Deep Learning Application on Automating Chest Diseases Detection from Radiographs

Theophane Godonou, Vamsi S Jandhayala, Orcun Kurugol, Phong Le

Abstract

In this paper, we demonstrated an ETL pipeline to store and use X-ray image information and classify 14 of the most common diseases found in the images. We used a simple CNN model and trained it with a limited dataset to form our baseline classification which resulted in 63% average validation AUC score while the PyTorch DenseNet-121 and ResNet-50 models we utilized using knowledge transfer and fine-tuning resulted in nearly 70% AUC scores. We demonstrated that using higher resolution images improves the AUC score while the fine-tuning improves it only marginally.

Our code is available on Github here
Our presentation slides are available here
Our video presentation is available here

Introduction

The detection of diseases from medical imaging is one of the most complex, expensive and time-consuming tasks. Radiologists receive years of training and they rely on patient history and demographics to identify diseases correctly. X-rays are the oldest form of medical imaging. A reliable disease classification method for x-rays would decrease the cost and time spent on them while improving the accuracy. Emergence of deep learning and big data tools along with big datasets enabled image classification algorithms. Recently developed methods of chest x-ray classification involve utilizing big data methods for extracting image features and utilizing deep convolutional neural networks to perform image classification.

Wang et al. [1] presented ChestX-ray8 which contains 108948 X-ray images of 32717 patients. They labelled the images with eight of the most common thoracic pathologies using text mining on the associated radiological reports. Their Deep Convolutional Neural Network architecture is a weakly-supervised multi-label image classification and pathology localization framework. Annarumma et al. [2] introduced a method of using NLP on diagnosis reports with two CNN models on chest radiographs to help prioritize on urgency which leads to reduction in wait time on critical cases. Irvin et al. [3] CheXpert dataset contains 224,316 chest radiographs of 65,240 patients. They managed to capture the presence of 14 labels and used different approaches with convolutional neural networks to predict the probabilities of those observations. Many approaches have been explored to achieve successful classification of the chest x-rays using convolutional neural networks architectures. Li et al. [4] presented a method to predict the probability of fourteen different Chest diseases by using CNN on a subset of both annotated regional level and unannotated image level. Their method consists of using ResNet then slicing patches to resize the convolutional features into PxP grid from ResNet before being passed into another CNN. Rajpurkar et al. [5] introduced a convolutional neural network algorithm trained on NIH Chest X-ray dataset. They replaced the final layer of the DenseNet-121 architecture with a sigmoid function for classification of pneumonia. They then extended it for other diseases using a 14 vector for the output. Guan et al. [6] introduced a category-wise residual attention learning (CRAL) framework for identifying diseases from x-ray images. They use their methods on both DenseNet-121 and ResNet-50 architectures. Liu et al. [7] implemented a segmentation based deep fusion network (SDFN) for Thoracic Disease Classification in Chest X-ray Images. They use the Chest X-ray 14 dataset and the JSRT Dataset containing 154 nodule and 93 non-nodule CXR images. Baltruschat et al. [8] implemented a 5-fold re-sampling scheme, weight initialization using pre training and transfer learning, ResNet-50 network architecture, and non-image features. They treat detection as a multi-label classification problem to minimize the class-averaged

binary cross entropy loss function. Zongyuan Ge et al. [9] introduced a novel error function Multi-label Softmax Loss MLSL to capture the characteristics of Multi-label Learning. They combined it with bilinear pooling to propose a mechanism for multi-label learning. They applied it to the ChestX-ray14 dataset.

In this project, we formed our baseline with a simple CNN design and a limited data. We then used knowledge transfer (KT) from pre-trained ResNet-50 and DenseNet-121 models by changing the last layer to output 14 classes and training them with x-ray images using binary cross entropy (BCE) loss. The models are using colored images of arbitrary size as input and a 14 vector as output. Once the best models were chosen based on the validation AUC score, those models were fine tuned by unfreezing all layers and training with the x-ray images again. We were able to train multiple models accepting different images sizes in parallel due to our ETL framework and compared the results.

