Untitled

November 17, 2023

[1]:

```
import os
import shutil
import random

# Set the path to your dataset containing different folders
dataset_path = "dataset"

# Set the ratio for the validation set (e.g., 0.2 for 20% validation)
validation_ratio = 0.2

# Set the path where you want to create the train and validation directories
output_path = "BI68"

# Create the output directory if it doesn't exist
os.makedirs(output_path, exist_ok=True)

# List all folders in the dataset directory
folders = [f for f in os.listdir(dataset_path) if os.path.isdir(os.path.
 ↪join(dataset_path, f))]

# Create train and validation subdirectories
train_dir = os.path.join(output_path, "train")
val_dir = os.path.join(output_path, "val")
os.makedirs(train_dir, exist_ok=True)
os.makedirs(val_dir, exist_ok=True)

# Iterate through each folder and divide the data
for folder in folders:
    folder_path = os.path.join(dataset_path, folder)
    files = [f for f in os.listdir(folder_path) if os.path.isfile(os.path.
 ↪join(folder_path, f))]
    random.shuffle(files)

    # Calculate the split point based on the validation ratio
    split_point = int(len(files) * validation_ratio)

    # Split files into train and validation sets
    train_files = files[split_point:]
    val_files = files[:split_point]

    # Copy files to train directory
    for file in train_files:
        src_path = os.path.join(folder_path, file)
        dest_path = os.path.join(train_dir, folder, file)
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(src_path, dest_path)

    # Copy files to validation directory
    for file in val_files:
        src_path = os.path.join(folder_path, file)
        dest_path = os.path.join(val_dir, folder, file)
        os.makedirs(os.path.dirname(dest_path), exist_ok=True)
        shutil.copy(src_path, dest_path)

print("Dataset division into train and validation sets is complete.")
```

```
Dataset division into train and validation sets is complete.
```

[20]:

```
import tensorflow as tf
from tensorflow.keras import layers, models
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import matplotlib.pyplot as plt
```

[8]:

```
train_data_dir = 'BI68/train'
test_data_dir = 'BI68/val'
```

[11]:

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   shear_range=0.2,
                                   zoom_range=0.2,
                                   horizontal_flip=True)

test_datagen = ImageDataGenerator(rescale=1./255)

# Create generators for training and testing datasets
train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)

test_generator = test_datagen.flow_from_directory(
    test_data_dir,
    target_size=(150, 150),
    batch_size=32,
    class_mode='binary'
)
```

```
Found 500 images belonging to 2 classes.
Found 124 images belonging to 2 classes.
```

[12]:

```
model = models.Sequential()
model.add(layers.Conv2D(32, (3, 3), activation='relu', input_shape=(150, 150,␣
 ↪3)))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(64, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Conv2D(128, (3, 3), activation='relu'))
model.add(layers.MaxPooling2D((2, 2)))
model.add(layers.Flatten())
model.add(layers.Dense(512, activation='relu'))
model.add(layers.Dense(1, activation='sigmoid'))
```

[13]:

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

[14]:

```
model.fit(train_generator, epochs=10, validation_data=test_generator)
```

```
Epoch 1/10
16/16 [==============================] - 45s 3s/step - loss: 0.9821 - accuracy:
0.5600 - val_loss: 0.6712 - val_accuracy: 0.6290
Epoch 2/10
16/16 [==============================] - 32s 2s/step - loss: 0.6616 - accuracy:
0.6420 - val_loss: 0.6373 - val_accuracy: 0.6290
Epoch 3/10
16/16 [==============================] - 34s 2s/step - loss: 0.6158 - accuracy:
0.6960 - val_loss: 0.5146 - val_accuracy: 0.8145
Epoch 4/10
16/16 [==============================] - 36s 2s/step - loss: 0.5215 - accuracy:
0.7520 - val_loss: 0.3234 - val_accuracy: 0.8548
Epoch 5/10
16/16 [==============================] - 34s 2s/step - loss: 0.4249 - accuracy:
0.8180 - val_loss: 0.2637 - val_accuracy: 0.8790
Epoch 6/10
16/16 [==============================] - 35s 2s/step - loss: 0.3699 - accuracy:
0.8320 - val_loss: 0.2235 - val_accuracy: 0.9274
Epoch 7/10
16/16 [==============================] - 36s 2s/step - loss: 0.3456 - accuracy:
0.8680 - val_loss: 0.2879 - val_accuracy: 0.8952
Epoch 8/10
16/16 [==============================] - 33s 2s/step - loss: 0.3830 - accuracy:
0.8400 - val_loss: 0.2204 - val_accuracy: 0.8871
```

