_SIZE = 32
EPOCHS = 5
```

```

train_datagen = ImageDataGenerator(
    rescale=1./255,
    rotation_range=15,
    zoom_range=0.2,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(
    rescale=1./255
)

train_generator = train_datagen.flow_from_directory(
    os.path.join(dataset_path, 'train'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

val_generator = test_datagen.flow_from_directory(
    os.path.join(dataset_path, 'val'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical'
)

test_generator = test_datagen.flow_from_directory(
    os.path.join(dataset_path, 'test'),
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='categorical',
    shuffle=False
)

class_names = list(test_generator.class_indices.keys())

print("Classes:", class_names)

early_stop = EarlyStopping(
    monitor='val_accuracy',
    patience=2,
    restore_best_weights=True
)

results = []

models = {
    "MobileNetV2": MobileNetV2,

```

```

    "EfficientNetB0": EfficientNetB0
}

for model_name, model_class in models.items():

    print("\n")
    print("="*50)
    print("Training:", model_name)
    print("="*50)

    base_model = model_class(
        weights='imagenet',
        include_top=False,
        pooling='avg',
        input_shape=(224,224,3)
    )

    base_model.trainable = False

    model = Sequential([
        base_model,
        Dense(256, activation='relu'),
        Dropout(0.3),
        Dense(
            train_generator.num_classes,
            activation='softmax'
        )
    ])

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

    history = model.fit(
        train_generator,
        validation_data=val_generator,
        epochs=EPOCHS,
        callbacks=[early_stop]
    )

    plt.figure(figsize=(8,4))
    plt.plot(history.history['accuracy'])
    plt.plot(history.history['val_accuracy'])
    plt.title(f'{model_name} Accuracy')
    plt.xlabel('Epoch')
    plt.ylabel('Accuracy')
    plt.legend(['Train', 'Validation'])

```

```
plt.show()
```

```
plt.figure(figsize=(8,4))
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title(f'{model_name} Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'])
plt.show()
```

```
predictions = model.predict(test_generator)
```

```
y_pred = np.argmax(predictions, axis=1)
y_true = test_generator.classes
```

```
accuracy = accuracy_score(y_true, y_pred)
```

```
precision = precision_score(
    y_true,
    y_pred,
    average='weighted'
)
```

```
recall = recall_score(
    y_true,
    y_pred,
    average='weighted'
)
```

```
f1 = f1_score(
    y_true,
    y_pred,
    average='weighted'
)
```

```
print("\nAccuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)
```

```
print("\nClassification Report\n")
```

```
print(
    classification_report(
        y_true,
        y_pred,
        target_names=class_names
    )
)
```

```

    )
)

cm = confusion_matrix(
    y_true,
    y_pred
)

plt.figure(figsize=(6,5))

sns.heatmap(
    cm,
    annot=True,
    fmt='d',
    cmap='Blues',
    xticklabels=class_names,
    yticklabels=class_names
)

plt.title(
    f'{model_name} Confusion Matrix'
)

plt.xlabel("Predicted")
plt.ylabel("Actual")

plt.show()

results.append([
    model_name,
    accuracy,
    precision,
    recall,
    f1
])

results_df = pd.DataFrame(
    results,
    columns=[
        'Model',
        'Accuracy',
        'Precision',
        'Recall',
        'F1 Score'
    ]
)

print(results_df)

```

```

results_df.set_index('Model').plot(
    kind='bar',
    figsize=(10,5)
)

plt.title("Model Comparison")
plt.ylabel("Score")
plt.grid(True)

plt.show()

results_df.to_csv(
    "/content/drive/MyDrive/results.csv",
    index=False
)

