Leveraging Convolutional Neural Networks for Automated Pneumonia Detection in Chest X-ray images

Rishani V

23/PCSA/101

Abstract

Pneumonia is a lung infection caused by bacteria, viruses, or fungi, leading to inflammation and fluid buildup in the lungs, making it hard to breathe. It's usually diagnosed using chest X-rays, but this process can take time and depends on the radiologist's expertise. This project focuses on creating a user-friendly application that uses Convolutional Neural Networks (CNNs) to automatically detect pneumonia. The CNN model, trained on a large dataset named chest X-ray images from Kaggle, can quickly and accurately differentiate between healthy individuals and those with pneumonia. Users can upload a chest X-ray image through the application and instantly receive a diagnosis. This tool provides a faster, reliable, and accessible way to support healthcare professionals in detecting pneumonia efficiently.

Keywords: *Pneumonia detection, Deep Learning, CNN, Chest X-ray images, Automated diagnosis.*

Base paper implementation:

In [1]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np


print(tf.__version__)
```

```
2.17.0
```

In [2]:
```
train_datagen = ImageDataGenerator(rescale = 1./255,
                                   height_shift_range=0.2,
                                   featurewise_center=True,
                                   rotation_range=0.4,
                                   horizontal_flip = True)

test_datagen = ImageDataGenerator(rescale = 1./255,)
```

In [3]:
```
training_set = train_datagen.flow_from_directory("train",
                                                 target_size = (400, 400),
                                                 batch_size = 32,
                                                 class_mode = 'categorical')
```

```
Found 4788 images belonging to 3 classes.
```

In [4]:
```
test_set = test_datagen.flow_from_directory("test",
                                            target_size = (400, 400),
                                            batch_size = 32,
                                            class_mode = 'categorical')
```

```
Found 1124 images belonging to 3 classes.
```

In [5]:
```
training_set.class_indices
```

Out[5]:
```
{'COVID19': 0, 'NORMAL': 1, 'PNEUMONIA': 2}
```

In [6]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPool2D, Flatten, Dense, Dropout
```

In [7]:
```
model = Sequential()

model.add(Conv2D(filters = 32, padding = "same", kernel_size=(2,2),strides=(2,2), activation = "relu", input_shape=[400,400,3]))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters = 32, padding = "same", kernel_size=(2,2),strides=(2,2), activation = "relu", input_shape=[400,400,3]))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Conv2D(filters = 64, padding = "same", kernel_size=(2,2),strides=(2,2), activation = "relu", input_shape=[400,400,3]))
model.add(MaxPool2D(pool_size=(2,2)))

model.add(Flatten())
model.add(Dense(units = 132, activation="relu"))
model.add(Dense(units = 60, activation="relu"))
model.add(Dense(units = 3, activation="softmax"))

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

```
C:\Users\rishani vedamuthu\anaconda3\lib\site-packages\keras\src\layers\convolutional\base_conv.py:107: UserWarning: Do not pas
s 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)
```

In [8]:
```
model.fit(training_set,validation_data=test_set,epochs=60)
```

```
Epoch 52/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 229s 1s/step - accuracy: 0.9722 - loss: 0.0785 - val_accuracy: 0.9564 - val_loss: 0.1293
Epoch 53/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 273s 2s/step - accuracy: 0.9636 - loss: 0.0959 - val_accuracy: 0.9546 - val_loss: 0.1321
Epoch 54/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 227s 1s/step - accuracy: 0.9636 - loss: 0.0986 - val_accuracy: 0.9573 - val_loss: 0.1446
Epoch 55/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 220s 1s/step - accuracy: 0.9721 - loss: 0.0776 - val_accuracy: 0.9520 - val_loss: 0.1449
Epoch 56/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 223s 1s/step - accuracy: 0.9688 - loss: 0.0843 - val_accuracy: 0.9555 - val_loss: 0.1398
Epoch 57/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 225s 1s/step - accuracy: 0.9745 - loss: 0.0716 - val_accuracy: 0.9546 - val_loss: 0.1457
Epoch 58/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 221s 1s/step - accuracy: 0.9734 - loss: 0.0742 - val_accuracy: 0.9520 - val_loss: 0.1424
Epoch 59/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 219s 1s/step - accuracy: 0.9656 - loss: 0.0889 - val_accuracy: 0.9609 - val_loss: 0.1612
Epoch 60/60
150/150 ━━━━━━━━━━━━━━━━━━━━ 222s 1s/step - accuracy: 0.9690 - loss: 0.0781 - val_accuracy: 0.9564 - val_loss: 0.1257
```

Out[8]:
```
<keras.src.callbacks.history.History at 0x19a0819e280>
```

In [9]:
```
model.save('covid1.h5')
```

```
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')`.
```

In [5]:
```
from numpy import loadtxt
from keras.models import load_model
import tensorflow as tf


cnn = load_model('covid1.h5')

cnn.summary()
```

```
WARNING:absl:Compiled the loaded model, but the compiled metrics have yet to be built. `model.compile_metrics` will be empty until you train or evaluate the model.
```

Model: "sequential"

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 200, 200, 32)	416
max_pooling2d (MaxPooling2D)	(None, 100, 100, 32)	0
conv2d_1 (Conv2D)	(None, 50, 50, 32)	4,128
max_pooling2d_1 (MaxPooling2D)	(None, 25, 25, 32)	0
conv2d_2 (Conv2D)	(None, 13, 13, 64)	8,256
max_pooling2d_2 (MaxPooling2D)	(None, 6, 6, 64)	0
flatten (Flatten)	(None, 2304)	0
dense (Dense)	(None, 132)	304,260
dense_1 (Dense)	(None, 60)	7,980
dense_2 (Dense)	(None, 3)	183

Total params: 325,225 (1.24 MB)

Trainable params: 325,223 (1.24 MB)

Non-trainable params: 0 (0.00 B)

Optimizer params: 2 (12.00 B)

In [6]:
```
import numpy as np
test_image = tf.keras.utils.load_img('covid2.jpg', target_size = (400,400))
test_image
```

Out[6]:

In [7]:
```
test_image = tf.keras.utils.img_to_array(test_image)
test_image = np.expand_dims(test_image, axis = 0)
result = cnn.predict(test_image/255.0)
classes = np.argmax(result, axis=1)[0]
if classes == 1:
    print( "normal")
elif classes == 2:
    print(" pneumonia")
else:
    print( "covid19")
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step
covid19
```

In [8]:
```
loss, accuracy = cnn.evaluate(test_set)
print(f'Test Loss: {loss:.4f}')
print(f'Test Accuracy: {accuracy * 100:.2f}%')
```

```
C:\Users\rishani vedamuthu\anaconda3\lib\site-packages\keras\src\trainers\data_adapters\py_dataset_adapter.py:122: UserWarning:
Your `PyDataset` class should call `super().__init__(**kwargs)` in its constructor. `**kwargs` can include `workers`, `use_mult
iprocessing`, `max_queue_size`. Do not pass these arguments to `fit()`, as they will be ignored.
  self._warn_if_super_not_called()
```

```
36/36 ━━━━━━━━━━━━━━━━━━━━ 29s 792ms/step - accuracy: 0.9578 - loss: 0.1143
Test Loss: 0.1257
Test Accuracy: 95.64%
```