Data

We chose the NIH Chest X-ray Dataset to use for our project. This dataset has been used in numerous previous works to build models that can detect diseases in X-ray images. It is composed of 112120 images of resolution 1024 x 1024 pixels from 30805 unique patients. The diseases mentioned in the radiological report attached to each image were extracted using text-mining techniques with an accuracy of more than 90%. Fifteen classes are identified with fourteen diseases plus a no findings class. The fourteen diseases are among the most common diseases identified using Chest X-ray images.

The dataset contains 14175 female patients and 16630 male patients. The patients have various numbers of images. Many have only one image and one of the patients has up to 184 images. Their ages vary between 0 and 95 years old, with an average of 46 years. The patients' X-rays images may be collected in numerous follow ups to the initial visit. On average, the patients have 8.57 follow-ups with the maximum being 15.

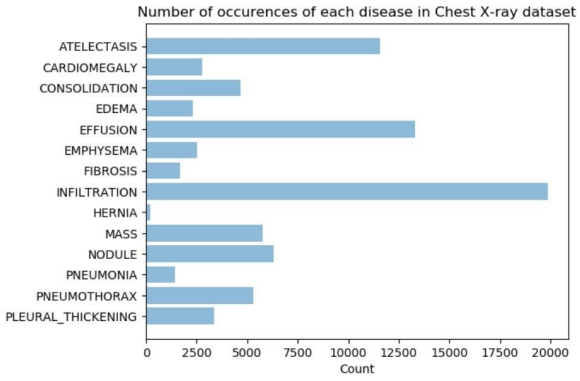

Figure 1 : Number of occurences for each disease.

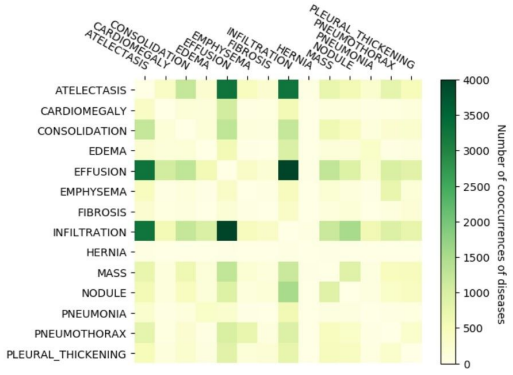

Figure 2: Heatmap of cooccurences of diseases with only the multilabel

Figure 1 shows the distribution of the various diseases in the dataset. We can see that Hernia is the least frequent and Infiltration occurs the most. Figure 2 shows the co-occurrence of diseases in the same image. We see that Infiltration and Effusion are more likely to occur together while Edema and Hernia are less likely to co-occur. Many patients have multiple diseases at the same time with one having nine of the fourteen. The dataset also includes bounding box information for a limited number of images to identify the relevant area on the X-ray images.

Approach Method

With NIH Chest X-ray repository of over 112,000 colored images of size 1024x1024, we have faced significant challenges in processing data for training our models. As such, we had to strategically design a data flow that would balance the infrastructure requirement while still maintaining the accuracy of the models. In addition, an appropriate technology that would enable vertical scaling was also essential to our process.

The backbone of our research relies on Azure Databricks (Databricks) and Azure Data Lake Storage (ADLS). We began our process by relocating the Chest-Xray repository to ADLS. Using the template code provided by Microsoft [10], we uploaded all images before mounting ADLS into our Databricks clusters.

a. Image Transformation

One of the challenges we encountered was to efficiently transform (i.e. resizing, normalizing, flattenning) the images for training multiple models. We could perform the transformation process on the fly during the training process. However, with the size of the dataset, it would be inefficient to repeat this transformation step which would also result in longer training time.

As such, we utilized Spark as our big data technology to pre-process these images and saved them into parquet files. Doing this enabled us to simply load the parquet files and train multiple models in parallel without transforming the images again.

To achieve this goal, we set up a Databrick cluster with one driver and eight workers (14GB RAM for each node). In our notebook, we used Torchvision to downsample the images into two sizes (512x512x3 and 256x256x3) and normalized those images before flattening into arrays of features. With the support of parallel processing, we could efficiently distribute the workload across eight workers and significantly reduce the transformation time (at least by 70% when we only used one driver).