print("Results saved to Google Drive")

```

Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).

Dataset Exists: True

['.DS_Store', 'train', 'val', 'test']

Found 5216 images belonging to 2 classes.

Found 16 images belonging to 2 classes.

Found 624 images belonging to 2 classes.

Classes: ['NORMAL', 'PNEUMONIA']

=====

Training: MobileNetV2

=====

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet_v2/mobilenet_v2_weights_tf_dim_ordering_tf_kernels_1.0_9406464/9406464 _____ 0s 0us/step

Epoch 1/5

163/163 _____ 721s 4s/step - accuracy: 0.9112 - loss: 0.2129 - val_accuracy: 0.7500 - val_loss: 0.3307

Epoch 2/5

163/163 _____ 354s 2s/step - accuracy: 0.9427 - loss: 0.1524 - val_accuracy: 0.8125 - val_loss: 0.3186

Epoch 3/5

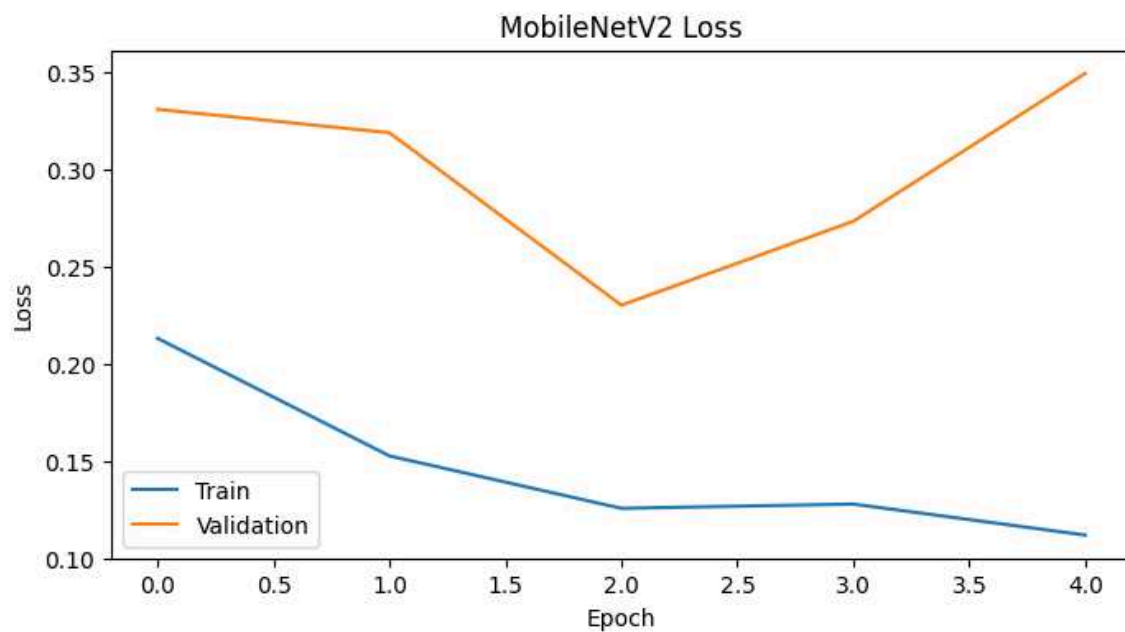
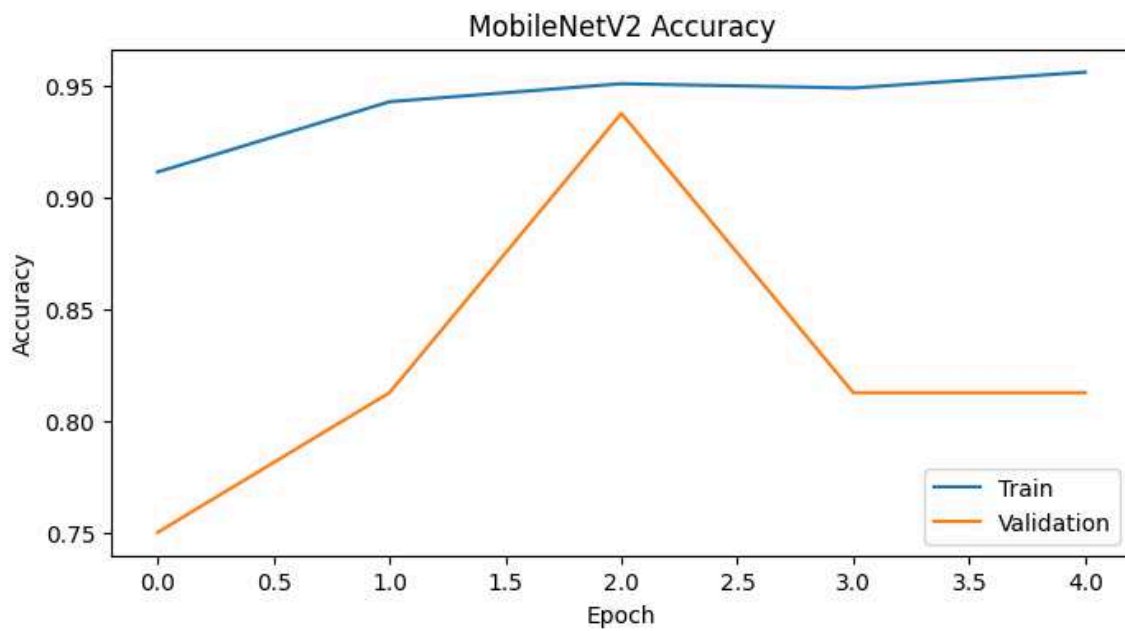
163/163 _____ 397s 2s/step - accuracy: 0.9507 - loss: 0.1255 - val_accuracy: 0.9375 - val_loss: 0.2299

Epoch 4/5

163/163 _____ 353s 2s/step - accuracy: 0.9488 - loss: 0.1276 - val_accuracy: 0.8125 - val_loss: 0.2731

Epoch 5/5

