

MobileNetV2

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

import numpy as np
import matplotlib.pyplot as plt

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

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

IMG_SIZE = (224,224)
BATCH_SIZE = 64
EPOCHS = 5

train_dir = "/content/drive/MyDrive/archive/chest_xray/train"
test_dir = "/content/drive/MyDrive/archive/chest_xray/test"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.15,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

val_gen = train_datagen.flow_from_directory(
    train_dir,
```

```

        target_size=IMG_SIZE,
        batch_size=BATCH_SIZE,
        class_mode='binary',
        subset='validation'
    )

test_gen = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

base_model = MobileNetV2(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3),
    pooling='avg'
)

base_model.trainable = False

model = Sequential([
    base_model,
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])

model.compile(
    optimizer=Adam(0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    callbacks=[early_stop]
)

loss, acc = model.evaluate(test_gen)

print(f"\nTest Accuracy: {acc:.4f}")

```

```

y_pred = model.predict(test_gen)
y_pred = (y_pred > 0.5).astype(int)

print("\nClassification Report")
print(classification_report(
    test_gen.classes,
    y_pred,
    target_names=['NORMAL', 'PNEUMONIA']
))

cm = confusion_matrix(test_gen.classes, y_pred)

ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=['NORMAL', 'PNEUMONIA']
).plot()

plt.show()

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.legend(['Train', 'Validation'])

plt.subplot(1,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.legend(['Train', 'Validation'])

plt.show()