```
Epoch 9/10
16/16 [==============================] - 34s 2s/step - loss: 0.3105 - accuracy:
0.8660 - val_loss: 0.1900 - val_accuracy: 0.9194
Epoch 10/10
16/16 [==============================] - 33s 2s/step - loss: 0.3185 - accuracy:
0.8760 - val_loss: 0.1719 - val_accuracy: 0.9355
```

[14]:
```
<keras.src.callbacks.History at 0x2f63982d150>
```

[15]:
```python
test_loss, test_acc = model.evaluate(test_generator)
print(f'Test accuracy: {test_acc}')
```

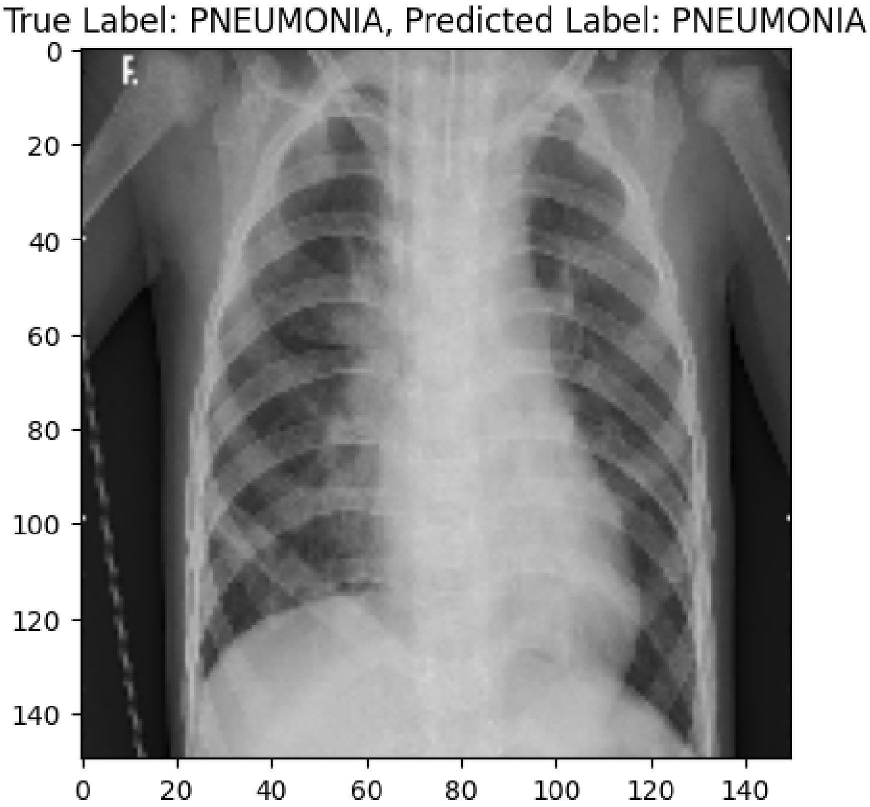
```
4/4 [==============================] - 4s 876ms/step - loss: 0.1719 - accuracy:
0.9355
Test accuracy: 0.9354838728904724
```