We also split the data based on patients instead of images. This helped us to reduce the bias of training and validating on different images that belong to the same patients. For training and validation, we used a ratio of 70:30 based on patient id, and for testing, we used all the images mentioned in the test patients list.

The output of our transformation for each image size (512x512 or 256x256) was 27 parquet files (14 for train, 6 for validation, and 7 for test). Each parquet file was partitioned into 4-8 smaller files with the total size of 1GB to 4GB (depending on the image size). The schema of our files contained image id, an array of response variables, and an array of normalized pixel features (length of 196,608 or 786,432 depending on image size)

b. Deep Learning Models

We set up a Databrick cluster with 2 GPUs for training 256x256 images and another one with 4 GPUs for training 512x512 images. Since we already transformed all images and saved them in parquet format, we could load this data back to our clusters for training. The next challenge we encountered was the memory limitation as we could not load the entire transformed dataset to memory. As such, we iterated through all parquet files for training, validation and test sets. This helped us greatly in keeping the data in limited memory.

Once the transformed data was loaded as Spark dataframe, we converted them to PyTorch data loader using Petastorm for each parquet file. This improved our data loading time compared to using Pandas as Databricks was optimized for loading parquet files. Each batch is converted to m x 3 x n x n tensor during the training and evaluation steps where m is the batch size and n is the image resolution.

In addition to using a simple CNN model as a baseline, we used pre-trained PyTorch ResNet-50 and DenseNet-121 models as our starting point, and utilized knowledge transfer (KT) and fine tuning methods to train them. The ResNet model uses residual layers where a skip layer connection is used to address the vanishing gradient problem. It has a higher number of feature channels in each layer than DenseNet, resulting in a higher number of trainable parameters. On the other hand, DenseNet has dense connections between the dense layers connecting each dense layer to every other dense layer which enables it to be more compact than ResNet and address the vanishing gradient problem more directly.

After loading the pre-trained networks, we replaced the fully connected output layer with our own fully connected classification layer, producing an output vector of size 14. We then applied BCE loss function with positive weights calculated based on the ratio between positive and negative labels for optimization. The training happens in 2 stages as shown in Figure 3. In the first stage, the pre-trained CNN layers are frozen and only the output layer is trained using our training data with a batch size of 5. Once the best model is found using the validation AUC score, we unfreeze all layers and train the whole network again with the same training data and batch size, and choose the best model again using the validation AUC score. The reasoning about this two stage training is further explained in the discussion section. Utilizing our ETL framework, we trained lower and higher resolution KT and fine-tuned models using ResNet-50 and DenseNet-121 in parallel and compared their results as well as how they fared against our baseline.