model.save("MobileNetV2_Pneumonia.h5")
print("Model Saved Successfully")

```

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

Found 4173 images belonging to 2 classes.

Found 1043 images belonging to 2 classes.

Found 624 images belonging to 2 classes.

Epoch 1/5

66/66 ————— 391s 6s/step - accuracy: 0.9073 - loss: 0.2257 - val_accuracy: 0.9358 - val_loss: 0.1594

Epoch 2/5

66/66 ————— 387s 6s/step - accuracy: 0.9473 - loss: 0.1430 - val_accuracy: 0.9367 - val_loss: 0.1390

Epoch 3/5

66/66 ————— 393s 6s/step - accuracy: 0.9561 - loss: 0.1185 - val_accuracy: 0.9377 - val_loss: 0.1518

Epoch 4/5

66/66 ————— 375s 6s/step - accuracy: 0.9559 - loss: 0.1078 -
val_accuracy: 0.9597 - val_loss: 0.1133

Epoch 5/5

66/66 ————— 360s 5s/step - accuracy: 0.9554 - loss: 0.1085 -
val_accuracy: 0.9703 - val_loss: 0.0869

10/10 ————— 150s 16s/step - accuracy: 0.8798 - loss: 0.2986

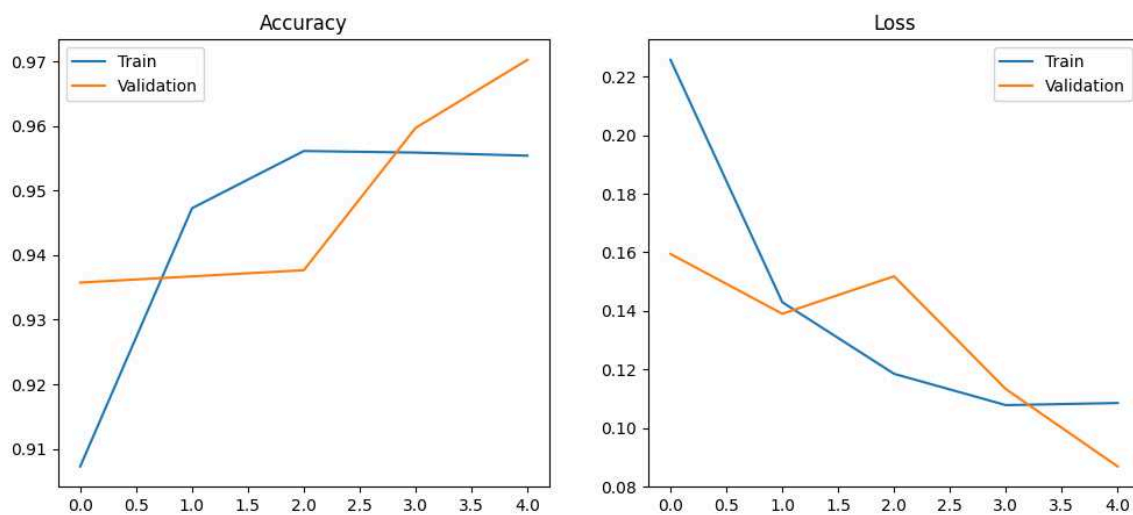
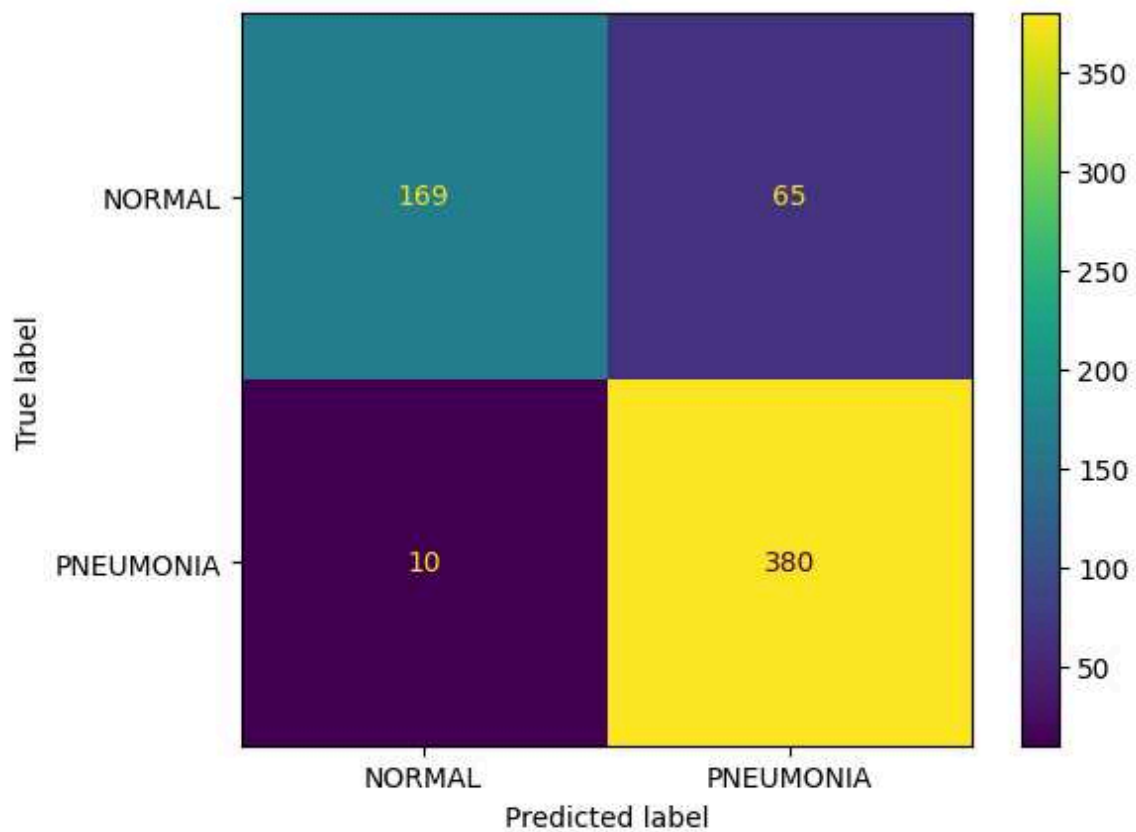
Test Accuracy: 0.8798

WARNING:tensorflow:5 out of the last 7 calls to <function TensorFlowTrainer.make_predict_function.<locals>.one_step_on_data_distributed at 0x7dbadc4c6d40> triggered tf.function retracing. Tracing is expensive and the excessive number of tracings could be due to (1) creating @tf.function repeatedly in a loop, (2) passing tensors with different shapes, (3) passing Python objects instead of tensors. For (1), please define your @tf.function outside of the loop. For (2), @tf.function has reduce_retracing=True option that can avoid unnecessary retracing. For (3), please refer to https://www.tensorflow.org/guide/function#controlling_retracing and https://www.tensorflow.org/api_docs/python/tf/function for more details.

10/10 ————— 39s 3s/step

Classification Report

	precision	recall	f1-score	support
NORMAL	0.94	0.72	0.82	234
PNEUMONIA	0.85	0.97	0.91	390
accuracy			0.88	624
macro avg	0.90	0.85	0.86	624
weighted avg	0.89	0.88	0.88	624



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')`.

Model Saved Successfully

EfficientNetB0

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

```
import numpy as np
```

```

import matplotlib.pyplot as plt

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

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

IMG_SIZE = (224, 224)
BATCH_SIZE = 64
EPOCHS = 5

train_dir = "/content/drive/MyDrive/archive/chest_xray/train"
test_dir = "/content/drive/MyDrive/archive/chest_xray/test"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.15,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(
    rescale=1./255
)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training'
)

val_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation'
)

```

```
test_gen = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)
```

```
base_model = EfficientNetB0(
    weights='imagenet',
    include_top=False,
    input_shape=(224,224,3),
    pooling='avg'
)
```