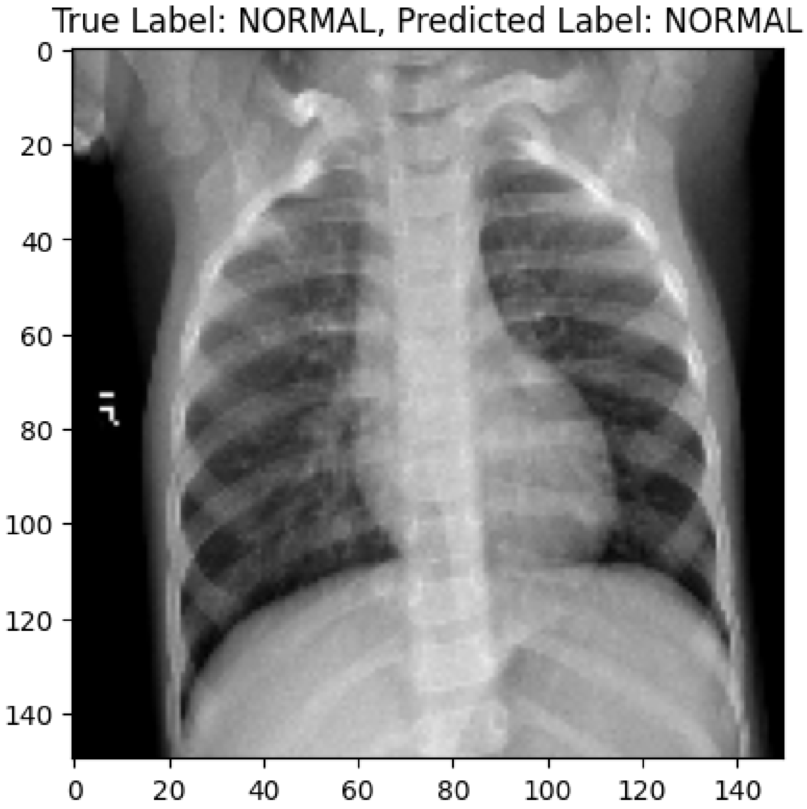
[16]:
```python
sample_images, sample_labels = next(test_generator)
predictions = model.predict(sample_images)
```

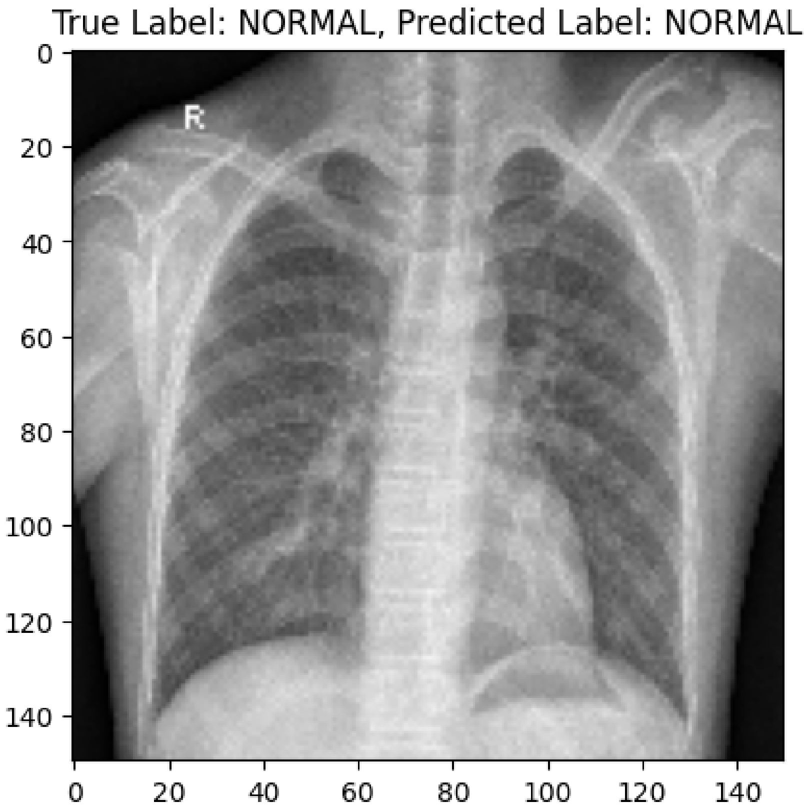
```
1/1 [==============================] - 1s 705ms/step
```

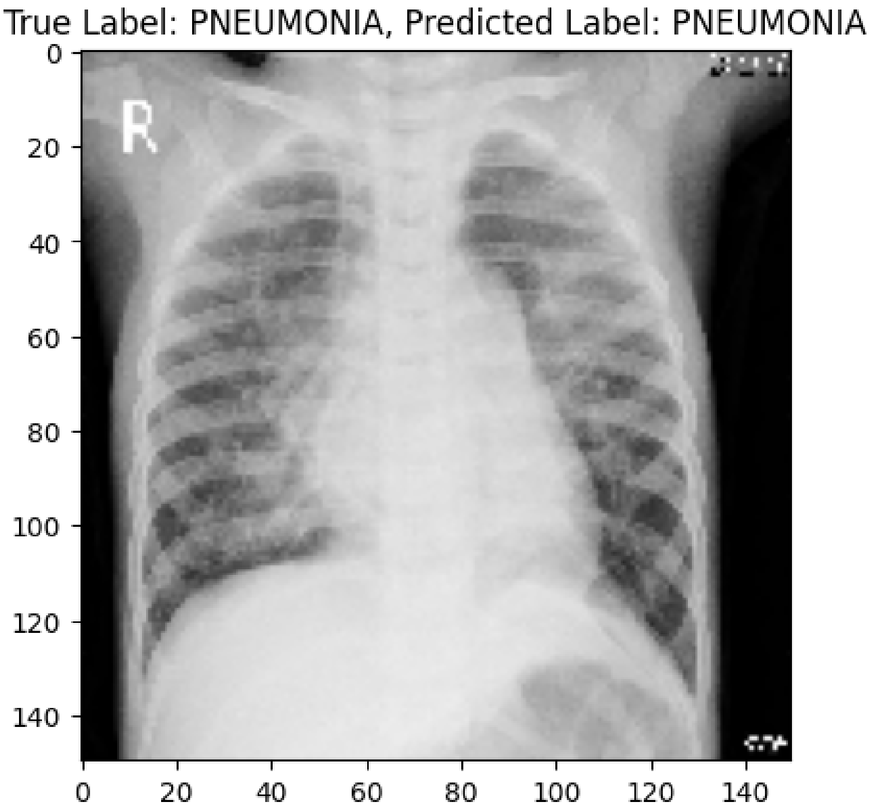
[17]:
```python
predicted_labels = [1 if p > 0.5 else 0 for p in predictions]
```

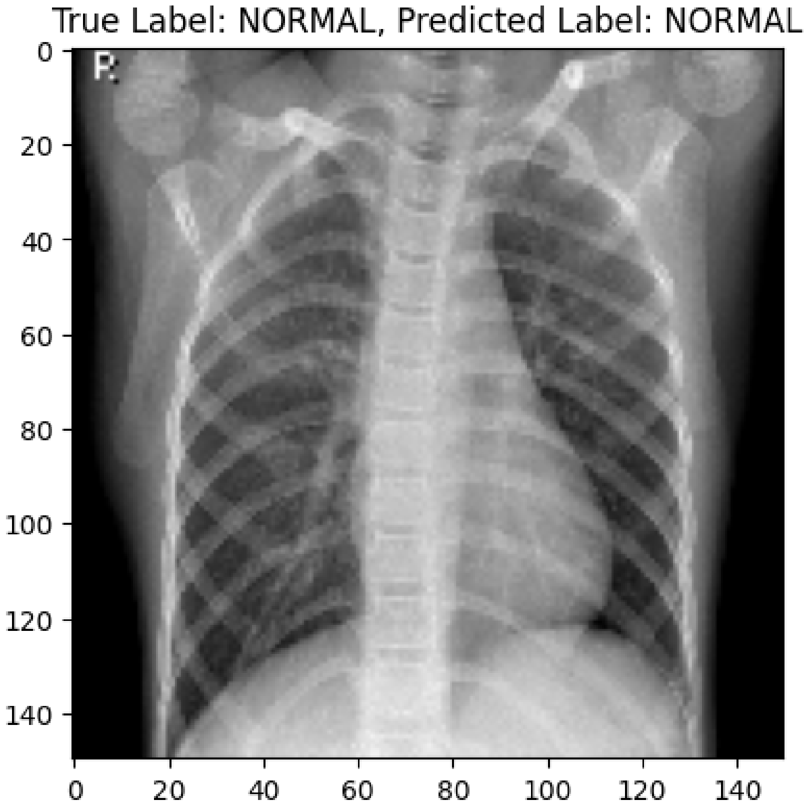
[21]:
```python
for i in range(5):
    true_label = "NORMAL" if sample_labels[i] == 0 else "PNEUMONIA"
    predicted_label = "NORMAL" if predicted_labels[i] == 0 else "PNEUMONIA"

    # Display the image
    plt.imshow(sample_images[i])
    plt.title(f"True Label: {true_label}, Predicted Label: {predicted_label}")
    plt.show()
```


True Label: PNEUMONIA, Predicted Label: PNEUMONIA
F.
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


True Label: NORMAL, Predicted Label: NORMAL
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


True Label: NORMAL, Predicted Label: NORMAL
R
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


True Label: PNEUMONIA, Predicted Label: PNEUMONIA
R
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


True Label: NORMAL, Predicted Label: NORMAL
R
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140

```
[ ]:
```