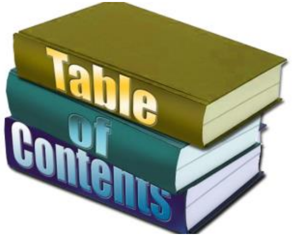
COVID-19 DETECTION FROM X-RAY IMAGES USING CONVOLUTIONAL NEURAL NETWORK (CNN)

FAIZA QAYYUM
AD20206810

Table of Contents



Project Source

Source: https://www.kaggle.com/technick/covid-detection-from-xray

Introduction

- The need for auxiliary diagnostic tools has increased as there is lack of automated toolkits to detect COVID-19.
- Researchers state that combining clinical image features with laboratory results may help in early detection of COVID-19 [1,2,3].
- Recent findings obtained using radiology imaging techniques suggest that such images contain salient information about the COVID-19 virus [4].
- In this project, a Convolutional Neural Network (CNN) is designed to automatically diagnose COVID-19 using x-ray images

Libraries

Kera’s

- High-level neural networks library

TensorFlow

- High performance numerical computation based library for ML applications

Numpy

- Fundamental scientific computations

Pandas

- **Data Analysis**: pandas is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool

Matplotlib

- Data visualization

```
from keras import backend as K
from keras.preprocessing.image import ImageDataGenerator,load_img, img_to_array
from keras.models import Sequential, Model
from keras.layers import Conv2D, MaxPooling2D,GlobalAveragePooling2D
from keras.layers import Activation, Dropout, BatchNormalization, Flatten, Dense, AvgPool2D,MaxPool2D
from keras.models import Sequential, Model
from keras.applications.vgg16 import VGG16, preprocess_input
from keras.optimizers import Adam, SGD, RMSprop

import tensorflow as tf

import os
import numpy as np
import pandas as pd

import matplotlib.pyplot as plt
%matplotlib inline
```

Data Set

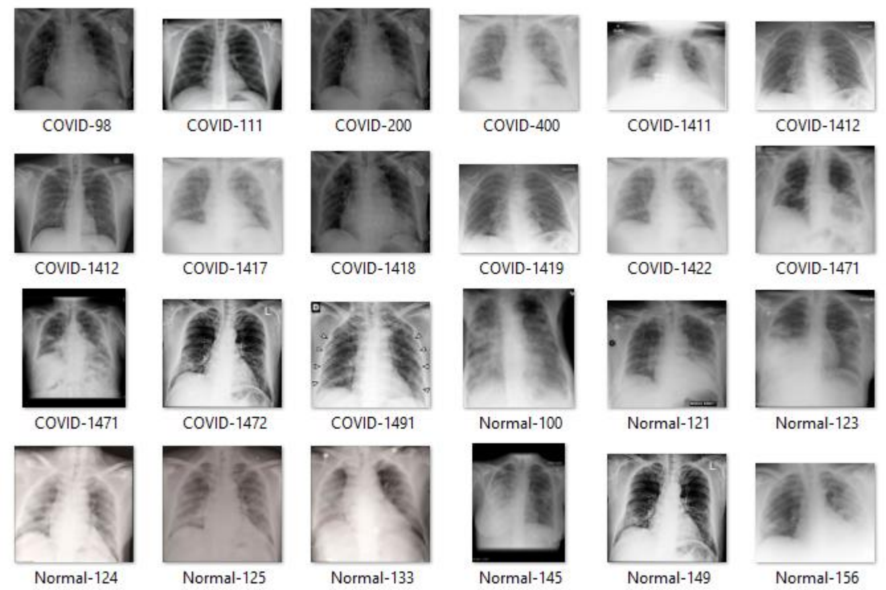
Data Set Overview

- **No. of X-ray Images** = 1098
- There are two classes:
  - 1) Covid (751)
  - 2) Normal (347)
- Dataset containing X-ray Images from two classes (‘covid’, ‘normal’) is loaded into “DATASET_DIR”
- Images from both the classes are placed in their respective folders

```
DATASET_DIR = "../input/covid-19-x-ray-10000-images/dataset"
```