```
base_model.trainable = False
```

```
model = Sequential([
    base_model,
    Dense(128, activation='relu'),
    Dropout(0.3),
    Dense(1, activation='sigmoid')
])
```

```
model.compile(
    optimizer=Adam(learning_rate=0.001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)
```

```
early_stop = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)
```

```
history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    callbacks=[early_stop]
)
```

```
loss, acc = model.evaluate(test_gen)
```

```

print(f"\nTest Accuracy: {acc:.4f}")

y_pred = model.predict(test_gen)
y_pred = (y_pred > 0.5).astype(int).flatten()

print("\nClassification Report\n")

print(
    classification_report(
        test_gen.classes,
        y_pred,
        target_names=[ 'NORMAL', 'PNEUMONIA' ]
    )
)

cm = confusion_matrix(
    test_gen.classes,
    y_pred
)

ConfusionMatrixDisplay(
    confusion_matrix=cm,
    display_labels=[ 'NORMAL', 'PNEUMONIA' ]
).plot(cmap='Blues')

plt.show()

plt.figure(figsize=(12,5))

plt.subplot(1,2,1)
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend(['Train', 'Validation'])

plt.subplot(1,2,2)
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend(['Train', 'Validation'])

plt.show()

```

```
model.save(
    "/content/drive/MyDrive/EfficientNetB0.keras"
)
```

```
print("\nModel Saved Successfully")
```

```
print("\nDataset Statistics")
print("Train Images      :", train_gen.samples)
print("Validation Images :", val_gen.samples)
print("Test Images       :", test_gen.samples)
```

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

Found 4173 images belonging to 2 classes.

Found 1043 images belonging to 2 classes.

Found 624 images belonging to 2 classes.

Epoch 1/5

66/66 ————— 611s 9s/step - accuracy: 0.7352 - loss: 0.5919 -
val_accuracy: 0.7430 - val_loss: 0.5734

Epoch 2/5

66/66 ————— 601s 9s/step - accuracy: 0.7429 - loss: 0.5775 -
val_accuracy: 0.7430 - val_loss: 0.5745

Epoch 3/5

66/66 ————— 587s 9s/step - accuracy: 0.7429 - loss: 0.5789 -
