Team20-Chest X-ray Pathology Detection with CNN

Xiao-Hong Deng[1], Gerry Meixiong[1], Chenxu Wen[1], Lan Yao[1]
[1]Georgia Institute of Technology, Atlanta, Georgia, USA

Abstract

This project uses a convolutional neural network to reproduce the model presented in [Rajpurkar et al., 2017]. We train and fine-tune the model on the CheXpert dataset provided by [Irvin et al., 2019]. The model is a multi-label image classifier that is able to predict and localize thoracic pathologies in X-ray images. We achieve performance comparable to the one presented in the original paper with a single model.

1 Project Links

Our project presentation slides are available here.

Our video presentation is available here or at the following link: https://www.youtube.com/watch?v=Tq4i1p9fv9s

The code is available on GitHub here.

2 Introduction

The chest X-ray is one of the most commonly accessible and effective radiological examinations for screening and diagnosis of many serious diseases in the United States. For example, more than 1 million adults are hospitalized with pneumonia and around 50,000 die from the disease in US every year according to CDC, 2017 and the chest X-ray is the best available method for diagnosing pneumonia.

However, even if chest X-ray is relatively cheap and can be performed with minimal procedural steps, detecting diseases such as pneumonia using chest X-ray is often challenging even for expert radiologists, and the availability of expert radiologists is very limited. Therefore, computer-aided diagnosis of diseases from chest X-rays with high-precision would not only have tremendous benefit in clinical settings, it would also be invaluable in delivery of health care to populations with inadequate access to diagnostic imaging specialists.

3 Related Work

In order to apply the rapidly developing techniques from deep learning and computer vision in the detection of diseases using chest X-ray, we will need lots of data. One large public accessible chest X-ray dataset, "ChestX-ray14", was presented by [Wang et al., 2017]. The dataset contains more than $100,000$ frontal-view X-ray images from over $30,000$ unique patients and it also comes with 14 disease image labels from associated radiological reports via natural language processing. The authors also provided a unified weakly supervised multi-label image classification and disease localization framework to detect and spatially locate those diseases. Another large dataset which is accessible by public, "CheXpert", which contains 224,316 chest (frontal and lateral views) radiographs of 65,240 patients labeled for the presence of 14 common chest radiographic observations, was presented by [Irvin et al., 2019]. The authors also investigated different approaches towards incorporating the uncertain labels into training convolutional neural networks.

[Li et al., 2019] presented an approach to perform disease identification and localization with a small number of location annotations. Their method involves first applying CNN to learn the entire image and then slicing the image into patch grid to learn the local information of the disease.

[Rajpurkar et al., 2017] developed an algorithm, CheXNet, a 121-layer Dense Convolutional Network (DenseNet) on the ChestX-ray14 dataset which can detect all 14 thoracic diseases at state-of-art level. Notably, CheXNet exceeds average radiologist performance on the F1 metric for pneumonia.

Many of the pathologies in lung X-ray are similar visually, and in general we have overwhelming normal samples v.s. disease samples in medical domain (imbalance data). In order to address those challenges, [Ge et al., 2018] proposed a novel error function based on softmax concept (Multi-label Softmax Loss, MSML) and designed a convolutional

deep network based on fine-grained classification methodology that incorporates MSML. They were able to improve the performance on the ChestX-ray14 compared to existing methods.

[Yao et al., 2018] introduced an approach by applying a novel architecture that learns at multiple resolutions while generating saliency maps with weak supervision. They also parameterized the Log-Sum-Exp pooling function with a learnable lower-bounded adaptation (LSE-LBA) to build in a sharpness prior and better handle localizing abnormalities of different sizes using only image-level labels. They were able to set the state of the art on 9 diseases on ChestX-ray14 while generating saliency maps with high resolution.

[Gündel et al., 2019] proposed a novel approach based on location-aware Dense Networks (DNetLoc), whereby they incorporate both high-resolution image data and spatial information for abnormality classification. They were able to obtain the best average AUC score on ChestX-ray14 dataset.

[Liu et al., 2019] presented a method called segmentation-based deep fusion network (SDFN), to address potential drawbacks from existing methods caused by noise from misalignment and existence of irrelevant objects in the entire image, and by information loss from the resizing operation. They were able to improve the results of Gündel et al [Gündel et al., 2019].

Class activation maps has been proved to be reliable in localization task, which helps in locating the areas that may be related to the diseases in X-Ray images. [Zhou et al., 2016] found that the CNN has remarkable localization ability without using any bounding box annotations. Weakly supervised object localization on the ILSVRC benchmark was conducted, and the result shows that the global average pooling CNNs can perform accurate object localization. Some other features of CAM are also discovered in their research. For example, the experiment of fine-grained classification was conducted and the result indicates that GoogLeNet-GAP with CAM can successfully localize and classify the objects. The ability of pattern discovery beyond objects is also demonstrated in the research.