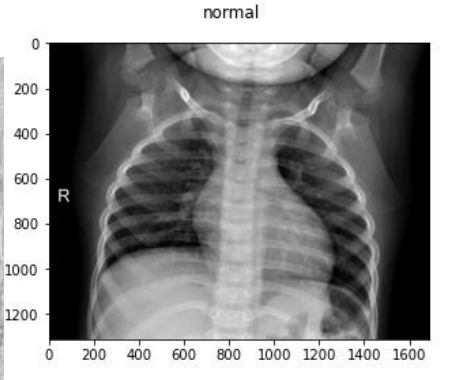
```
os.listdir(DATASET_DIR)
```

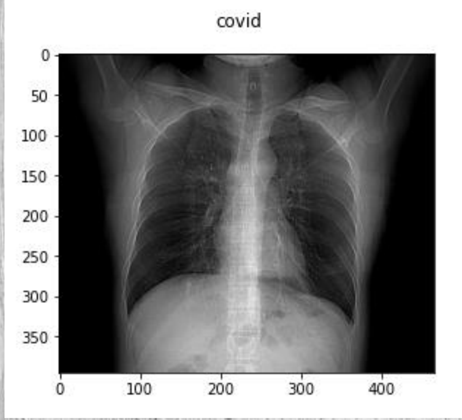
```
['covid', 'normal']
```



Data Analysis

```
import glob
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
%matplotlib inline

normal_images = []
for img_path in glob.glob(DATASET_DIR + '/normal/*'):
    normal_images.append(mpimg.imread(img_path))

fig = plt.figure()
fig.suptitle('normal')
plt.imshow(normal_images[0], cmap='gray')

covid_images = []
for img_path in glob.glob(DATASET_DIR + '/covid/*'):
    covid_images.append(mpimg.imread(img_path))

fig = plt.figure()
fig.suptitle('covid')
plt.imshow(covid_images[0], cmap='gray')
```





Convolutional Neural Network (CNN)

Hyper parameters Initialization:

- All the images are transformed into fixed size WIDTH and HEIGHT (i.e., 150 x 150).

CHANNELS:

- 3 (for R, G and B)

INPUT_SHAPE:

- Give array as an input to CNN

NB_CLASSES:

- binary

EPOCHS:

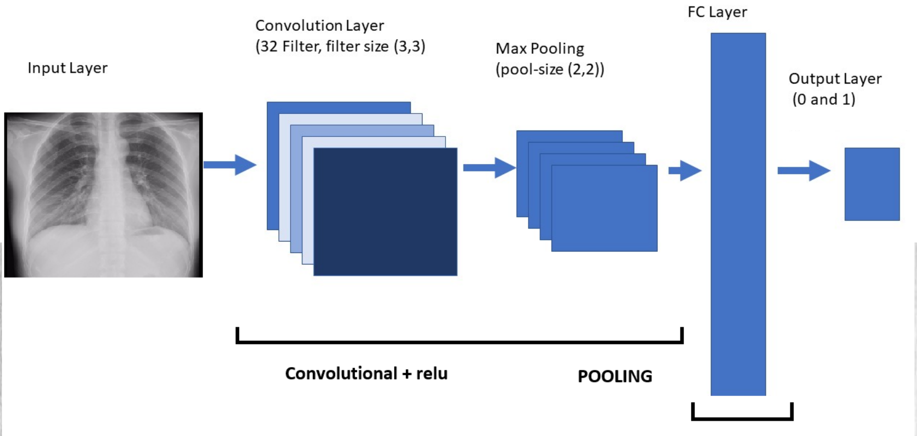
- Epoch is when an ENTIRE dataset is passed forward and backward through the neural network only ONCE.

BATCH_SIZE:

- Since one epoch is too big to feed to the computer at once we divide it in several smaller batches.

```
IMG_W = 150
IMG_H = 150
CHANNELS = 3

INPUT_SHAPE = (IMG_W, IMG_H, CHANNELS)
NB_CLASSES = 2
EPOCHS = 48
BATCH_SIZE = 6
```

Convolutional Neural Network (CNN)



Conv2D

- Conv2D is used to creates a convolution filter

Filter Size:

- 32, 32, 64, 128

Kernel Size:

- 3 x 3

Activation Function

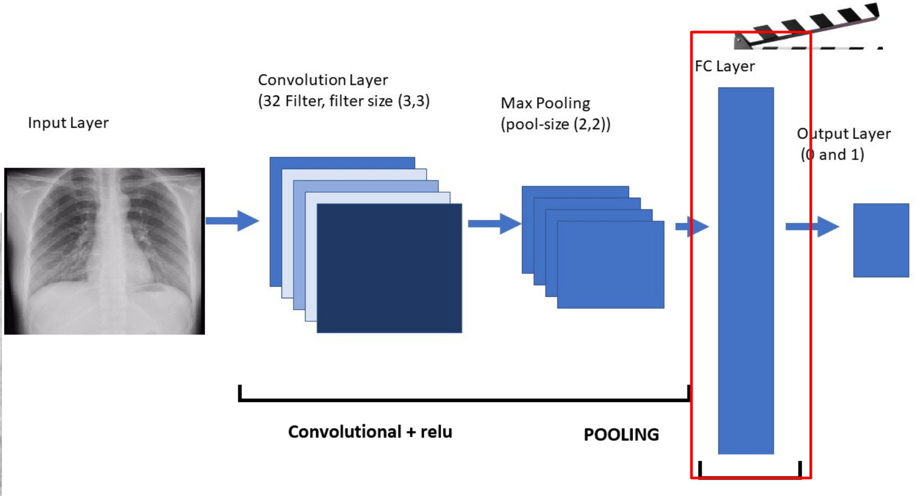
- RELU is used as an activation function to add non-linearity