val_accuracy: 0.7430 - val_loss: 0.5698

Epoch 4/5

66/66 ————— 553s 8s/step - accuracy: 0.7429 - loss: 0.5769 -
val_accuracy: 0.7430 - val_loss: 0.5705

Epoch 5/5

66/66 ————— 522s 8s/step - accuracy: 0.7429 - loss: 0.5811 -
val_accuracy: 0.7430 - val_loss: 0.5720

10/10 ————— 54s 5s/step - accuracy: 0.6250 - loss: 0.6919

Test Accuracy: 0.6250

10/10 ————— 63s 5s/step

Classification Report

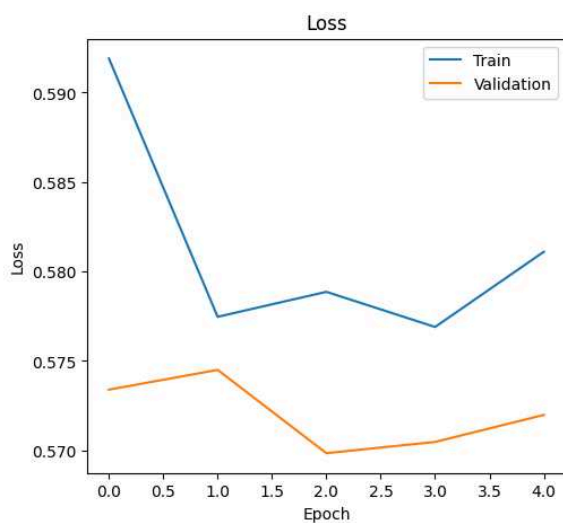
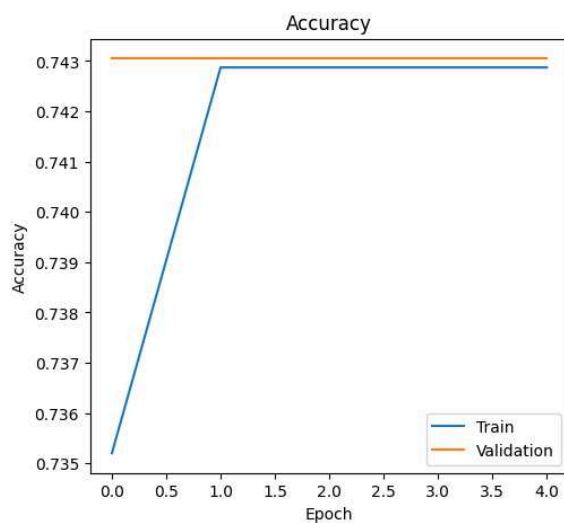
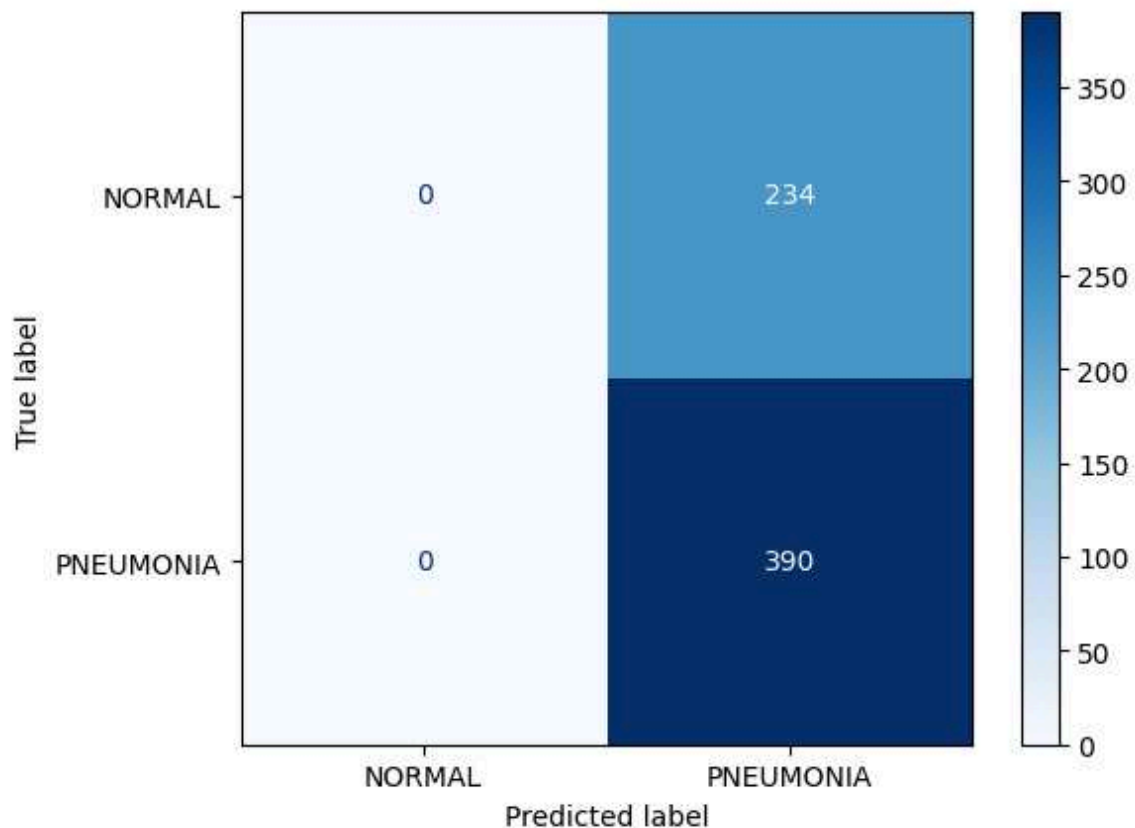
	precision	recall	f1-score	support
NORMAL	0.00	0.00	0.00	234
PNEUMONIA	0.62	1.00	0.77	390
accuracy			0.62	624
macro avg	0.31	0.50	0.38	624
weighted avg	0.39	0.62	0.48	624

/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))
```



Model Saved Successfully

Dataset Statistics

Train Images : 4173
Validation Images : 1043
Test Images : 624

ResNet50

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

import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import ResNet50
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay

IMG_SIZE = (224, 224)
BATCH_SIZE = 64
EPOCHS = 5

train_dir = "/content/drive/MyDrive/archive/chest_xray/train"
test_dir = "/content/drive/MyDrive/archive/chest_xray/test"

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.15,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
```

```

        subset='training',
        shuffle=True
    )

val_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation',
    shuffle=False
)

test_gen = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

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

base_model.trainable = False

model = Sequential([
    base_model,
    GlobalAveragePooling2D(),
    Dense(256, activation='relu'),
    Dropout(0.4),
    Dense(1, activation='sigmoid')
])

model.compile(
    optimizer=Adam(learning_rate=0.0001),
    loss='binary_crossentropy',
    metrics=['accuracy']
)

model.summary()

early_stop = EarlyStopping(
    monitor='val_loss',
    patience=3,
    restore_best_weights=True
)

```