163/163 _____ 339s 2s/step - accuracy: 0.9559 - loss: 0.1117 - val_accuracy: 0.8125 - val_loss: 0.3492



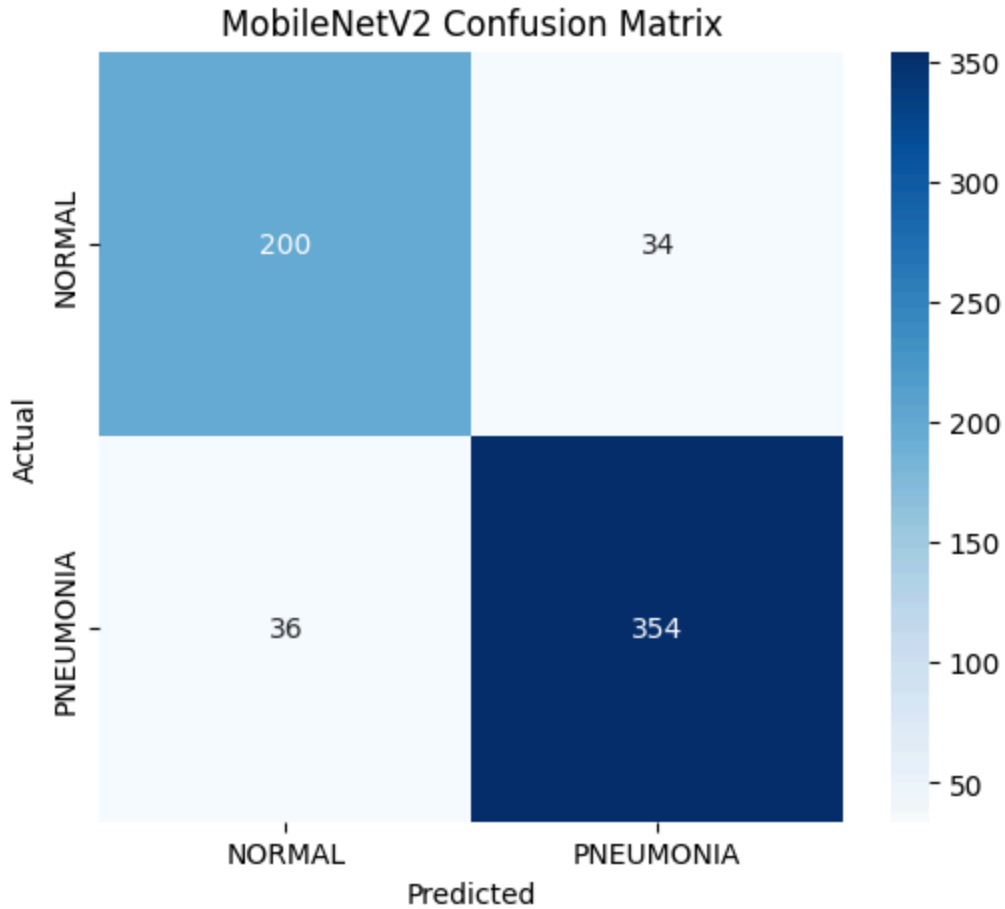
20/20 ————— 201s 10s/step

Accuracy: 0.8878205128205128
Precision: 0.8880285689323781
Recall: 0.8878205128205128
F1 Score: 0.8879150030082591

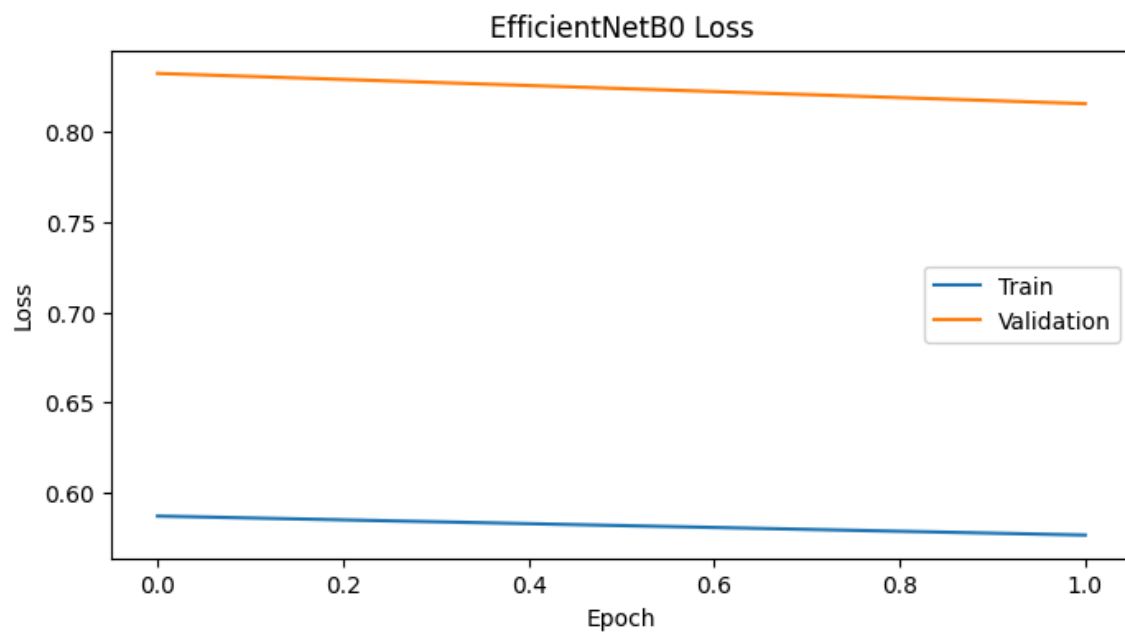
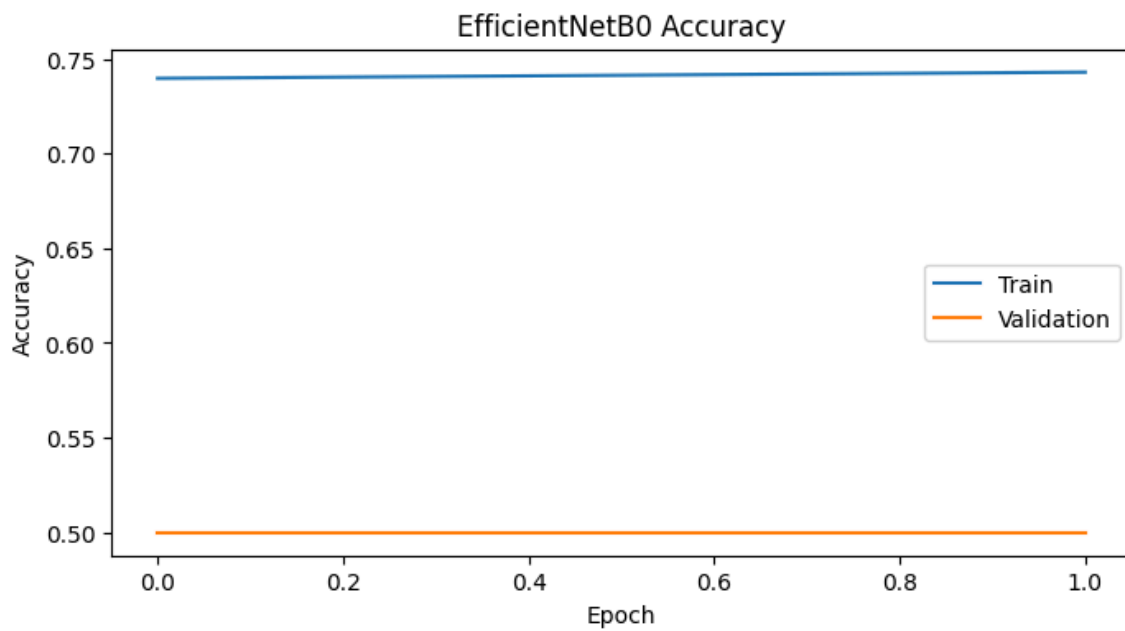
Classification Report

	precision	recall	f1-score	support
NORMAL	0.85	0.85	0.85	234
PNEUMONIA	0.91	0.91	0.91	390
accuracy			0.89	624
macro avg	0.88	0.88	0.88	624

weighted avg 0.89 0.89 0.89 624