MaxPooling2D

- Reduce the amount of parameters and computation in the network

```
model = Sequential()
model.add(Conv2D(32, (3, 3), input_shape=INPUT_SHAPE))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))

model.add(Conv2D(32, (3, 3)))
model.add(Activation('relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))


model.add(Conv2D(64,(3,3)))
model.add(Activation("relu"))
model.add(Conv2D(250,(3,3)))
model.add(Activation("relu"))

model.add(Conv2D(128,(3,3)))
model.add(Activation("relu"))
model.add(AvgPool2D(2,2))
model.add(Conv2D(64,(3,3)))
```

Convolutional Neural Network (CNN)



Flatten Layer

- Used for converting a two-dimensional matrix of features into a vector that can be fed into a fully connected neural network classifier

Dense Layer

- Each neuron in dense layer receives input from all the neurons in the previous layer, thus densely connected.

Dropout

- Used in regular interval for generalization purpose

Sigmoid

- method is used as an activation method for output layer.

```
model.add(Conv2D(256,(2,2)))
model.add(Activation("relu"))
model.add(MaxPool2D(2,2))

model.add(Flatten())
model.add(Dense(32))
model.add(Dropout(0.25))
model.add(Dense(1))
model.add(Activation("sigmoid"))
```

Convolutional Neural Network (CNN)

Compile Deep CNN model:

- For a **binary classification** like our example, the **typical loss function** is the **binary cross-entropy / log loss**

Optimizer:
- RMSprop

Metrics:
- Accuracy

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN}$$

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

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_7 (Conv2D)            (None, 148, 148, 32)      896
_________________________________________________________________
activation_7 (Activation)    (None, 148, 148, 32)      0
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 74, 74, 32)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 72, 72, 32)        9248
_________________________________________________________________
activation_8 (Activation)    (None, 72, 72, 32)        0
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 36, 36, 32)        0
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 34, 34, 64)        18496
```

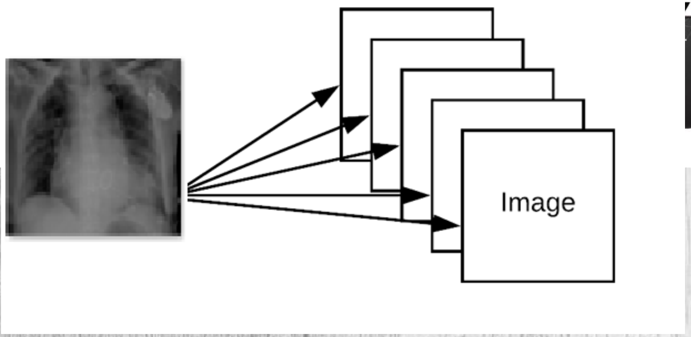
Convolutional Neural Network (CNN)

```
activation_9 (Activation)    (None, 34, 34, 64)        0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 32, 32, 250)       144250
_________________________________________________________________
activation_10 (Activation)   (None, 32, 32, 250)       0
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 30, 30, 128)       288128
_________________________________________________________________
activation_11 (Activation)   (None, 30, 30, 128)       0
_________________________________________________________________
average_pooling2d_2 (Average (None, 15, 15, 128)       0
_________________________________________________________________
conv2d_12 (Conv2D)           (None, 13, 13, 64)        73792
_________________________________________________________________
activation_12 (Activation)   (None, 13, 13, 64)        0
_________________________________________________________________
average_pooling2d_3 (Average (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_13 (Conv2D)           (None, 5, 5, 256)         65792
```

```
_________________________________________________________________
activation_13 (Activation)   (None, 5, 5, 256)         0
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 256)         0
_________________________________________________________________
flatten (Flatten)            (None, 1024)              0
_________________________________________________________________
dense (Dense)                (None, 32)                32800
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
_________________________________________________________________
activation_14 (Activation)   (None, 1)                 0
=================================================================
Total params: 633,435
Trainable params: 633,435
Non-trainable params: 0
_________________________________________________________________
```

Training and Testing

Data Augmentation



ImageDataGenerator:

- increase number of images by augmenting a few images

Shear_range:

- shear_range is for randomly applying shearing transformations

Zoom_range:

- randomly zooming inside pictures

Horizontal flip:

- randomly flipping half of the images horizontally

Validation_split:

- Set 0.3 (**70% data for testing** and **30% for training**)

target_size

- size of your input images, every image will be resized to this size.

batch_size:

- no. of images to be yielded from the generator per batch.

class_mode:

- set "binary" if you have only two classes to predict,

shuffle:

- set True if you want to shuffle the order of the image that is being yielded, else set False.