```

history = model.fit(
    train_gen,
    validation_data=val_gen,
    epochs=EPOCHS,
    callbacks=[early_stop]
)

loss, acc = model.evaluate(test_gen)

print("\nTest Accuracy =", round(acc*100,2), "%")

pred_probs = model.predict(test_gen)

preds = (pred_probs > 0.5).astype(int)

print(classification_report(
    test_gen.classes,
    preds,
    target_names=list(test_gen.class_indices.keys())
))

cm = confusion_matrix(test_gen.classes, preds)

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

disp.plot(cmap="Blues")
plt.show()

plt.figure(figsize=(10,5))

plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')

plt.title("ResNet50 Accuracy")
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.legend()

plt.show()

model.save("ResNet50_Pneumonia.h5")

```

```
print("Model Saved Successfully")
```

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

Found 4173 images belonging to 2 classes.

Found 1043 images belonging to 2 classes.

Found 624 images belonging to 2 classes.

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/resnet/resnet50_weights_tf_dim_ordering_tf_kernels_notop.h5
94765736/94765736 _____ 1s 0us/step

Model: "sequential_9"

Layer (type)	Output Shape
resnet50 (Functional)	(None , 7 , 7 , 2048)
global_average_pooling2d_5 (GlobalAveragePooling2D)	(None , 2048)
dense_18 (Dense)	(None , 256)
dropout_9 (Dropout)	(None , 256)
dense_19 (Dense)	(None , 1)

Total params: [24,112,513](#) (91.98 MB)

Trainable params: [524,801](#) (2.00 MB)

Non-trainable params: [23,587,712](#) (89.98 MB)

Epoch 1/5

66/66 _____ 1041s 16s/step - accuracy: 0.7170 - loss: 0.6167 - val_accuracy: 0.7430 - val_loss: 0.5619

Epoch 2/5

66/66 _____ 1043s 16s/step - accuracy: 0.7383 - loss: 0.5780 - val_accuracy: 0.7430 - val_loss: 0.5467

Epoch 3/5

66/66 _____ 1060s 16s/step - accuracy: 0.7407 - loss: 0.5595 - val_accuracy: 0.7430 - val_loss: 0.5355

Epoch 4/5

66/66 _____ 1067s 16s/step - accuracy: 0.7443 - loss: 0.5429 - val_accuracy: 0.7430 - val_loss: 0.5237

Epoch 5/5

66/66 _____ 1093s 17s/step - accuracy: 0.7441 - loss: 0.5265 - val_accuracy: 0.7450 - val_loss: 0.5094

10/10 _____ 123s 12s/step - accuracy: 0.6250 - loss: 0.6261

Test Accuracy = 62.5 %

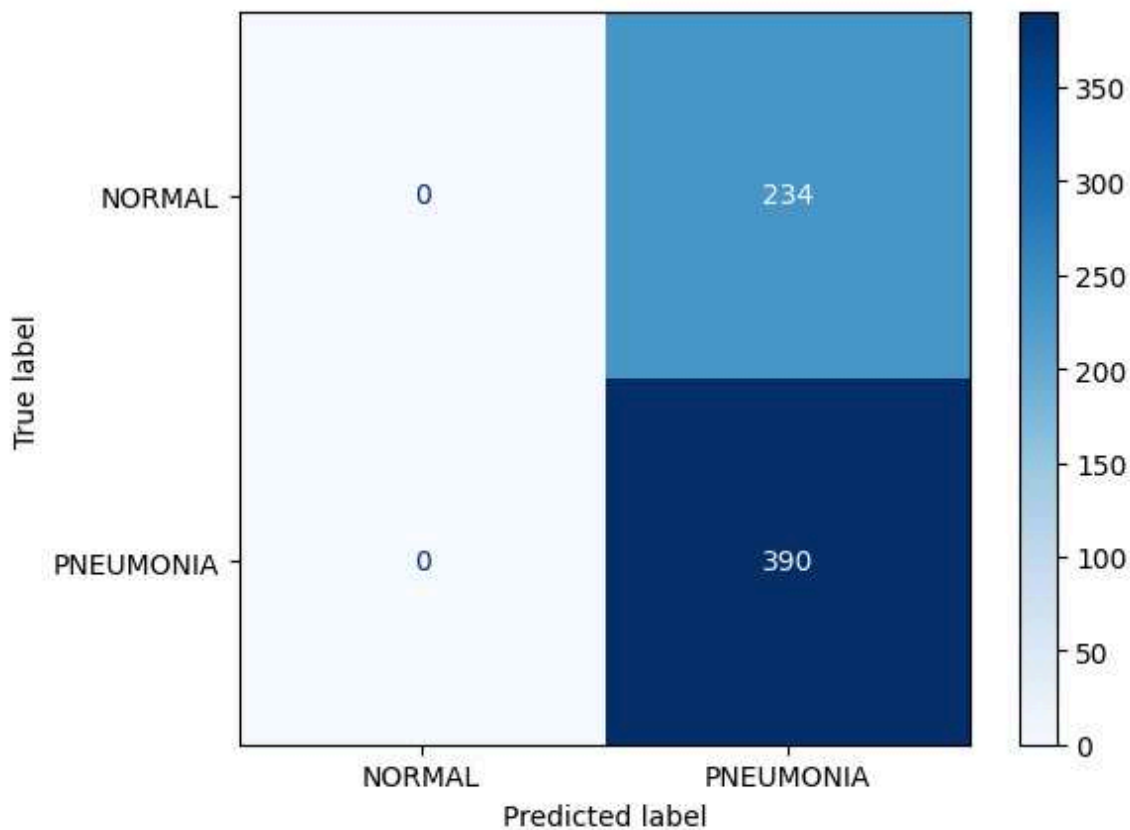
10/10 _____ 125s 12s/step

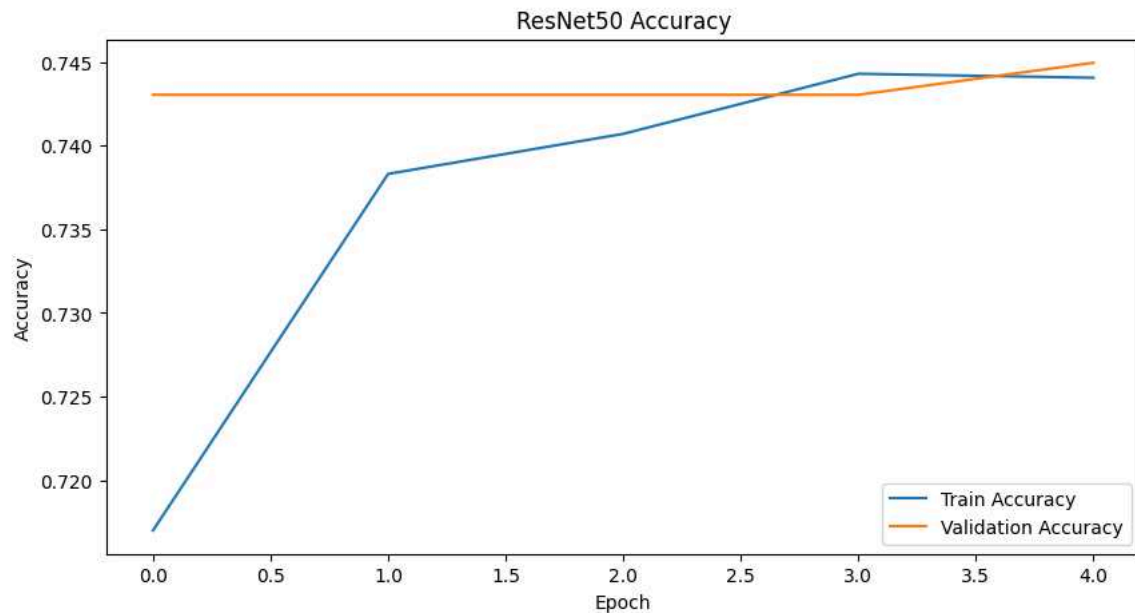
```

