Abnormal and Normal Chest X-ray Classification

- Navaneesh Gangala
Data Science

Problem Statement

01

Assigned :

Develop a model to classify Abnormal and Normal Chest X-ray and to predict the probability of X-ray Images to be normal or abnormal

02

Improvements:

Production level code with exceptional handling and deployment ready

03

Future Scope:

With huge data Medical Application creation ,API building and can handle similar binary image classification problems across domains/verticals

Proposed Solution

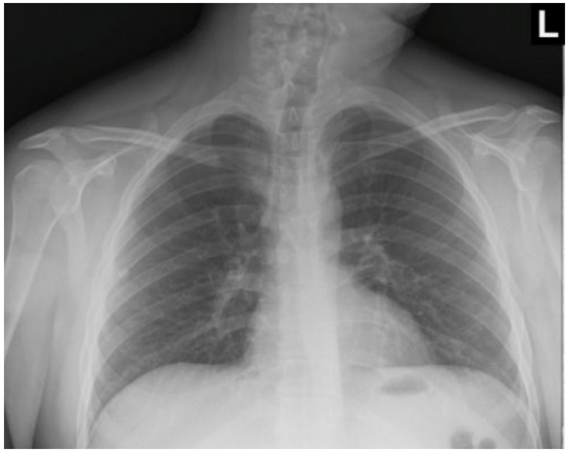
Chest X Ray image analysis using Deep Learning and exploring Deep Transfer Learning technique for it with TensorFlow.

The maxpool-5 layer of a pre trained VGGNet-16(Deep Convolutional Neural Network) model has been used as the feature extractor here and then further trained on a 2-layer Deep neural network with SGD optimizer and Batch Normalization for classification of Normal vs Abnormal Chest X Ray Images.

Data Understanding

Grey scale chest X ray pictures which have been pre classified has normal and abnormal .

Normal



Abnormal

Data Preprocessing

Loading the image data along with its Target variable – Normal or Abnormal in different variable . One variable will be storing the image name, target variable and other stores images with pixels

We would create Train and Test data and will take care of random shuffling

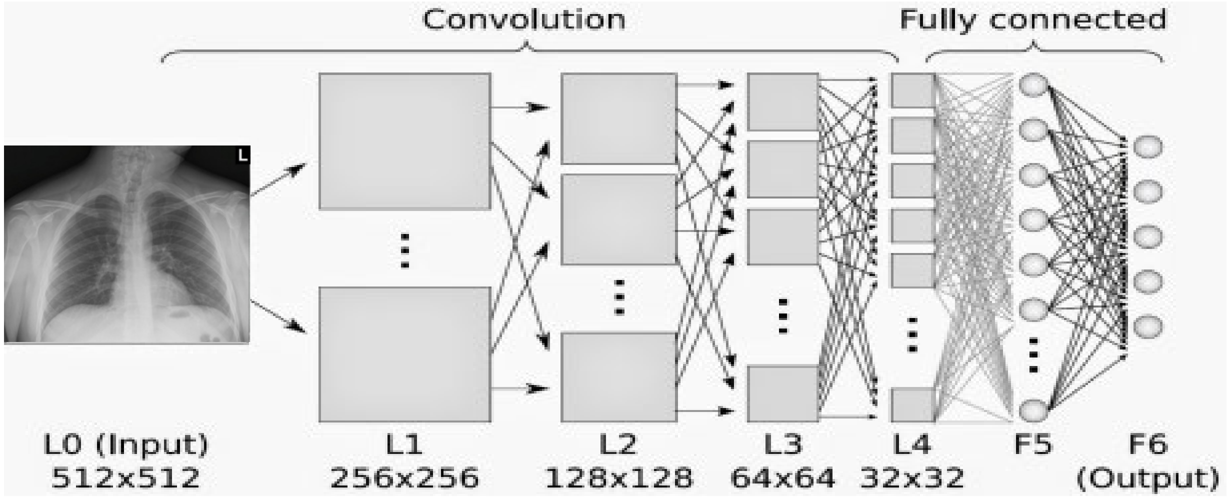
- Training images folder - All images for training
- Testing images Folder - All images for testing

- Training image labels file - Pickled file with training labels
- Testing image labels file - Pickled file with testing labels

Feature Extraction

Extract features (**CNN Codes**) from the **maxpool:5** layer of Pretrained CNN (VggNet) and save them beforehand for faster training of Neural network.

