Multi-Class Disease Diagnosis on Chest X-Ray Images

Adiki, Madhu Babu
madiki3@gatech.edu

Arora, Mansi
marora33@gatech.edu

Sekar, Srinivas
srinivas.sekar@gatech.edu

I. INTRODUCTION

Chest X-rays are often among the first procedures a person undergoes if the doctor suspects a heart or lung disease. A chest X-ray can reveal many things about the condition of lungs. They can be used to help diagnose pneumonia, heart failure and other heart problems, emphysema or cystic fibrosis lung cancer, fluid or air collection around the lungs and other medical conditions. However, detecting pathologies in chest X-rays is a challenging task that relies on the availability of an expert radiologist. It is challenging because, the appearance of the disease in the image is often vague, can overlap with other diagnoses and can mimic many other benign abnormalities. Moreover, there remains a discrepancy and considerable variation in the interpretation of the X-rays among radiologists. Moreover, the diagnosis of a chest X-ray by a human doctor can be very costly in terms of money and time. Not only that, a lot of developing countries dont have access to expert radiologists, or the ratio of a doctor to patient is extremely low. We hope to alleviate this problem for millions of people across the world, by presenting an automated diagnosis system that would speed up processing, reduce effort from clinicians, reduce errors, and make X-rays more practical for diagnoses that currently rely on more expensive methods.

II. LITERATURE REVIEW

Several attempts have been made in past to diagnose medical conditions using chest X-rays using Convoutional Neural Networks, both before [8] and after the NIH dataset was released. All of them frame the task as a supervised multi-class classification problem; with some of them [1][4] extending the method to determine the localised areas in the X-ray that are most indicative of the pathology. Given the size of the dataset, popular methods include use of pre-trained image classification architectures such as GoogLeNet and AlexNet with some additional layers and use of DenseNet architecture [2][9]. To tackle the class imbalance issue, the literature suggests multiple ways such as using GANs (Generative Adversarial Network) [3] to generate new data, applying balancing weight factors [1] to enforce learning of positive examples and augmenting positive examples after manipulating images by rotating, translating and scaling [5]. Other pre-processing techniques such as contrast enhancement [6] and bone-shadow removal [7] have also been tried.

III. METHOD

A. Exploratory Data Analysis

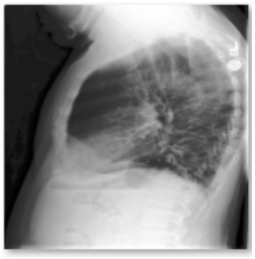
The NIH dataset is the largest publicly available chest x-ray dataset. The dataset comprises 112,120 X-ray images of 30,805 unique patients with the text-mined fourteen disease image labels (where each image can have multi-labels). The image labels are mined from the associated radiological reports using natural language processing. Fourteen common thoracic pathologies include Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule, Mass and Hernia.

The data is in the form of images in 1024x1024 resolution. The data also consists of meta data for all images such as text labels of all diagnosed diseases from the image, Follow-up number, Patient ID, Patient Age, Patient Gender, View Position, Original Image Size and Original Image Pixel Spacing. Some of the limitations of the dataset are that the image labels are NLP extracted so there would be some erroneous labels but the NLP labelling accuracy is estimated to be $\geq 90\%$. Also, there are very limited numbers of disease region bounding boxes. We also made a co-occurrence

matrix to understand class-prevalence of our 14 different disease types which is shown below. We can see from the co-occurrence matrix that Hernia has the lowest frequency of occurrence of the 14 diseases (225 cases). Also, there is noticeable co-occurrence between Effusion and Infiltration, and between Effusion and Atelectasis.

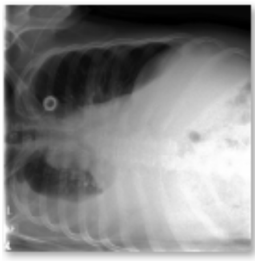
B. Data Pre-processing

The first step in preprocessing our data is to remove images that fall into the following 4 categories:
1. Images with a bounding box drawn around the disease area
2. Images with side x-rays or other non-frontal X-rays
3. Images that are rotated by more than 15 degrees from the vertical
4. Images pre-selected by the authors as being low quality
The figures below show a couple of sample images of images in these blacklisted categories.



00005260_000.png

Fig. 1: Non-Frontal view



00017606_037.png

Fig. 2: Rotated View

Then we consolidated the training and test sets and also created a validation set in the ratio 7:1:2 for train:val:test. Here, we ensured that all images belonging to a particular patient remain in one of the train/validation/test partitions and are not shared between these sets. We also converted the labeled disease variable to a multi-hot-encoding which would be our response variable in the model.