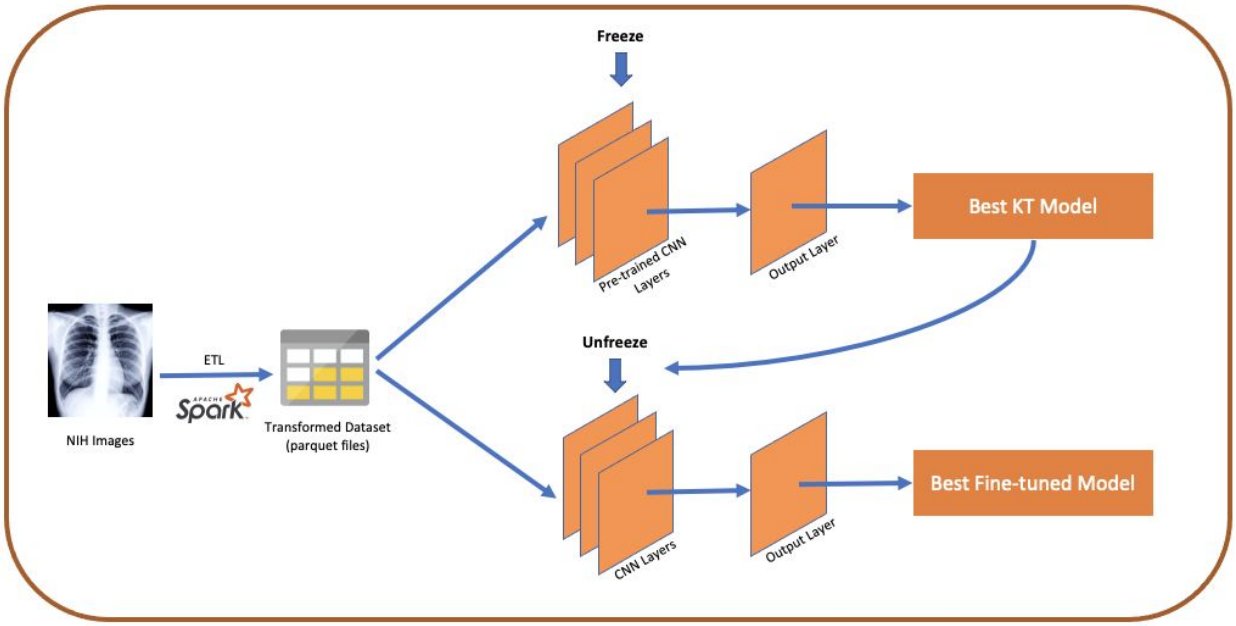

Figure 3: Implementation Architecture. After transforming the images in our ETL framework, knowledge transfer is utilized to train ResNet-50 and DenseNet-121 layers by replacing the output layer. Once the best KT model was found, all layers were unfrozen and the model was further trained for fine-tuning.

Experimental Results

The simple CNN, the ResNet-50, and the DenseNet-121 architectures were evaluated based on loss per epoch and Area Under the Curve (AUC) score from a Receiving Operating Characteristic (ROC) curve. Figure 4 shows the loss scores per epoch for 8 epochs for the DenseNet-121 model trained on 512 x 512 pixel images for both pre-trained knowledge transfer (KT) based and the fine-tuned versions on the training and validation datasets. The

training dataset for the DenseNet-121 KT based model approached losses of around 1.39 for training and 1.25 for validation while the fine-tuned DenseNet-121 model had losses of 1.25 and 1.33 respectively. Thus, in Figure 5 the loss for the training set for both fine-tuned and KT models approached 1.39 with increasing number of epochs while the loss for the validation set for the KT model decreased and was lower than the fine tuned model after 8 epochs.

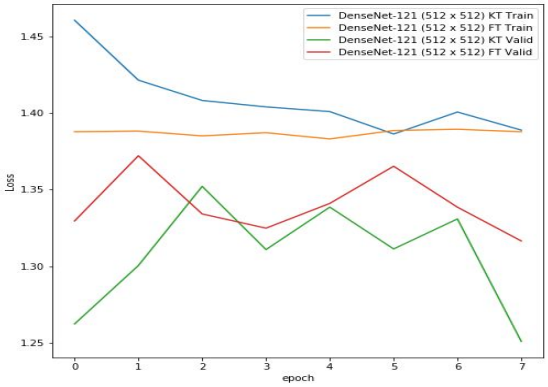

Figure 4: Loss per epoch for 8 epochs

The evaluation metric we used for our model is the AUC score for each disease on the testing dataset and the average AUC score was computed across all diseases. ROC curves for the 14 diseases with the true positive rate versus false positive rate were plotted and the AUC score was computed. Based on Figure 5, Hernia seemed to show an ROC curve with a higher AUC score of approximately 0.86, whereas Pneumonia had the lowest AUC score of around 0.62. Overall, Hernia had a higher AUC score across the different models than the other diseases.

