CNN For Detecting Pneumonia from X-ray Images

Meral Balik

May 23, 2020

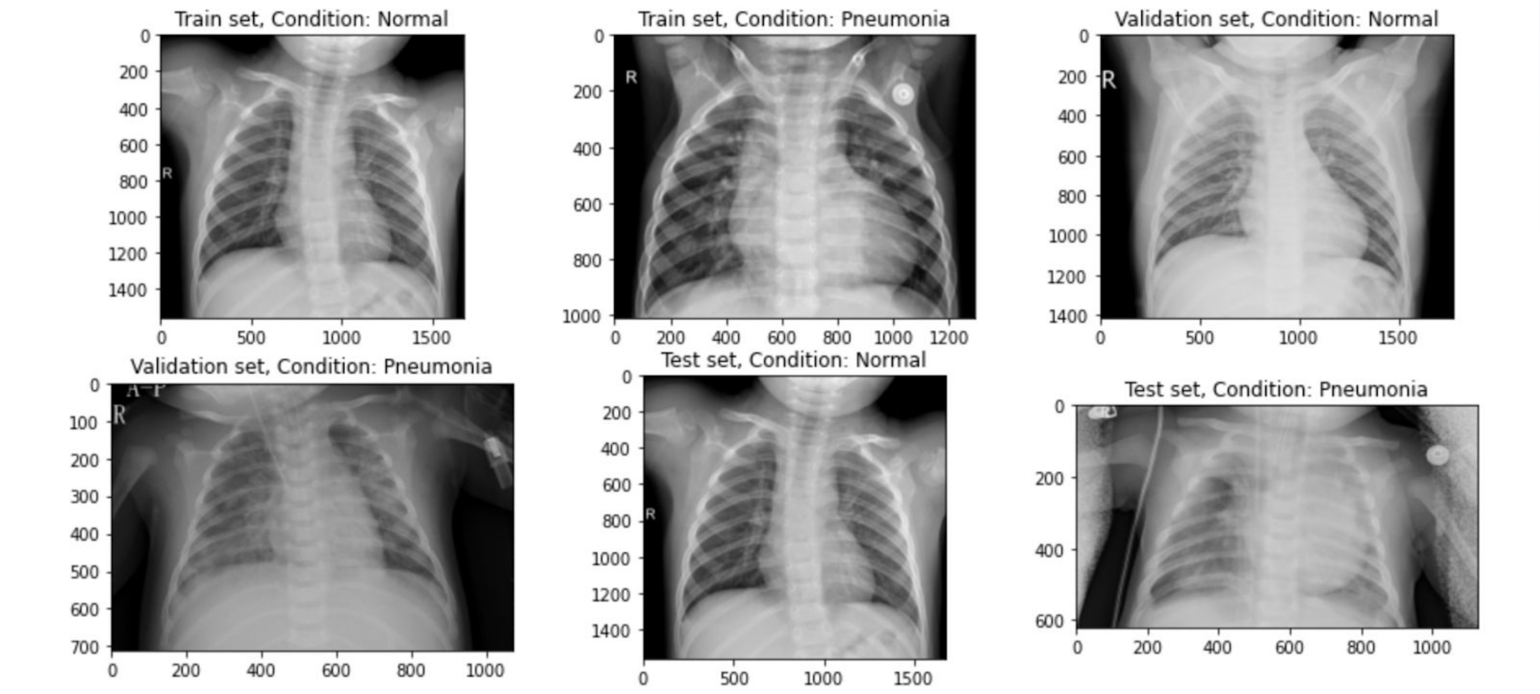
Springboard

Contents

Problem Statement

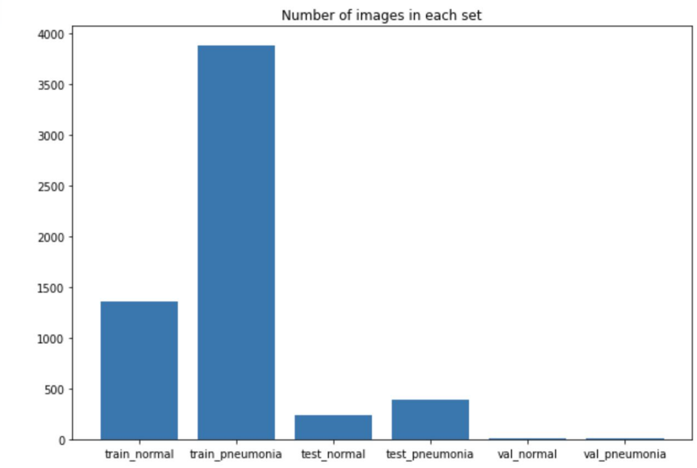
- Globally, 450 million get infected by pneumonia in a year
- 4 million people die from the disease in the world
- 50 thousand people die from the disease in the United States of America
- Analyzing and classifying chest x-rays can be very tedious for radiologists
- Artificial intelligence-based solutions can provide support for experts in the medical domain in performing time-consuming works

Dataset


Train set, Condition: Normal
0 200 400 600 800 1000 1200 1400
0 500 1000 1500
Train set, Condition: Pneumonia
0 200 400 600 800 1000
0 200 400 600 800 1000 1200
Validation set, Condition: Normal
0 200 400 600 800 1000 1200 1400
0 500 1000 1500
Validation set, Condition: Pneumonia
0 100 200 300 400 500 600 700
0 200 400 600 800 1000
Test set, Condition: Normal
0 200 400 600 800 1000 1200 1400
0 500 1000 1500
Test set, Condition: Pneumonia
0 200 400 600
0 200 400 600 800 1000

Dataset

- https://www.kaggle.com/paultimothymooney/chest-xray-pneumonia
- Organized into 3 folders (train, test, val)
- 5,863 X-Ray images (JPEG) and 2 categories (Pneumonia / Normal)




```
Number of images in train_normal_dir: 1354
Number of images in train_pneumonia_dir: 3875
Number of images in val_normal_dir: 8
Number of images in val_pneumonia_dir: 8
Number of images in test_normal_dir: 234
Number of images in test_pneumonia_dir: 390
```

Figure 1. *Number of images in each set*

Data Acquisition Using Colab

- Keras
  - High level library for building deep learning models
  - Tensorflow framework backend
  - Leverage backend engine to compute on GPU
- Google Colab
  - Jupyter notebook environment running Python in Cloud
  - Access to GPU resources
  - Free

Data Preprocessing

- ImageDataGenerator
  - reads images from disk and convert them to float32 tensors and feed them to the network
  - sets up generators that are capable of loading the required amount of data directly from the source folder, converting them into training data
  - rescales the images
  - performs data augmentation
    - zoom_range =0.3 and vertical_flip = True
- flow_from_directory
  - load images from the disk, applies rescaling, and resizes the images into the required dimensions.

Application of CNN

- Input
  - ImageDataGenerator
- Convolutional Base
  - Four convolutional layers
  - Max Pooling – downsampling
  - Batch normalization – speed up training
  - Learns patterns at different scales
- Dense Classifier
  - Four dense layers
  - Dropout layers – prevent overfitting
  - Sigmoid – binary classification problem