C. Image Processing

Apart from data-preprocessing, we also have to process images to feed into the Densenet Model. Below are the main steps in image processing as they are fed into the model:
1. Downsize the image to 224x224 pixels. This is done because smaller images will train significantly faster, and possibly even converge quicker (all other factors held constant) as we can train on bigger batches (e.g. 50-1000 images in one pass based on CPU and GPU capacity, which we might not be able to do with higher resolution images).
2. Normalize based on the mean and standard deviation of the ImageNet training set. This is hard-coded in our function. This is done to ensure faster model convergence. Also, since deep learning networks traditionally share many parameters, sharing wouldn't happen very easily because to one part of the image weight is a lot and to another it's too small.
3. We augment the training set by randomly flipping the images horizontally, by randomly rotating the images by up to 15 degrees, and by randomly rescaling the images by a factor from 0.8-1.1. Data augmentation is a way in which we can reduce overfitting our models while increasing the amount of training data that can be fed into the model.
In our approach to processing the images, we initially used pyspark with the sparkdl library. However, given that we will still have to read all the images and feed to our GPU on the fly, we found that we were quite severely bottlenecked on the GPU. Hence, to reduce processing times (pyspark was quite slow to run) and fully utilize our CPU as well as the GPU, we decided to use pytorch's built in dataloader functionalities to perform image pre-processing on the fly.

D. Model Development

Data Loading: Since the size of the dataset is large ($\approx$ 45 GB), we need a method to read a batch of images on the fly, perform the above mentioned data transformation steps, and feed into the network. For this, we used the Dataset and Dataloader classes of PyTorch utils.data library.

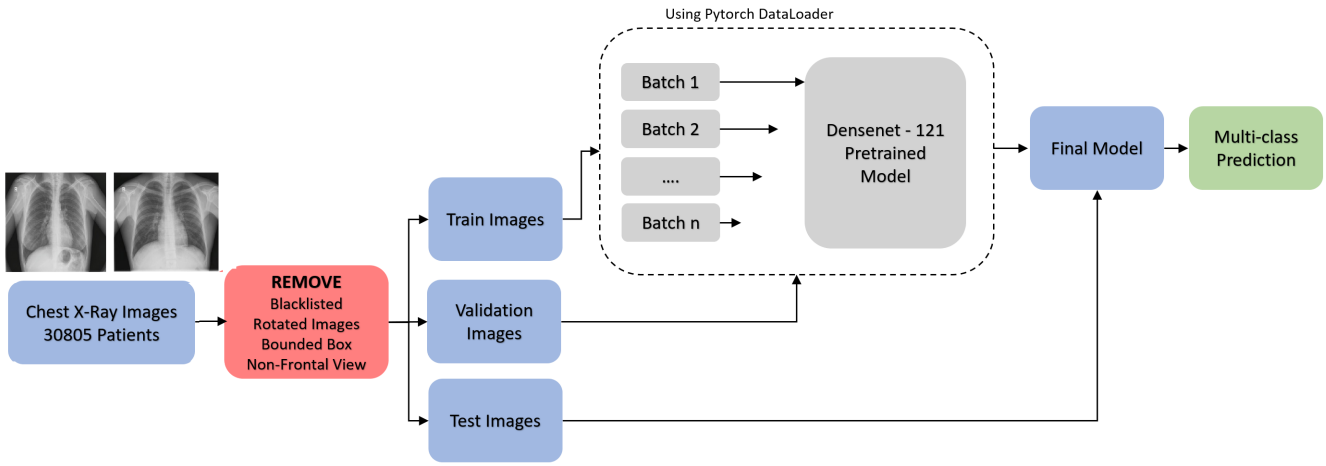


Fig. 3: Model Development Pipeline

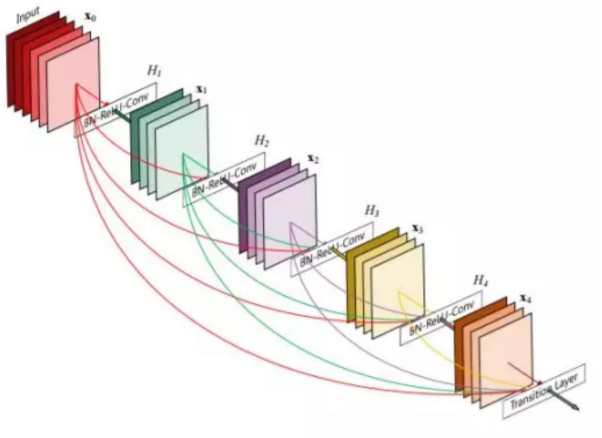


Fig. 4: Densenet-121 architecture

This is memory efficient because all the images are not stored in the memory at once but read as required. We defined a Dataset class for our X Ray dataset. This class takes an optional argument transform so that any required processing can be applied on the sample.

