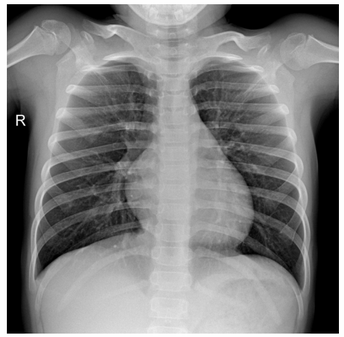
Diagnosis-of-Pneumonia

Deep Learning CNN Project



AI Model

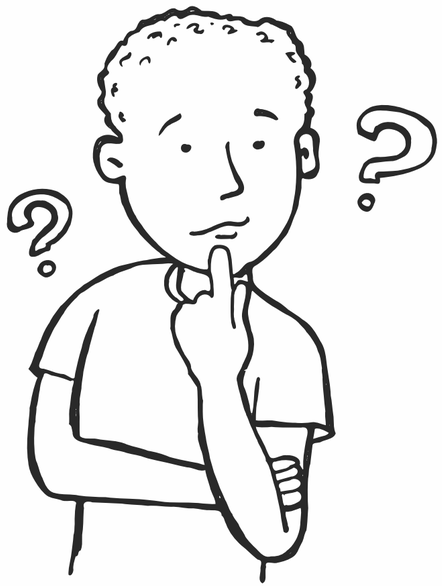
Presented by Stephen Fok

Content

- Reason for this project
- Project objective
- Project value
- Brief Project Introduction and Result
- Demonstration
- Conclusion

Hospitals overwhelmed by Omicron outbreak in early 2022

What can we help?



We need to do something!!!

Objective

To train up a model which can diagnose pneumonia precisely and quickly.

Project value

It is known that pneumonia (COVID) has swept through the world since early 2020.

- To reduce workload of doctors to diagnose pneumonia for patients.
- To shorten the time for diagnosis.

Project Brief Introduction

- Thousands of images of X-Rays of Pneumonia confirmed cases and normal people are used as training and test datasets.
- Input shape of training dataset: 280, 280, 3
- Output class: Binary
- Model Accuracy: about 90%
- Output: 1 stands for confirmed case of pneumonia, 0 stands for negative.
- Languages: Python
- Platform used: Jupyter Notebook
- Library used: Deep Learning Tensorflow, scikit-image, Numpy, Matplotlib, OpenCV etc.

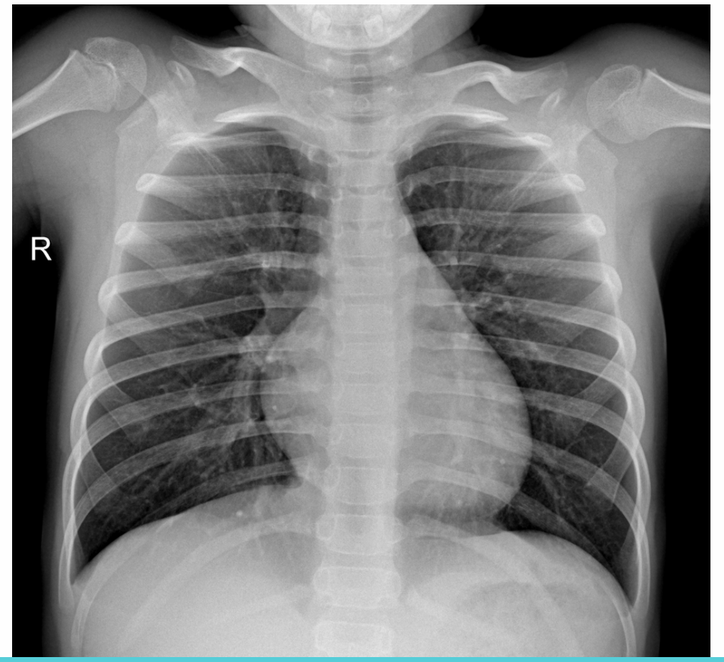
X-Ray Images(from Kaggle)