```
train_datagen = ImageDataGenerator(rescale=1./255,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True,
    validation_split=0.3)

train_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_H, IMG_W),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    subset='training')

validation_generator = train_datagen.flow_from_directory(
    DATASET_DIR,
    target_size=(IMG_H, IMG_W),
    batch_size=BATCH_SIZE,
    class_mode='binary',
    shuffle= False,
    subset='validation')
```

Fit the model

Model.fit_generator:

steps_per_epoch

- value as the total number of training data points divided by the batch size. Once Keras hits this step count it knows that it's a new epoch.

```
history = model.fit_generator(
    train_generator,
    steps_per_epoch = train_generator.samples // BATCH_SIZE,
    validation_data = validation_generator,
    validation_steps = validation_generator.samples // BATCH_SIZE,
    epochs = EPOCHS)
```

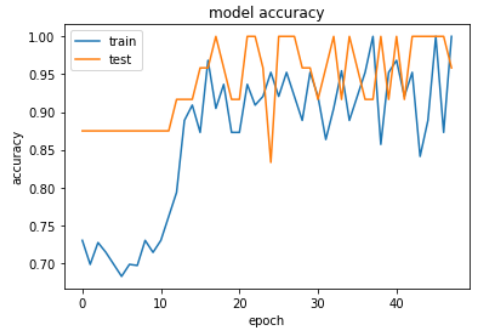
```
Epoch 1/48
11/11 [==============================] - 5s 454ms/step - loss: 0.9752 -
accuracy: 0.7302 - val_loss: 0.5168 - val_accuracy: 0.8750
Epoch 2/48
11/11 [==============================] - 5s 427ms/step - loss: 0.6296 -
accuracy: 0.6984 - val_loss: 0.5840 - val_accuracy: 0.8750
Epoch 3/48
11/11 [==============================] - 5s 450ms/step - loss: 0.7205 -
accuracy: 0.7273 - val_loss: 0.6117 - val_accuracy: 0.8750
Epoch 4/48
11/11 [==============================] - 5s 423ms/step - loss: 0.6340 -
accuracy: 0.7143 - val_loss: 0.4250 - val_accuracy: 0.8750
```

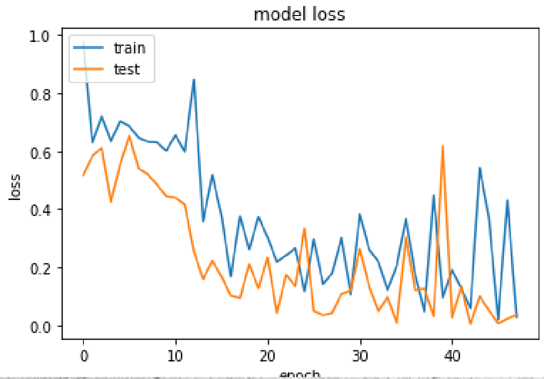
```
Epoch 48/48
11/11 [==============================] - 5s 445ms/step - loss: 0.0291 -
accuracy: 1.0000 - val_loss: 0.0364 - val_accuracy: 0.9583
```

Performance Analysis

- The epochs history shows that accuracy gradually increases and achieved +95% accuracy on both training and validation set.
- The loss value of the model gradually decreases as the number of training epoch increases.

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```





Conclusion

- CNN model has been proven useful in detecting “covid” via X-ray images data.
- The model has tendency to predict true positives with significant accuracy
- However, the size of data is too small to draw a generic conclusion

References

[1] Wu F., Zhao S., Yu B. A new coronavirus associated with human respiratory disease in China. Nature. 2020;579(7798):265–269. [PMC free article] [PubMed] [Google Scholar]

[2] Huang C., Wang Y. Clinical features of patients infected with 2019 novel coronavirus in Wuhan, China. Lancet. 2020;395(10223):497–506.

[3] World Health Organization . World Health Organization (WHO); 2020. Pneumonia of Unknown Cause–China. Emergencies Preparedness, Response, Disease Outbreak News.

[4] Wu Z., McGoogan J.M. Characteristics of and important lessons from the coronavirus disease 2019 (COVID-19) outbreak in China: summary of a report of 72 314 cases from the Chinese Center for Disease Control and Prevention. Jama. 2020;323(13):1239–1242.

THANKS!