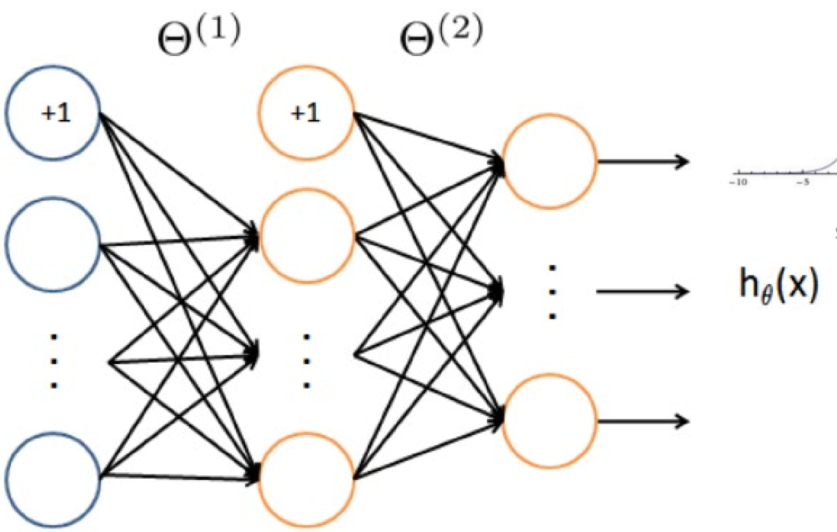
- Train images codes folder – Train Data Feature file
- Test images codes folder – Test Data Feature file



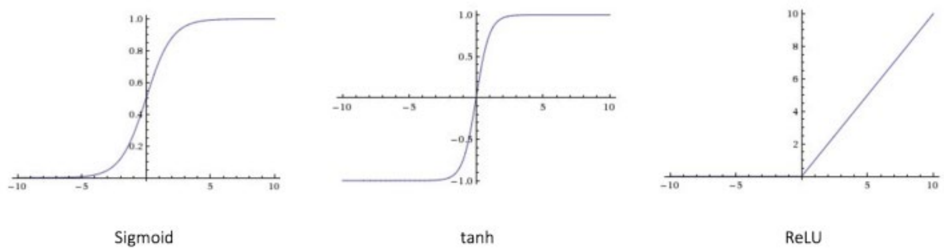
train.py

Model Training

The extracted features are now used for training our **2-Layer Neural Network** from scratch. The computed models are saved as TensorFlow checkpoint after every **Epoch**



Input Layer	Hidden Layer	Output Layer
$a^{(1)} = x$ (add $a_0^{(1)}$)	$z^{(2)} = \Theta^{(1)} a^{(1)}$ $a^{(2)} = g(z^{(2)})$ (add $a_0^{(2)}$)	$z^{(3)} = \Theta^{(2)} a^{(2)}$ $a^{(3)} = g(z^{(3)}) = h_\theta(x)$

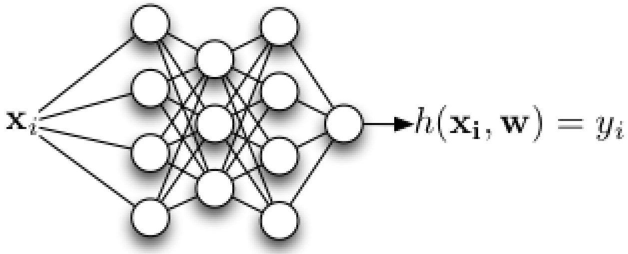


train_model.py

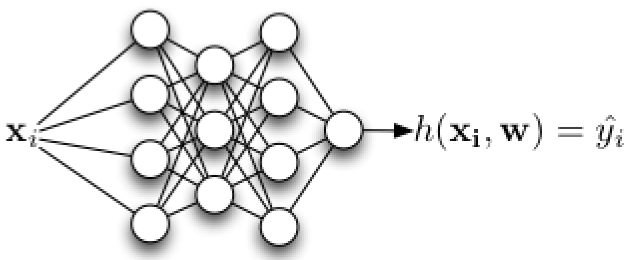
Model Testing

Finally the saved models(TensorFlow) are used for making predictions. Confusion Matrix is used as the Performance Metrics for this classification task.

Training: use labeled $(\mathbf{x}_i, y_i)$ pairs to learn weights.



Testing: use unlabeled data $(\mathbf{x}_i, ?)$ to make predictions.