Confirmed case

X-Ray images:1349pcs

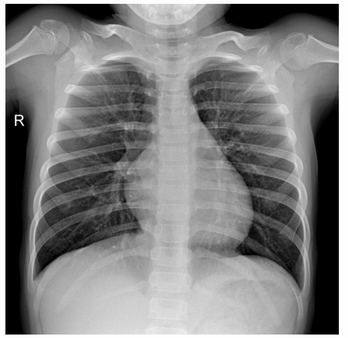
Normal

X-Ray images:1349pcs



Conversion from images to dataset array

```python
# create a function to convert images to dataset - training dataset

def create_datasets_train(img_size):
    imgs_infected = []
    for file in tqdm(list_filename_infected_train[:1349]):  #tqdm is just to show the progress bar
        img_infected = cv2.imread(path_img_infected_train+file)
        img_infected = cv2.cvtColor(img_infected, cv2.COLOR_BGR2RGB)
        img_infected = cv2.resize(img_infected, (img_size,img_size))    #can use this to resize the image
        imgs_infected.append(img_infected)

    imgs_uninfected = []
    for file in tqdm(list_filename_uninfected_train[:1349]):
        img_uninfected = cv2.imread(path_img_uninfected_train+file)
        img_uninfected = cv2.cvtColor(img_uninfected, cv2.COLOR_BGR2RGB)
        img_uninfected = cv2.resize(img_uninfected, (img_size,img_size))    #can use this to resize the image
        imgs_uninfected.append(img_uninfected)

    imgs= imgs_infected + imgs_uninfected

    # not yet normalized
    imgs = np.array(imgs)
    label = []
    for i in list_filename_infected_train[:1349]:
        i = True
        label.append(i)
    for i in list_filename_uninfected_train[:1349]:
        i = False
        label.append(i)
    label = np.array(label)

    return imgs, label
```




```
array([[[[179, 179, 179],
         [ 84,  84,  84],
         [ 48,  48,  48],
         ...,
         [ 37,  37,  37],
         [ 52,  52,  52],
         [ 83,  83,  83]],

        [[183, 183, 183],
```

Normalization

```
X_train = X_train.astype('float32')
X_train /= 255.
y_train = y_train.astype('int')
y_train
```

```
array([1, 1, 1, ..., 0, 0, 0])
```

CNN modeling

```
num_classes = 1
input_shape = IMG_SIZE, IMG_SIZE, 3
batch_size = 100
epochs = 10
```

```
# Define the model
model = Sequential()

# Create a CNN to classify the images
model.add(Conv2D(32, kernel_size=(3, 3),activation='relu',input_shape=input_shape))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Conv2D(32, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.5))
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dense(64, activation='relu'))
#model.add(Dropout(0.5))
model.add(Dense(num_classes, activation='sigmoid'))
```