Baseline Model

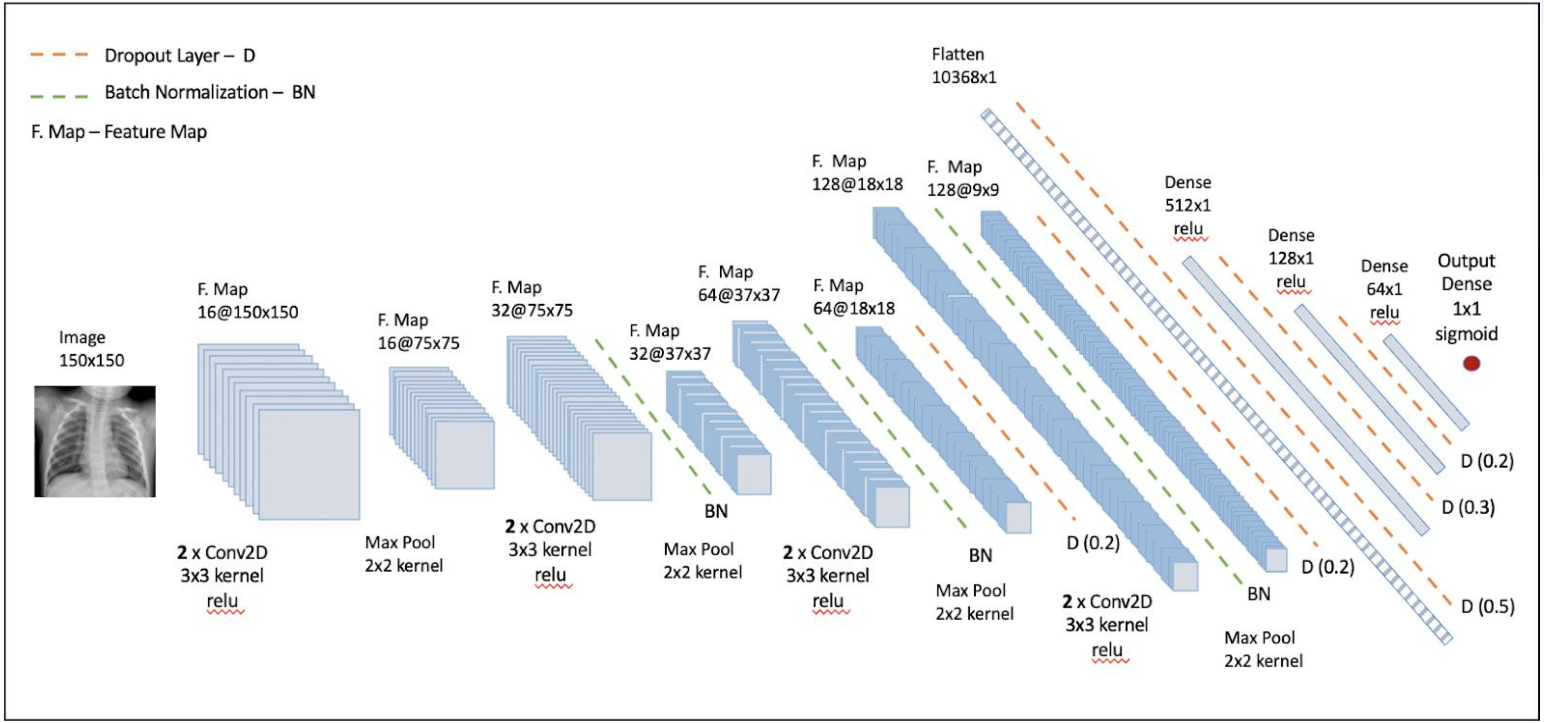


Figure 2. *Baseline Model*

Compiling the Model

Loss Function	Optimizer	Output Metrics
How performance is measured on training data	How network parameters are updated during training	What measures to record during training
Binary – cross entropy (two classes – ‘normal’ and ‘pneumonia’)	Adam - automatically adapts the learning rate during training	Accuracy

Training the Model

Callback Functions		
ModelCheckpoint	ReduceLROnPlateau	EarlyStopping
Saves model or weights at some intervals so the model or weights can be loaded later to continue the training.	Monitors a quantity and if no improvement is seen for a 'patience' number of epochs, the learning rate is reduced.	Prevents overtraining of the model by terminating the training process if it's not really learning anything.

```
from tensorflow.keras.callbacks import EarlyStopping, ReduceLROnPlateau, ModelCheckpoint

checkpoint = ModelCheckpoint(filepath='best_weights.hdf5', monitor='val_loss', save_best_only=Tru
e, save_weights_only=True)
lr_reduce = ReduceLROnPlateau(monitor='val_loss', factor=0.3, patience=2, verbose=1, mode='max')
early_stopping = EarlyStopping(monitor='val_loss', min_delta=0.1, patience=5, mode='auto',verbose=
1)
```