/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))

```

	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





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')`.

Model Saved Successfully

DenseNet121

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

import numpy as np
import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.applications import DenseNet121
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, GlobalAveragePooling2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.callbacks import EarlyStopping

from sklearn.metrics import classification_report
from sklearn.metrics import confusion_matrix
from sklearn.metrics import ConfusionMatrixDisplay
from sklearn.utils.class_weight import compute_class_weight

IMG_SIZE = (224,224)
BATCH_SIZE = 64
EPOCHS = 5

train_dir = "/content/drive/MyDrive/archive/chest_xray/train"
test_dir = "/content/drive/MyDrive/archive/chest_xray/test"
```

```

train_datagen = ImageDataGenerator(
    rescale=1./255,
    validation_split=0.2,
    rotation_range=15,
    zoom_range=0.15,
    horizontal_flip=True
)

test_datagen = ImageDataGenerator(rescale=1./255)

train_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training',
    shuffle=True
)

val_gen = train_datagen.flow_from_directory(
    train_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='validation',
    shuffle=False
)

test_gen = test_datagen.flow_from_directory(
    test_dir,
    target_size=IMG_SIZE,
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle=False
)

class_weights = compute_class_weight(
    class_weight='balanced',
    classes=np.unique(train_gen.classes),
    y=train_gen.classes
)

class_weights = dict(enumerate(class_weights))

print(class_weights)

base_model = DenseNet121(
    weights='imagenet',