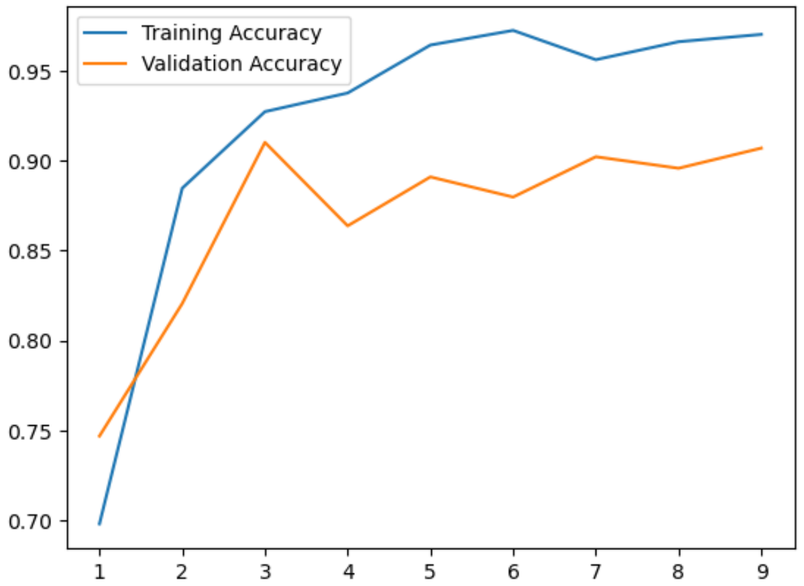
```
#   define compile to minimize categorical loss, use some optimizer to optimize, and optimize to maximizing accuracy
model.compile(loss='binary_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])

#   Train the model and test/validate the mode with the test data after each cycle (epoch) through the training data
#   Return history of loss and accuracy for each epoch
hist = model.fit(X_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          verbose=1,
          validation_data=(X_test, y_test))
```

```
Epoch 1/10
27/27 [==============================] - 167s 6s/step - loss: 2.0657 - accuracy: 0.5374 - val_loss: 0.6926 - val_accuracy: 0.6266
Epoch 2/10
27/27 [==============================] - 176s 7s/step - loss: 0.6488 - accuracy: 0.7157 - val_loss: 0.5748 - val_accuracy: 0.6538
Epoch 3/10
27/27 [==============================] - 182s 7s/step - loss: 0.3099 - accuracy: 0.8773 - val_loss: 0.4520 - val_accuracy: 0.7420
Epoch 4/10
27/27 [==============================] - 178s 7s/step - loss: 0.2100 - accuracy: 0.9181 - val_loss: 0.3552 - val_accuracy: 0.8766
Epoch 5/10
```

Model Evaluation

```
#   Evaluate the model with the test data to get the scores on "real" data.
score = model.evaluate(X_train, y_train, verbose=0)
print('Test loss:', score[0])
print('Test accuracy:', score[1])

#   Plot data to see relationships in training and validation data
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
epoch_list = list(range(1, len(hist.history['accuracy']) + 1))  # values for x axis [1, 2, ..., # of epochs]
plt.plot(epoch_list, hist.history['accuracy'], epoch_list, hist.history['val_accuracy'])
plt.legend(('Training Accuracy', 'Validation Accuracy'))
plt.show()
```

```
Test loss: 0.10908608138561249
Test accuracy: 0.9781319499015808
```



Accuracy about 90%

Result with unseen X-Ray images

NORMAL-8876914-0001.jpeg is negative
NORMAL-8944072-0001.jpeg is negative
NORMAL-9077486-0001.jpeg is negative
NORMAL-9092354-0001.jpeg is negative
NORMAL-9145111-0001.jpeg is negative
NORMAL-9194822-0001.jpeg is negative
NORMAL-9217241-0001.jpeg is positive
NORMAL-9260113-0001.jpeg is negative
NORMAL-9275512-0001.jpeg is negative
NORMAL-9334007-0001.jpeg is negative
NORMAL-9343440-0001.jpeg is negative
NORMAL-9441169-0001.jpeg is negative

VIRUS-9175020-0003.jpeg is positive
VIRUS-9176009-0002.jpeg is positive
VIRUS-9176369-0002.jpeg is positive
VIRUS-9188112-0002.jpeg is positive
VIRUS-9197445-0002.jpeg is positive
VIRUS-9208098-0003.jpeg is positive
VIRUS-9237186-0002.jpeg is positive
VIRUS-9243857-0001.jpeg is positive
VIRUS-9248466-0001.jpeg is positive
VIRUS-9250380-0006.jpeg is positive
VIRUS-9844887-0001.jpeg is positive
VIRUS-9867917-0002.jpeg is positive

Conclusion

- Faster diagonosis achieved
- More precise diagonosis achieved (~90%)
- Workload of doctors significantly reduced
- Shorter diagnosis time

Thank you!


Stay
Safe