Evaluating the Model

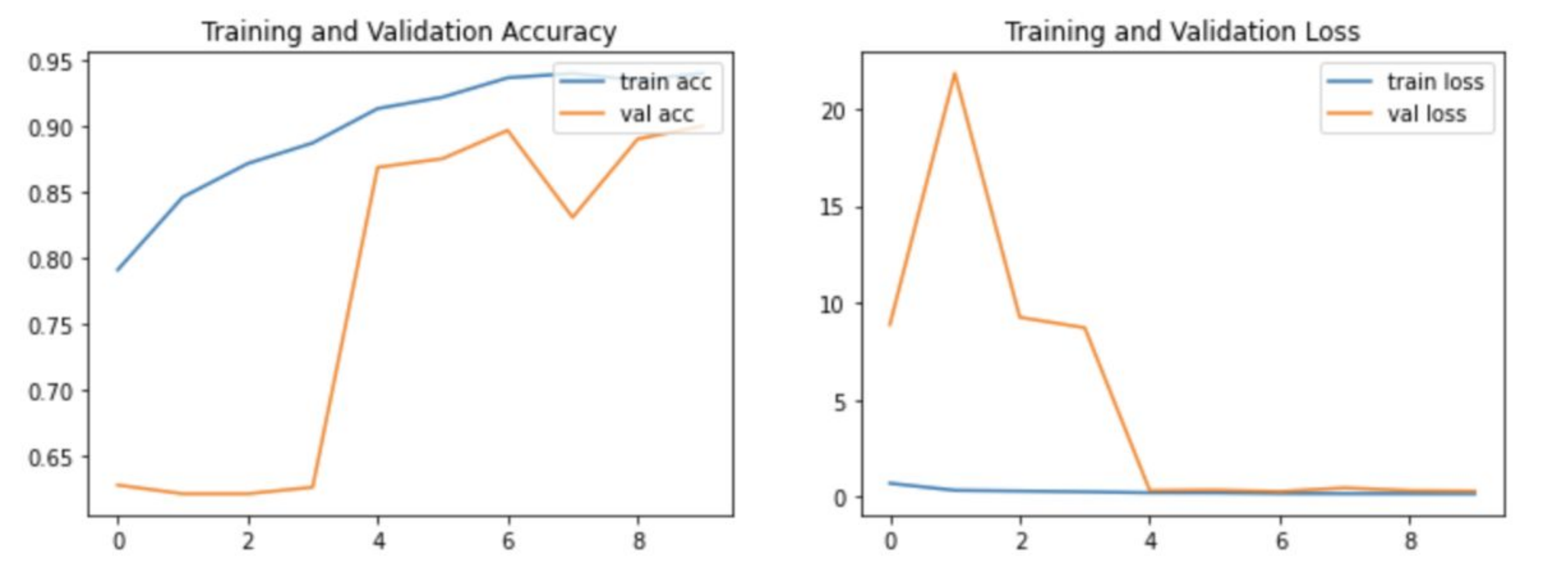


Figure 3. *Training History Plots*

Model Performance

Overall accuracy on test data: 90% with 0.29 loss

	precision	recall	f1-score	support
0	0.95	0.77	0.85	234
1	0.88	0.97	0.92	390
accuracy			0.90	624
macro avg	0.91	0.87	0.89	624
weighted avg	0.90	0.90	0.89	624

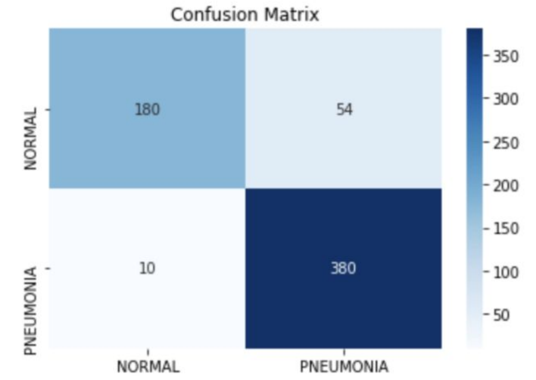


Figure 4. *Classification Report and Confusion Matrix*

Summary of Results

- Applied Deep Learning with CNN to classify chest x-ray images
- Used Google Colab to apply the architecture (using Keras)
- Developed a baseline CNN model with 90% accuracy
- Four convolutional layers (max-pooling, batch normalization)
- Four dense layers (drop out layers, sigmoid function)

Follow-on Work

- Improve model accuracy with different approaches
  - Increase the number of convolutional layers
  - Change predetermined values (batch size, epochs, callbacks…)
- Increase the number of data using
  - Data augmentation
  - Generative Adversarial Networks
- Train the data using ‘Transfer Learning’ techniques

Thank you!

Meral BALIK
E-mail : tekmer@hotmail.com
Linkedin : www.linkedin.com/in/meralbalik
Project @ https://github.com/Meralbalik/Capstone-Project-2