```
=====
Training: EfficientNetB0
=====
Downloading data from https://storage.googleapis.com/keras-
applications/efficientnetb0_notop.h5
16705208/16705208 ----- 0s 0us/step
Epoch 1/5
163/163 ----- 548s 3s/step - accuracy: 0.7396 - loss: 0.5870 -
val_accuracy: 0.5000 - val_loss: 0.8326
Epoch 2/5
163/163 ----- 553s 3s/step - accuracy: 0.7429 - loss: 0.5765 -
val_accuracy: 0.5000 - val_loss: 0.8158
```



20/20 ————— 65s 3s/step

Accuracy: 0.625

Precision: 0.390625

Recall: 0.625

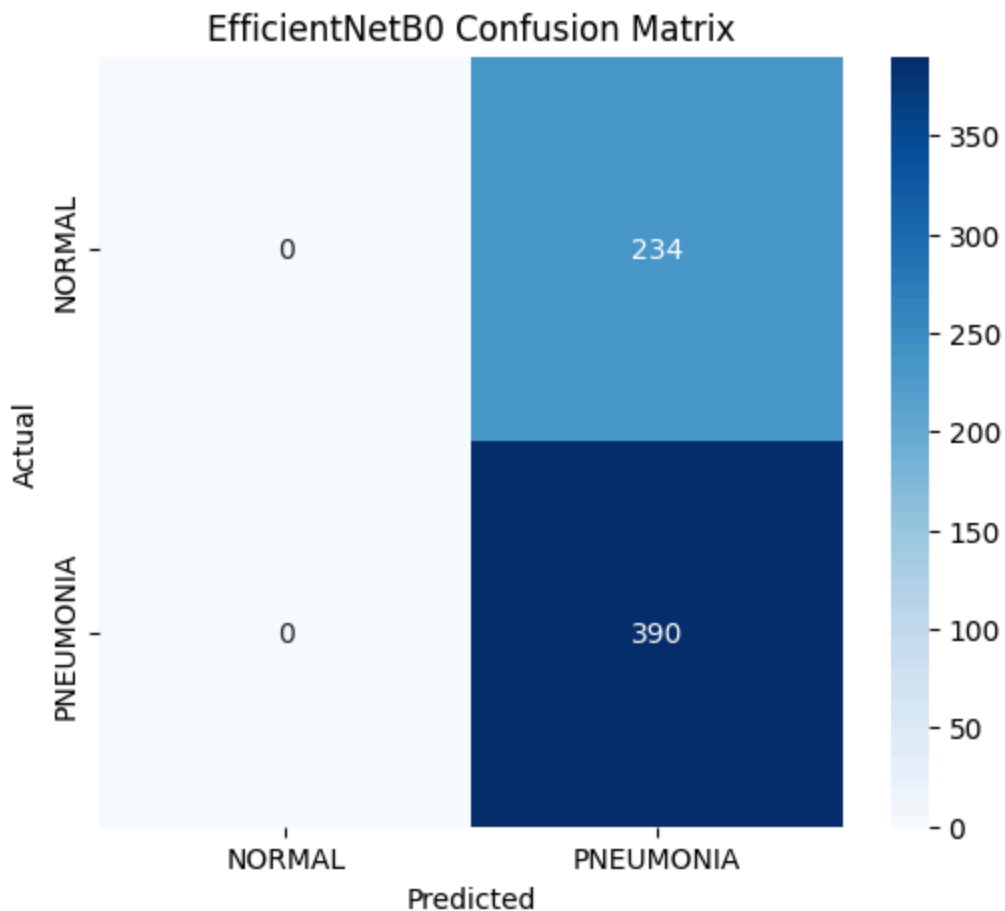
F1 Score: 0.4807692307692308