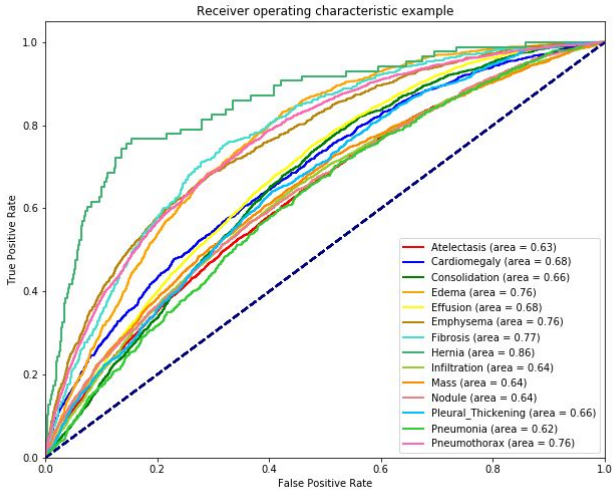

Figure 5: ROC curves for fine-tuned DenseNet-121 model

Additionally, as seen from Table 1, for both the ResNet-50 and DenseNet-121 models the average AUC scores of the models trained on 512 x 512 pixel images were consistently greater than the average AUC scores of the models trained on 256 x 256 pixel images while the AUC scores of fine-tuned models were only slightly higher than the KT models where only the output layer was trained. The average AUC score of all of the diseases was the highest for

the DenseNet-121 (512x512) fine tuned model with a value of about 0.70 and the lowest for the simple CNN model with a value of 0.63.

Model	Average AUC Score
CNN with 2 Hidden Layers	0.6329
ResNet-50 KT (256x256)	0.6663
ResNet-50 fine-tuned (256x256)	0.6670
ResNet-50 KT (512x512)	0.6868
ResNet-50 fine-tuned (512x512)	0.6901
DenseNet-121 KT (256x256)	0.6842
DenseNet-121 fine-tuned (256x256)	0.6843
DenseNet-121 KT (512x512)	0.6977
DenseNet-121 fine-tuned (512x512)	0.6978

Table 1: Average AUC scores of various models

Discussion

The overall framework we implemented for this project can be used for any kind of image classification task. Although the X-ray images are grayscale, our implementation reads them as colored images since that is a more general process and the pretrained ResNet and DenseNet models which expect to receive colored images. Our ETL framework assumes all images are colored and flattens them accordingly. Similarly our reloading process knows that the flattened data is composed of colored images of arbitrary size and converts them to tensors accordingly. As a result, our implementation only requires minimal changes when a different classification task is desired.

As our data is imbalanced, the training process will favor Type-II errors (false negatives) unless we utilize a balancing method during that stage. Especially for diseases such as Hernia where we only had 141 positive labels, predicting them all false would yield almost 100% accuracy. We penalized Type-II errors based on the negative to positive label ratio to establish balancing. While this decreased Type-II errors, it caused increasing Type-I errors (false positives) as they were penalized less resulting in mixed results for AUC score for different diseases.

For our machine learning models, after establishing the baseline with a simplified CNN design that was trained with limited data, we utilized knowledge transfer by using pre-trained ResNet and DenseNet architectures that comprise a more complex set of layers. The key aspect of training the pre-trained datasets was to do it in two stages. The first stage freezes the layers that were already trained so that we would train only the final fully connected layer that we replaced during the initial training. If the layers other than the final layer are not frozen, then the backpropagation would carry large errors throughout the network, destroying the knowledge that the pre-trained network carries. After finding the best model using validation AUC metric in the first stage, we unfroze all layers in the network so that we could fine tune it. This is possible after the first stage since the final layer was already trained and the errors carried via backpropagation will smoothen out the network weights instead of destroying the knowledge. Fine tuning stage also has another benefit: the new data may carry additional information for feature extraction as some of the important features in the new dataset may not exist in the original data. As the layers other than the final layer were trained with a different dataset and the feature extraction was optimized for that dataset. The fine tuning stage allows the network to learn the newly available important features and improves the quality of the network predictions by capturing this information. In our case, this only resulted in marginal gains. We gained less than 0.1% during fine tuning which shows that most X-ray image features were represented well in the original data.

As we increased the image size from 256x256 pixels to 512x512 pixels, our AUC improved by around 2%. While we downsample the images from the original size of 1024x1024, we are losing some crucial information such as sharp edges and other features, so an AUC improvement was expected. Similarly, we would expect to reach a higher

score if we used the full size images which would be an extension of this project. Since the whole images are used to train all diseases, irrelevant regions are also used to train the models. Bounding box information was available for a subset of images, and usage of this information can also be used as an extension to improve the results.