```

```

        include_top=False,
        input_shape=(224,224,3)
    )

    base_model.trainable = False

    model = Sequential([
        base_model,
        GlobalAveragePooling2D(),
        Dense(256, activation='relu'),
        Dropout(0.4),
        Dense(1, activation='sigmoid')
    ])

    model.compile(
        optimizer=Adam(learning_rate=0.0001),
        loss='binary_crossentropy',
        metrics=['accuracy']
    )

    model.summary()

    early_stop = EarlyStopping(
        monitor='val_loss',
        patience=3,
        restore_best_weights=True
    )

    history = model.fit(
        train_gen,
        validation_data=val_gen,
        epochs=EPOCHS,
        class_weight=class_weights,
        callbacks=[early_stop]
    )

    loss, acc = model.evaluate(test_gen)

    print(f"\nTest Accuracy = {acc*100:.2f}%")

    pred_probs = model.predict(test_gen)

    preds = (pred_probs > 0.5).astype(int)

    print(classification_report(
        test_gen.classes,
        preds,
        target_names=list(test_gen.class_indices.keys())
    ))

```

```

))

cm = confusion_matrix(test_gen.classes, preds)

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

disp.plot(cmap='Blues')
plt.show()

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

plt.plot(history.history['accuracy'], label='Train')
plt.plot(history.history['val_accuracy'], label='Validation')

plt.title('DenseNet121 Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()

plt.show()

model.save('DenseNet121_Pneumonia.h5')

print("Model Saved Successfully")

```

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

Found 4173 images belonging to 2 classes.

Found 1043 images belonging to 2 classes.

Found 624 images belonging to 2 classes.

{0: np.float64(1.9445479962721341), 1: np.float64(0.6730645161290323)}

Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/densenet/densenet121_weights_tf_dim_ordering_tf_kernels_notop.h5

29084464/29084464 ————— 0s 0us/step

Model: "sequential_10"

Layer (type)	Output Shape
densenet121 (Functional)	(None , 7, 7, 1024)
global_average_pooling2d_6 (GlobalAveragePooling2D)	(None , 1024)
dense_20 (Dense)	(None , 256)

dropout_10 (Dropout)	(None, 256)
dense_21 (Dense)	(None, 1)

Total params: 7,300,161 (27.85 MB)

Trainable params: 262,657 (1.00 MB)

Non-trainable params: 7,037,504 (26.85 MB)

Epoch 1/5

66/66 ————— 1143s 17s/step - accuracy: 0.7251 - loss: 0.5290 -
val_accuracy: 0.8581 - val_loss: 0.3471

Epoch 2/5

66/66 ————— 1136s 17s/step - accuracy: 0.8773 - loss: 0.3040 -
val_accuracy: 0.8917 - val_loss: 0.2526

Epoch 3/5

66/66 ————— 1142s 17s/step - accuracy: 0.8936 - loss: 0.2648 -
val_accuracy: 0.8993 - val_loss: 0.2507

Epoch 4/5

66/66 ————— 1150s 17s/step - accuracy: 0.9029 - loss: 0.2409 -
val_accuracy: 0.8888 - val_loss: 0.2707