Training: We used the DenseNet-121 architecture with pre-trained weights from ImageNet as initialization parameters. Densenet121 has four dense blocks, which have 6, 12, 24, 16 dense layers. This allows us to both pass the gradient more efficiently and train a deeper model and the architecture alleviates the vanishing-gradient problem and enables feature map reuse, which makes it possible to train very deep neural networks. We add a fully connected sequential layer with 14 nodes to DenseNet's 1000 node output and the activation is through a sigmoid function in each of the nodes. This made sure that the output layer gives the probability of each of the 14 classes. For implementation, we included parallelization of the model training process, for faster processing. Our final dataset is 112,120 images and we trained our model for 50 epochs.

Parameters:

- Optimizer: Adam optimizer with learning rate 0.0001. β ranges from 0.9 to 0.999.
- Loss function: Binary Crossentropy Loss
- Batch size: 128

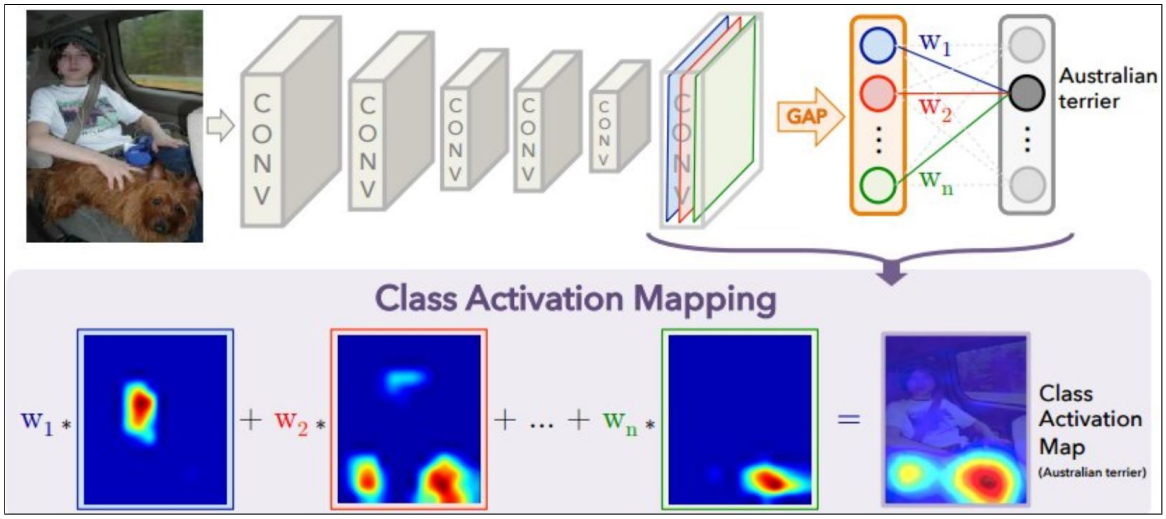


Fig. 5: Concept of Localization

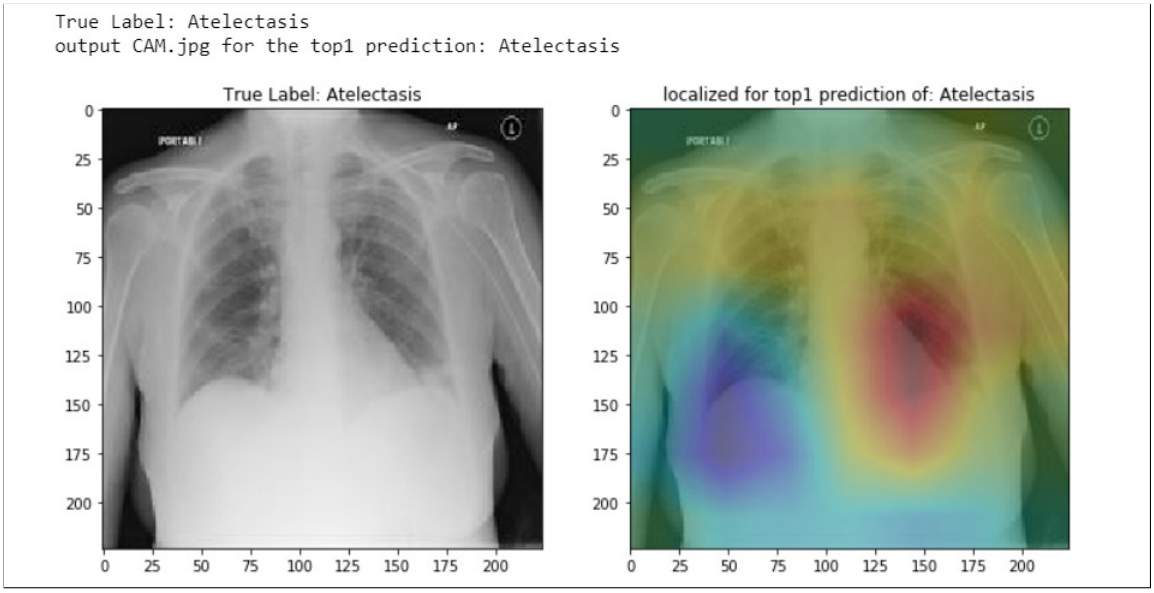


Fig. 6: Localization on an X-ray with Atelectasis

E. Computing Infrastructure

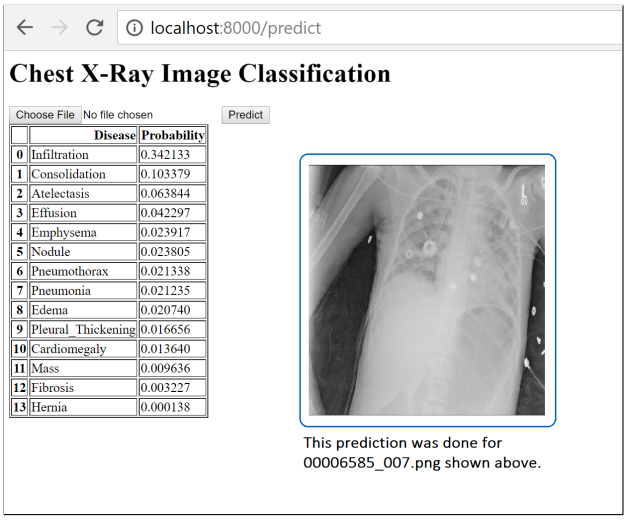
We experimented with various infrastructure tools to run our models. We first ran a few epochs with few images in our personal computers to make sure the code ran without errors. We experimented with using AWS instances and Crestle.com for training the model with the full dataset. We found that the Crestle.com GPUs (one NVIDIA K80) took the longest time per epoch at around 850s per epoch. We used a P3.2xlarge instance with a V100 GPU on Amazon AWS and were able to achieve a per epoch training time of around 450s. However, we noticed that there was a little bit of bottlenecking at the CPU, since there were