Classification Report

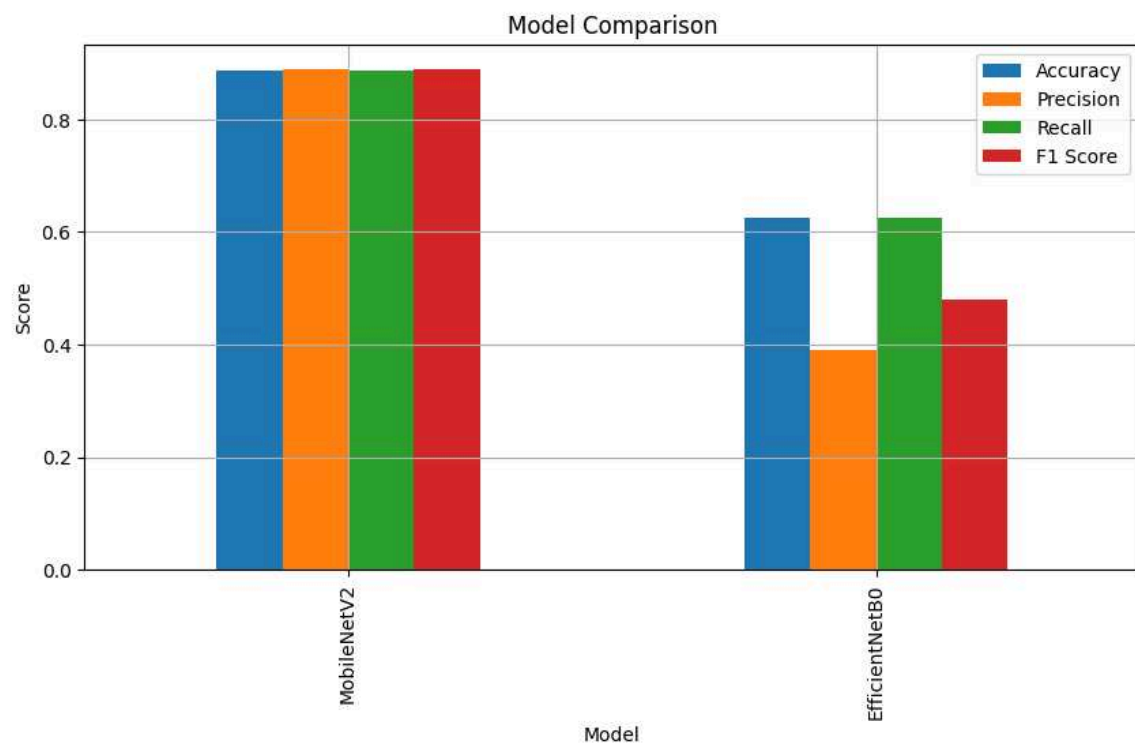
	precision	recall	f1-score	support
NORMAL	0.00	0.00	0.00	234
PNEUMONIA	0.62	1.00	0.77	390
accuracy			0.62	624
macro avg	0.31	0.50	0.38	624

weighted avg 0.39 0.62 0.48 624

```
/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))  
/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))
```



	Model	Accuracy	Precision	Recall	F1 Score
0	MobileNetV2	0.887821	0.888029	0.887821	0.887915
1	EfficientNetB0	0.625000	0.390625	0.625000	0.480769



Results saved to Google Drive