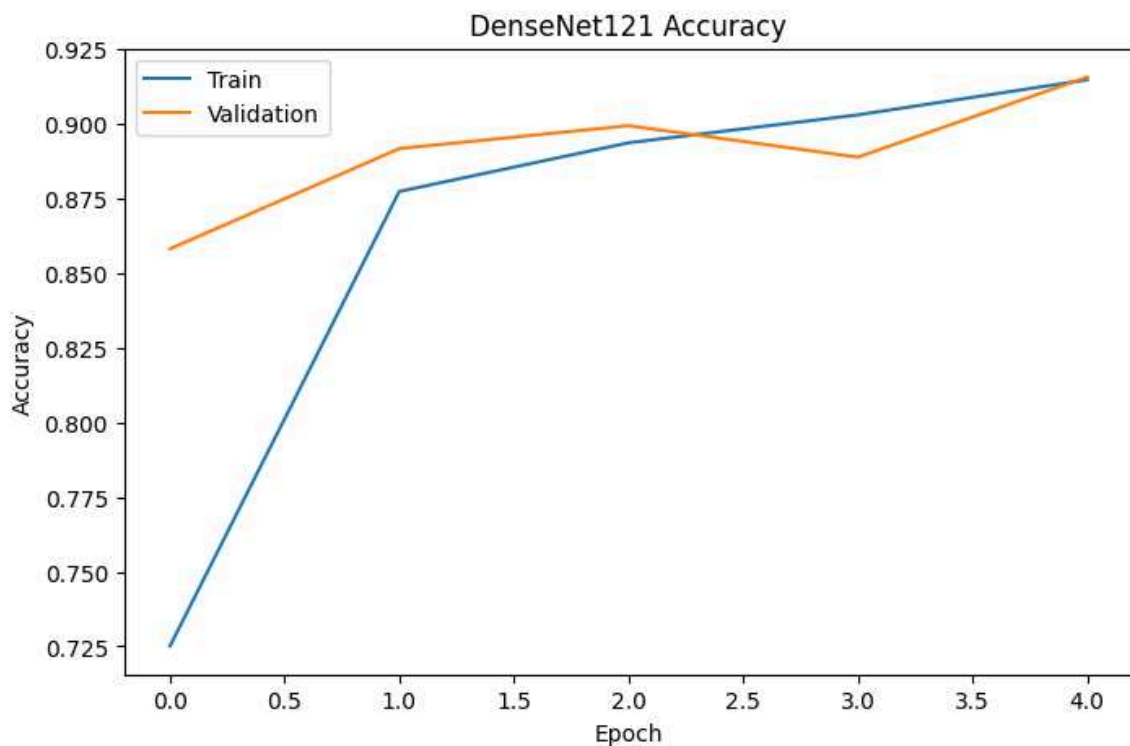
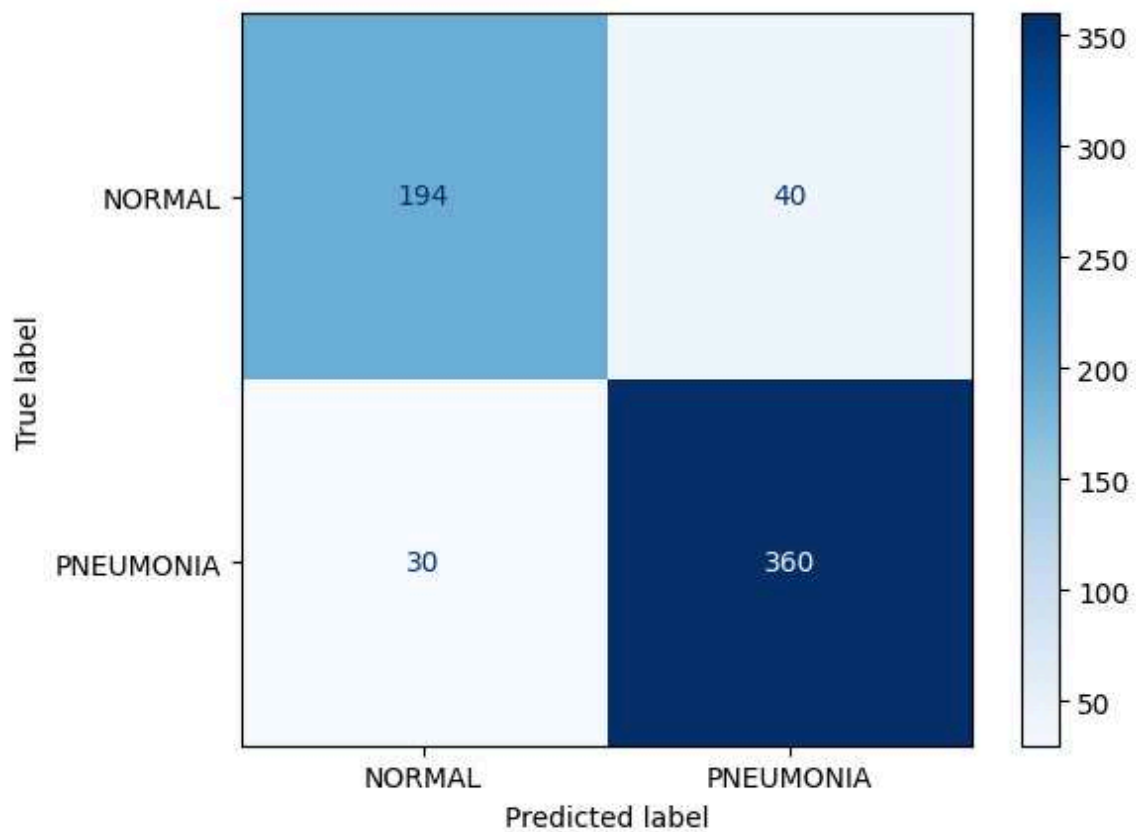
Epoch 5/5

66/66 ————— 1172s 18s/step - accuracy: 0.9147 - loss: 0.2144 -
val_accuracy: 0.9156 - val_loss: 0.2120

10/10 ————— 133s 13s/step - accuracy: 0.8878 - loss: 0.3007

Test Accuracy = 88.78%

10/10	—————	144s 14s/step			
	precision	recall	f1-score	support	
NORMAL	0.87	0.83	0.85	234	
PNEUMONIA	0.90	0.92	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	



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')`.

Model Saved Successfully