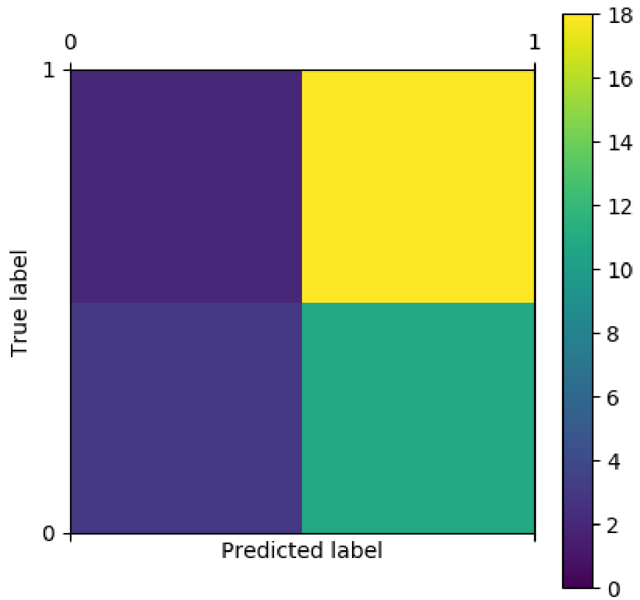


test_model.py

Results

Accuracy : 72%

By Comparing the predicted data we will derive the C-stats




```
test_label (25, 2)
0.72
[[ 0.51496774  0.48503232]
 [ 0.02616028  0.97383976]
 [ 0.30853093  0.69146907]
 [ 0.27975312  0.72024685]
 [ 0.00872291  0.9912771 ]
 [ 0.00301013  0.99698985]
 [ 0.04476063  0.95523936]
 [ 0.01051811  0.98948193]
 [ 0.04137915  0.95862085]
 [ 0.04659028  0.95340973]
 [ 0.00170158  0.99829847]
 [ 0.00313216  0.99686784]
 [ 0.55115074  0.44884923]
 [ 0.22645003  0.77354997]
 [ 0.09592484  0.90407521]
 [ 0.53699112  0.46300885]
 [ 0.0453838   0.95461625]
 [ 0.00129902  0.99870098]
 [ 0.73397863  0.26602137]
 [ 0.0234396   0.97656041]
 [ 0.2832084   0.71679163]
 [ 0.05576651  0.94423348]
 [ 0.04423559  0.95576447]
 [ 0.14274462  0.85725546]
 [ 0.95218134  0.04781863]]
predicted [0 1 1 1 1 1 1 1 1 1 1 1 0 1 1 0 1 1 0 1 1 1 1 1 0]
actual [1 1 0 1 0 1 1 1 1 0 1 1 0 0 1 0 1 1 0 1 1 1 0 1 1]
(array([ 0.,  1.]), array([ 8, 17], dtype=int64))
(9, 25088)
test_label (9, 2)
0.333333
[[ 0.02137722  0.97862279]
 [ 0.21060883  0.78939116]
 [ 0.08265685  0.9173432 ]
 [ 0.08265685  0.9173432 ]
 [ 0.29388231  0.70611775]
 [ 0.07365517  0.92634481]
 [ 0.00508486  0.99491513]
 [ 0.01596139  0.98403859]
 [ 0.35589236  0.64410764]]
predicted [1 1 1 1 1 1 1 1 1]
actual [1 0 0 0 0 0 1 1 0]
(array([ 0.,  1.]), array([14, 20], dtype=int64))
[ 0.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  1.  0.  1.  1.  0.  1.  1.
  0.  1.  1.  1.  1.  1.  0.  1.  1.  1.  1.  1.  1.  1.  1.  1.]
[ 1.  1.  0.  1.  0.  1.  1.  1.  1.  0.  1.  1.  0.  0.  1.  0.  1.  1.
  0.  1.  1.  1.  0.  1.  1.  1.  0.  0.  0.  0.  0.  1.  1.  0.]
[[ 3 11]
 [ 2 18]]
precision :  [ 0.6         0.62068966]
recall :  [ 0.21428571  0.9       ]
fscore :  [ 0.31578947  0.73469388]
support :  [14 20]
```

Total Source Code

As the complete deployable model is around 2 GB .Placed in the **Google Drive** :

https://drive.google.com/file/d/0B7JNoOQgz9GRUFJ6OEFqUVl4SHc/view?usp=sharing

Thank You

Domain: Data Science

Details: Navaneesh Gangala