Chest X-Ray (CXR) Disease Diagnosis with DenseNet

Doug Beatty, Filip Juristovski, Rushi Desai, Mohamed Abdelrazik
Georgia Institute of Technology, Atlanta, Georgia

Abstract

Chest X-ray imaging[1] *is a crucial medical technology used by physicians to diagnose disease and monitor treatment outcomes. Training a human radiologist is a lengthy and costly process. Deep learning techniques combined with availability of larger data sets increases the feasibility of building automated models with performance approaching human radiologists.*

We present a scalable deep learning model trained on the ChestXray14[7] *data set of X-ray images to detect and correctly classify presence of pneumonia. We plan to expand the model to classify 14 different diseases.*

Introduction

A chest radiograph[1], or a chest X-ray (CXR) is one of the oldest and most common forms of medical imaging. A human radiologist requires significant training time and cost to be able to perform a comprehensive chest X-ray analysis with minimal error. Several types of abnormalities can show up in a chest radiograph that can lead to detection and diagnosis of several kinds of diseases but due to the vast number of different abnormalities and the overlapping reasons that might cause them, theres plenty of room for human error.

The revolution of machine learning and deep learning techniques combined with the availability of larger data sets[2] and big data processing systems[3] makes the analysis of X-ray images increasingly more realistic and the creation of automated models more feasible. The objective of this project is to train an efficient and scalable deep learning model which can learn from a data set of X-ray images to detect and correctly classify 14 different diseases. Automating the X-ray analysis makes the overall diagnosing process faster and less error-prone which significantly improves the patients treatment procedure.

Approach

1. Data set acquisition
2. Image preprocessing - Apache Spark
3. Training DenseNet-121 deep learning model - Keras
4. Model validation and fine tuning
5. Model evaluation

Data acquisition

We acquired two different datasets of chest radiographs. The first is ChestXray14 from the NIH and all results from this paper are using this dataset. The second is CheXpert which we plan to use for our final results.

The CheXpert dataset was acquired upon registration and acceptance of the Stanford University School of Medicine CheXpert Dataset Research Use Agreement terms and conditions.[2]

Dataset format

The full ChestXray14 dataset consists of 112,120 chest radiographs of 30,805 patients. We are using this dataset for our initial training and results.

CheXpert consists of 224,316 chest radiographs of 65,240 patients. Each imaging study can pertain to one or more images, but most often are associated with two images: a frontal view and a lateral view. Images are provided with 14 labels derived from a natural language processing tool applied to the corresponding free-text radiology reports.

Image files are provided following a specific directory structure:

```
/[Data set type]/[Patient ID]/[Study ID]/[View ID]_[view type].jpg
```

where data set type can be train (for training set) or valid (for validation set), and view type can be frontal or lateral.

e.g. an input sample of a frontal study for a patient in the training set will be available at:

```
CheXpert/train/patient00001/study1/view1_frontal.jpg
```

Each image is stored as a .jpg file with single channel (grey scale) where each pixel is stored as unsigned byte.

A CSV file is provided for each data set type (validation or training). Each image record contains a file path, labels for different diseases (such as Cardiomegaly, Edema, Pneumonia, and so on), along with some demographic information about the patient such as sex and age.

Dataset Pre-Processing

CheXpert[2] images are provided in high resolution which is not suitable for use as input to the model. Using a high resolution image significantly increases the number of input feature vectors which would require an increase in the model complexity and training time. Also, our strategy is to use pre-trained DenseNet with ImageNet weights, so we need to use the same number of input features. Data set images where preprocessed before training using Apache Spark which is a scalable big data processing technology. Several down-sampling techniques were used to reduce image size.

A convolution neural network (CNN) is said to have an invariance property when it is capable to robustly classify objects even if its placed in different orientations. To enrich the input data set and increase the number of available training samples, we performed data augmentation by generating several images with different orientations from a subset of input images.

Each input image is down sampled by resizing to 224 by 224 pixels. An input image generates one or more augmented versions of itself (e.g. by horizontal flipping). Each output image is assigned an ID of type Long, and inherits all the labels from the original input image.

Pre-processed images are saved to HDFS, which is a highly distributed and scalable Big data storage system. Due to HDFS implementation and API limitations, storing several tiny image files is not an efficient operation. We decided to change the output format to be textual. Each image can be represented with the unsigned values of its byte stream (a vector of length $224 \cdot 224 = 50,176$). Each Spark RDD Partition will save its images as a space separated file with this format:

```
[Image ID] [bytes values]
```

where Image ID is the newly assigned ID, and the bytes are space separated vector of row based pixel values.