only 8 vCPU units in this instance. We finally used Georgia Tech's Eclipse7 server to train our final model. The system was equipped with two NVIDIA P40 GPUs and a much more powerful 28 core CPU. This meant that we could run a multi-gpu model and still not be bottlenecked at the CPU level for larger batch sizes. With this setup, we were able to get to 300s per epoch, leading to 10 hours of training for 50 epochs. We used 24/28 CPU cores to perform dataloading.

F. Localization

We also attempted Localization on chest X-ray images. This technique is used to generate class activation maps, which could be used to interpret the prediction decision made by the CNN such that the CNN is triggered by different semantic regions of the image for different predictions using the global average pooling in CNNs. Figure 6 shows the localization (red zone) on an X-ray which has the Atelectasis disease. Atelectasis is the collapse or incomplete expansion of the lung or part of the lung. As we can see, this is evident in the X-ray and the algorithm was successfully able to identify the region where it occurs.We used code written in Pytorch provided by CSAIL team at MIT.

G. Web interactive tool

The goal was to have an interface that can take an image input from the user, pre-process the image i.e. resize, and normalize. This image is then tested with our trained model. The model outputs the predicted probabilities for all 14 classes, which is displayed on the web page. On the backend, Flask is used as a webserver to route webpage traffic. After the image is uploaded, the HTML calls the python function to preprocess the image. Currently, our front end displays a table of disease probabilities and in the future, we could create a visualization for the graphical representation of the results, as well as displaying the uploaded image on the webpage.



