Using Computer Vision to Detect Pneumonia in Chest X-Rays

CS 221: Artificial Intelligence: Principles and Techniques
Avi Khemani

Problem Statement

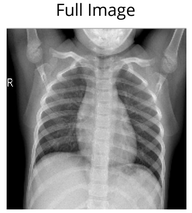
- 450 million cases of pneumonia globally every year → 4 million deaths per year
- Radiologists can diagnose false positives and false negatives - life threatening consequences
- Stanford study in 2018 used AI on chest X-rays to accurately classify 14 diseases[1]
- Another study applied deep learning on radiographs to diagnose unique elbow injuries
- **Objective: use computer vision to accurately detect presence of pneumonia in chest X-ray**

1. https://med.stanford.edu/news/all-news/2018/11/ai-outperformed-radiologists-in-screening-x-rays-for-certain-diseases.html

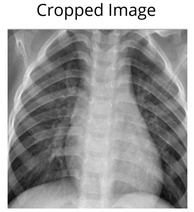
Input-Output Behavior

- Dataset from Kaggle containing 5863 images of chest X-rays that may contain pneumonia
  - 4274 pneumonia, 1589 normal
- Dataset split
  - 80% training set - 4689 images
  - 10% validation set - 587 images
  - 10% test set - 587 images
- Maintained skewed ratio of pneumonia to normal images within each data set
- Resized each image to 200 x 200 pixels to ensure consistency and maintain resolution
- Experimented with two feature vectors
  - Pixel values of full image to ensure no loss of any important information
  - Pixel values of cropped image that focuses only on inflammation of lungs

Full Image



Cropped Image



- The inputs are a flattened array of the 200^2 pixels and the outputs are 0 or 1, where 1 indicates the presence of pneumonia

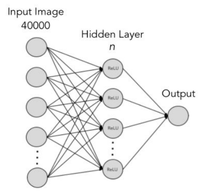
Approach

I have implemented and tested the following algorithms (scikit-learn, PyTorch):

- **Logistic Regression** using SGD to minimize logistic loss function

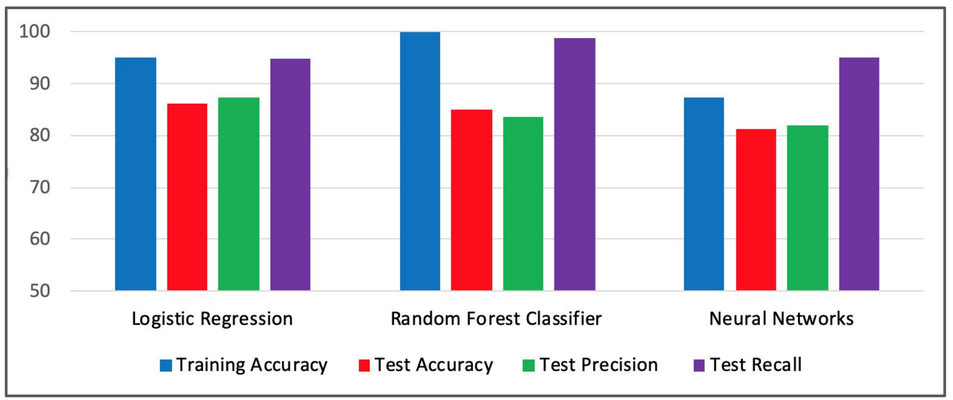
$$Loss(x, y, w) = (\sigma(w \cdot \phi(x)) - y)^2$$

- **Random Forest Classifier**, which uses the average of the outputs of many decision trees that split on strongly influential features
  - Experimented with gini impurity and information gain to decide splits
- **Neural Networks**
  - Maps pixel values to *n* nodes in hidden layer
  - ReLU / Sigmoid activation functions to account for non-linearity in relationship
  - SGD to minimize Cross Entropy / NLL loss
  - Fixed 50 epochs, 0.001 learning rate, 1 batch



Results

- Optimal hyperparameters after tuning on my validation set:
  - Logistic Regression - full image feature, 'optimal' eta
  - Random Forest Classifier - full image feature, gini impurity, 50 trees
  - Neural Networks - full image feature, cross entropy loss, ReLU activation function, 8000 hidden layer size
- Across all models, the full image feature vector performed much better
  - Possibly losing information from crop: inflamed lungs → expanded ribs



- The bar graph compares training accuracy and test accuracy to assess how well the model generalizes to new data
- The graph also shows precision and recall to assess how well the model can avoid false positives and/or false negatives respectively

Analysis

- Logistic regression was the most accurate and precise model - avoids false positives
  - SGD adjusts weights for single data points, which is effective considering small data size
- Random forest had the highest recall
  - Important to avoid chance of false negative
  - Each tree is uncorrelated so likely that one tree will identify feature of pneumonia
- Neural networks had worst accuracy
  - Only one batch and one hidden layer makes it difficult to train to convergence
- Neural networks generalized the best
  - With many nodes, can effectively model true relationship between pixels and pneumonia

Challenges

- Relatively small dataset to train each model
- Working with the imbalance of ratio between pneumonia and normal data points
  - Had to maintain skewed ratio in datasets
- Properly training my neural network - to avoid huge gradients and minimize loss, I had to:
  - Normalize image pixel values
  - Significantly lower learning rate
  - Reduce data imbalance in training
  - Initialize weights with xavier initializers

Future Work

- Train on larger / less skewed data set
- Add more batches & layers to neural network
- Implement convolutional neural networks instead of only linear layers
- Experiment with other feature vectors, such as Canny edge detection (shown to the right)