A corresponding CSV file for labels is generated which starts with the Image ID along with all the inherited labels from the original image.

This new output storage format resulted in significant decrease of the pre-processing job run time, and was suitable as a direct input to the model.

Metrics and Experimental Results

We will use AUC to determine performance of our results. We will compare it with the previous best performing results.

To guide our training and see if we are on track, we will use loss curves and model accuracy plots. This will help us diagnose if our model is over-fitting, under-fitting etc. From our initial training, we have the loss curve as shown in Figure 1.

The loss curve indicates that we might have an issue with high variance. For this loss curve, we took a pre-trained

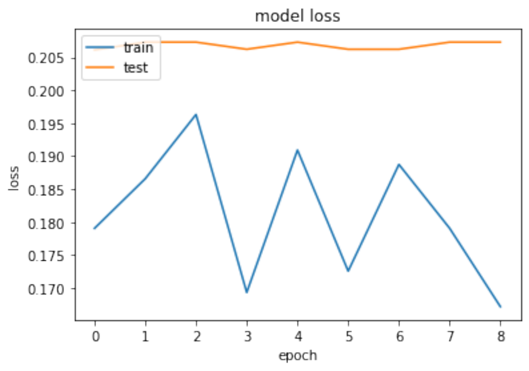


Figure 1: Model loss

DenseNet with ImageNet weights and added an additional 3 fully connected layers. We then froze all layers and trained only the additional layers. Once the additional layers were trained, we then unfroze a total of 9 layers including the newly added ones and went through training again.

The loss curve suggests trying approaches like procuring more data, training deeper into architecture by unfreezing more layers etc.

Figure 2 shows the accuracy curve. At this point we aren't getting a monotonic trend in our accuracy curve so we will continue to investigate methods to improve the accuracy. We also see that like model loss, accuracy also is performing poorly on the test set as compared to the training set. So we will focus on getting both training loss lower and accuracy higher and bring train and test metrics as close as possible.

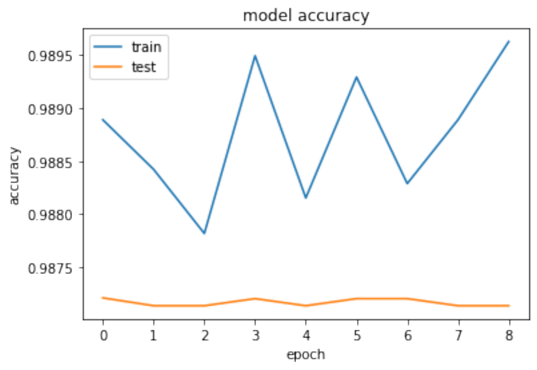


Figure 2: Model Accuracy

Discussion

Dense Convolution Networks (DenseNets) are a form of residual networks. Residual networks allow us to train much deeper networks than conventional CNN architecture since they handle the vanishing/exploding gradient problem much more effectively. Theoretically, it is expected that performance of models should increase as architecture grows deeper, and we should get monotonically decreasing performance. But in reality we don't see that since as the layers get deeper, the optimizer finds it more and more difficult to train the network due to the vanishing/exploding gradient problem. ResNet allow us to match the expected theoretical issue. Figure 3 shows the depth vs performance characteristic.

ResNets however have a lot of parameters compared to conventional CNN networks. DenseNet retains all features of ResNet and goes further by eliminating some pitfalls of ResNet. DenseNets have much less parameters to train compared to ResNets (typically up to 3x less parameters). The base model we are using is DenseNet-121 with pretrained