	Disease	Probability
0	Infiltration	0.342133
1	Consolidation	0.103379
2	Atelectasis	0.063844
3	Effusion	0.042297
4	Emphysema	0.023917
5	Nodule	0.023805
6	Pneumothorax	0.021338
7	Pneumonia	0.021235
8	Edema	0.020740
9	Pleural_Thickening	0.016656
10	Cardiomegaly	0.013640
11	Mass	0.009636
12	Fibrosis	0.003227
13	Hernia	0.000138

Fig. 7: Demo of web interactive tool

IV. Model Results and Discussion

Table 1 below shows the AUC accuracy comparison of our model with previous benchmarks Wang et al, and Rajpurkar et al. As we can see our results are on par with that of Wang et al, however has lower accuracy as compared to Rajpurkar et al. One of the reasons could be overfitting, as we achieved an average AUC score of 84.0% and 73.8% on the validation set and test set respectively. Hence, in the future, we would use regularization methods such as dropout and L2 regularization to the cost function to prevent overfitting. Also, as we can see in figure 7, we can see that the test data AUC of all diseases is well above the baseline.

Disease	Wang et al	Rajpurkar	Our Model
Atelectasis	0.7158	0.8209	0.7189
Cardiomegaly	0.7843	0.8831	0.7709
Consolidation	0.7078	0.7939	0.7219
Edema	0.8345	0.8932	0.8111
Effusion	0.7843	0.8831	0.7973
Emphysema	0.8149	0.9260	0.7559
Fibrosis	0.7688	0.8044	0.7634
Hernia	0.7667	0.9387	0.7429
Infiltration	0.6089	0.7204	0.6779
Mass	0.7057	0.8618	0.7307
Nodule	0.6706	0.7766	0.6754
Pleural Thickening	0.7082	0.8138	0.7095
Pneumonia	0.6326	0.7632	0.6681
Pneumothorax	0.8055	0.8932	0.7900

TABLE I: AUC Accuracy comparison of our model with previous benchmarks

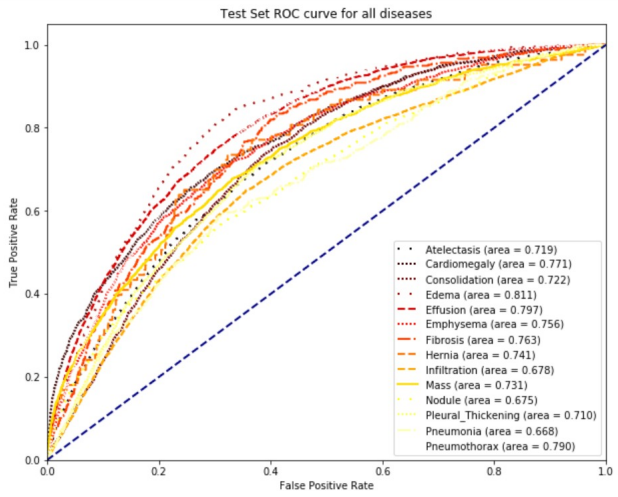


Fig. 8: Test AUC for each disease

REFERENCES

[1] Wang, Xiaosong, et al. ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017, doi:10.1109/cvpr.2017.369.

[2] P. Rajpurkar, J. Irvin, K. Zhu, B. Yang, H. Mehta, T. Duan, D. Ding, A. Bagul, C. Langlotz, K. Shpanskaya, M. P. Lungren, and A. Y. Ng. Chexnet: Radiologist-level pneumonia detection on chest x-rays with deep learning. arXiv preprint arXiv:1711.05225, 2017.

[3] Salehinejad, Hojjat, et al. Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks. Generalization of Deep Neural Networks for Chest Pathology Classification in X-Rays Using Generative Adversarial Networks, (IEEE ICASSP), 2018, 12 Feb. 2018, arxiv.org/abs/1712.01636.

[4] Li, Zhe et al. Thoracic Disease Identification and Localization with Limited Supervision. CoRR abs/1711.06373 (2017): n. pag.

[5] Yao, Li, et al. Learning to Diagnose from Scratch by Exploiting Dependencies among Labels. Learning to Diagnose from Scratch by Exploiting Dependencies among Labels, ArXiv:1710.10501, 1 Feb. 2018, arxiv.org/abs/1710.10501.

[6] Tataru, Christine A. et al. Deep Learning for abnormality detection in Chest X-Ray images. (2017).

[7] Gordienko, Yuri et al. Deep Learning with Lung Segmentation and Bone Shadow Exclusion Techniques for Chest X-Ray Analysis of Lung Cancer. CoRR abs/1712.07632 (2017): n. pag.

[8] Dong, Yuxi, et al. Learning to Read Chest X-Ray Images from 16000 Examples Using CNN. 2017 IEEE/ACM International Conference on Connected Health: Applications, Systems and Engineering Technologies (CHASE), 2017, doi:10.1109/chase.2017.59.

[9] Cortana Intelligence and ML Blog Team. Using Microsoft AI to Build a Lung-Disease Prediction Model Using Chest X-Ray Images. Machine Learning Blog, 7 Mar. 2018