Another extension that we were planning to implement but were not able to, due to time constraint, was using the demographics information supplied to us as a separate file. Since the demographics data has additional information that may not be possible to extract from images, forming a parallel path to our CNN architecture and merging them with a fully connected final layer is expected to improve our results. A simpler integration of the demographics information would be to train a separate machine learning model and ensemble two models by training voting weights. The ensembling strategy can also be used to combine ResNet and DenseNet architectures which is expected to supply a better model by decreasing the overall error variance.

Conclusion

In this paper, we presented a method to implement our framework for classifying X-ray images. We analyzed the input data to find out potential classification challenges due to the frequency of 14 diseases and their co-occurrences. Our ETL pipeline and CNN models can be implemented independent of each other and also be used for other image classification applications with minimal change. Widely used ResNet-50 and DenseNet-121 models fared much better than our simple CNN as they could extract more complex image features. During our experiments, we observed that using higher resolution images improved the AUC score by 2% while there were only marginal gains from fine tuning the models after knowledge transfer. Our results were slightly worse than the state of the art research results which can be explained by our usage of downsampled images or the additional methods utilized in the other research such as usage of demographics data present in the NIH Chest X-ray dataset. These are possible extensions we are planning to explore further in future research.

References

[1] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., & Summers, R. M. (2017). 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)*. doi:10.1109/cvpr.2017.369

[2] Annarumma, M., Withey, S. J., Bakewell, R. J., Pesce, E., Goh, V., & Montana, G. (2019). Automated Triaging of Adult Chest Radiographs with Deep Artificial Neural Networks. *Radiology, 291*(1), 272-272. doi:10.1148/radiol.2019194005

[3] Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., . . . Ng, A. Y. (2019). CheXpert: A Large Chest Radiograph Dataset with Uncertainty Labels and Expert Comparison. *Proceedings of the AAAI Conference on Artificial Intelligence, 33*, 590-597. doi:10.1609/aaai.v33i01.3301590

[4] Li, Z., Wang, C., Han, M., Xue, Y., Wei, W., Li, L., & Fei-Fei, L. (2018). Thoracic Disease Identification and Localization with Limited Supervision. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition*. doi:10.1109/cvpr.2018.00865

[5] Rajpurkar, P., Irvin, J., Zhu, K., Yang, B., Mehta, H., Duan, T., Ding, D., Bagul, A., Langlotz, C., Shpanskaya, K., Lungren, M. P.,and Ng, A. Y. 2017. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning. *arXiv:1711.05225 [cs, stat]*. arXiv: 1711.05225.

[6] Guan, Q., & Huang, Y. (2020). Multi-label chest X-ray image classification via category-wise residual attention learning. *Pattern Recognition Letters, 130*, 259-266. doi:10.1016/j.patrec.2018.10.027

[7] Liu, H., Wang, L., Nan, Y., Jin, F., Wang, Q., & Pu, J. (2019). SDFN: Segmentation-based deep fusion network for thoracic disease classification in chest X-ray images. *Computerized Medical Imaging and Graphics, 75*, 66-73. doi:10.1016/j.compmedimag.2019.05.005

[8] Baltruschat, I. M., Nickisch, H., Grass, M., Knopp, T., & Saalbach, A. (2019). Comparison of Deep Learning Approaches for Multi-Label Chest X-Ray Classification. *Scientific Reports, 9*(1). doi:10.1038/s41598-019-42294-8

[9] Ge, Z., Mahapatra, D., Sedai, S., Garnavi, R., Chakravorty, R.: Chest xrays classification: A multi-label and fine-grained problem. In: arXiv preprint arXiv:1807.07247 (2018)

[10] Azure. (2020, December 04). Azure/azure-sdk-for-net. Retrieved December 07, 2020, from https://github.com/Azure/azure-sdk-for-net/blob/master/sdk/storage/Azure.Storage.Blobs/README.md

Contribution

All team members contributed equally

Team Member	Contribution
Theophane Godonou	25%
Vamsi S Jandhayala	25%
Orcun Kurugol	25%
Phong Le	25%

Table 2: Team contribution