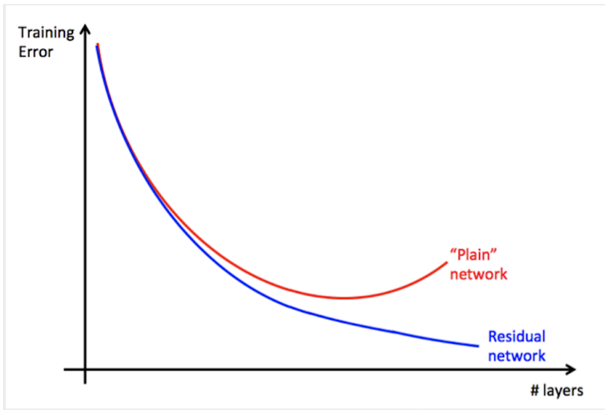
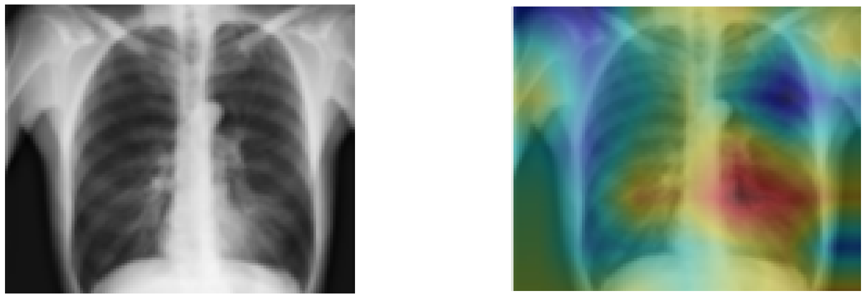


Figure 3: Model Accuracy



(a) Source X-ray image **(b)** Heatmap of pneumonia detection positive (85%)

Figure 4: CAM heatmap[5]

weights from ImageNet. For feature extraction purposes, we load a network that doesn't include the classification layers at the top.

Currently we are only focusing on improving performance on the pneumonia disease class and later we will expand to other diseases. This is to reduce complexity and to focus on producing a working model. Adding other diseases will mainly comprise of changing the classification layer and retraining on the dataset with all labels included. Currently we are using only 33,000 images for training. Using a sampled dataset allows us to iterate faster and focus on removing bugs etc in our model and training code.

Our main theory for getting poor results is due to a class imbalance of cases of pneumonia (1430 cases out of a total of 112,120 images; only 1.2%). We hypothesize that re-balancing with a combination of data augmentation and sub-sampling the data set for non-pnuemonia cases will improve results. Data augmentation methods include flipping images on horizontal and vertical axes, etc. We also used a phased approach to training where we kept most of the layers frozen during initial training and subsequently unfroze more and more layers. This helped in debugging and gave a sanity check during development.

Conclusion

Although the performance of the model is below par currently with unsatisfactory AUC, we feel we have a solid pipeline to start from and a strategy for improvement. Next we will re-balance the dataset and tune the model to get better performance. We also intend to create class activation maps (CAM) heatmaps for better visualization as seen in Figure 4.

References

1. Siamak N. Nabili, M. (2019). Chest X-Ray Normal, Abnormal Views, and Interpretation. [online] eMedicine-Health.
2. CheXpert: A Large Dataset of Chest X-Rays and Competition for Automated Chest X-Ray Interpretation. [Internet]. Stanfordmlgroup.github.io. 2019.
3. FAN M, XU S. Massive medical image retrieval system based on Hadoop. Journal of Computer Applications. 2013;33(12):3345-3349.
4. Huang G, Liu Z, van der Maaten L, Weinberger K. Densely Connected Convolutional Networks [Internet]. arXiv.org. 2019.
5. Rajpurkar P, Irvin J, Zhu K, Yang B, Mehta H, Duan T et al. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning [Internet]. arXiv.org. 2019.
6. Liu H, Wang L, Nan Y, Jin F, Pu J. SDFN: Segmentation-based Deep Fusion Network for Thoracic Disease Classification in Chest X-ray Images [Internet]. arXiv.org. 2019.
7. Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers R. ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. 2019.
8. Yao L. Weakly supervised medical diagnosis and localization from multiple resolutions [Internet]. Arxiv.org. 2019.
9. Guendel S, Grbic S, Georgescu B, Zhou K, Ritschl L, Meier A et al. Learning to recognize Abnormalities in Chest X-Rays with Location-Aware Dense Networks [Internet]. arXiv.org. 2019.
10. Raoof S, Feigin D, Sung A, Raoof S, Irugulpati L, Rosenow E. Interpretation of Plain Chest Roentgenogram. 2019.
11. Zhou B, Khosla A, Lapedriza A, Oliva A, Torralba A. Learning deep features for discriminative localization [Internet]. Arxiv.org. 2019.
12. http://www.stat.harvard.edu/Faculty_Content/meng/JCGS01.pdf
13. Pryor TA, Gardner RM, Clayton RD, Warner HR. The HELP system. J Med Sys. 1983;7:87-101.
14. Gardner RM, Golubjatnikov OK, Laub RM, Jacobson JT, Evans RS. Computer-critiqued blood ordering using the HELP system. Comput Biomed Res 1990;23:514-28.