4 Dataset Description and Statistics

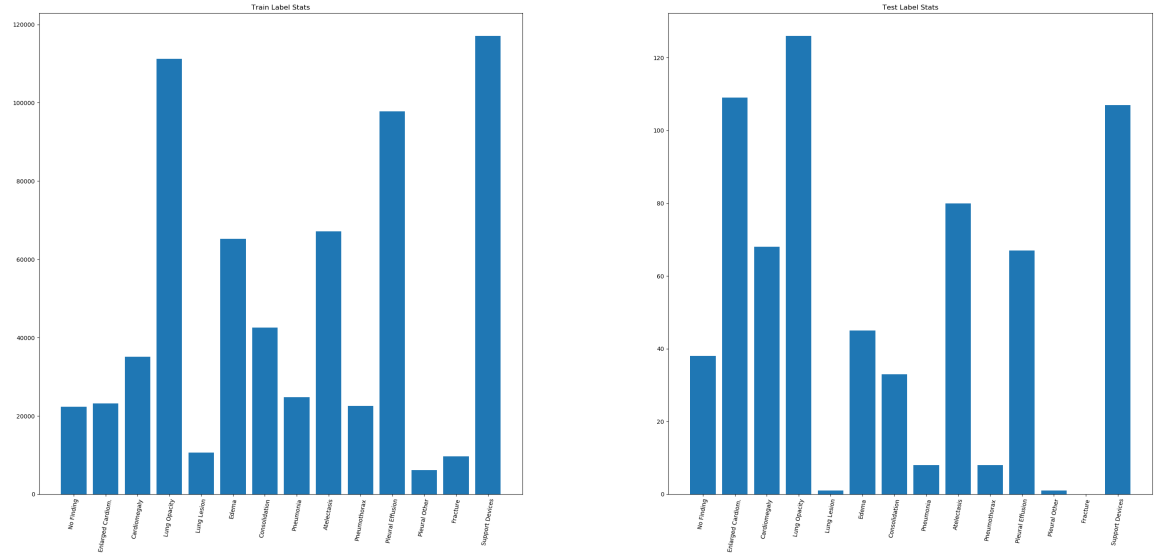
In this project, we will use the CheXpert dataset [Irvin et al., 2019]. CheXpert is a large public dataset for chest radiograph interpretation, consisting of 224,316 chest radiographs of 65,240 patients labeled for the presence of 14 observations as positive, negative, or uncertain. Each sample consists of a frontal-view image and a lateral-view image. It provides a version of the dataset with high image resolution and a downscaled version which is easier to work with.

The radiographic images and their associated radiology reports were collected from Stanford Hospital, between October 2002 and July 2017. 14 observations are chosen as labels based on the prevalence in the reports and clinical relevance. Most of the observations are pathologies with a few exceptions:

- No Finding: If there is no pathology labeled as positive or uncertain, give 1 to this observation.
- Pneumonia: Overall diagnosis.
- Support Devices: Observation that is not a pathology.

The training data was processed by an automatic labeler. The labeler extracted observation mentions from the associated radiology reports. After mentions were extracted, there were three stages in the labeling process: pre-negation, negation, and post-negation. During the three stages, an observation could be marked as uncertain(u), negative(0), or certain(1), respectively. If a mention was not marked after the three stages, it was marked positive. If an observation was not extracted from the report, it was marked blank or NaN. The validation set was labeled by 3 board-certified radiologists. The test set was labeled by 5 board-certified radiologists.

In Figure 1, we list the numbers of positive samples for each label. As we can see with ”Lung Lesion” and ”Pleural Other”, some labels have few samples in both the training and validation sets. This implies we need data augmentation to obtain balanced performance on all labels.



(a) Number of positive samples for each label in the training set **(b)** Number of positive samples for each label in the validation set

Figure 1: Positive label counts for all diseases

5 Preprocessing

The images can be classified according to the positioning of the patient into three classes: frontal-AP (anterposterior), frontal-PA (posterior-anterior), and lateral. This information may be useful for model tuning. We grouped the training set into these three classes.

Each image is loaded into the RGB format and then is randomly horizontal flipped. Then the image is downsized and center-cropped to the size of 224×224. After the image is transformed into tensors, we normalized the tensors with $mean = [0.485, 0.456, 0.406]$ and $std = [0.229, 0.224, 0.225]$.

There are various ways to encode uncertainty values in the labels in the dataset, which are marked by -1. For simplicity we use U-Ones encoding in [Irvin et al., 2019]. That is replacing -1 with 1. The reasoning is this kind of models are used to help human experts with the first-round screening. False negative is considered less preferable than false positive. Besides that we replace NaN with $0's$, turning missing values to negative values.

6 Approach

6.1 Problem Formulation

For each pathology, the detection task is a binary classification problem. We would like to optimize the *binary cross entropy loss*, with default set-ups provided by PyTorch. For a single example in the training set, the binary cross entropy loss is

$$L(X, y) = y \log p(y = 1|X) + (1 - y) \log p(y = 0|X).$$

6.2 Model Architecture

We would like to reproduce CheXNet by [Rajpurkar et al., 2017] which is based on DenseNet-121 [Huang et al., 2017]. DenseNets improve the flow of information and gradients through the network, making the optimization of very deep networks tractable. The structure of DenseNet-121 is shown in Figure 2.

CheXNet replaces the final fully connected layer with one that has a single output, after which a sigmoid nonlinearity

Layers	Output Size	DenseNet-121	DenseNet-169	DenseNet-201	DenseNet-264
Convolution	112 × 112	7 × 7 conv, stride 2			
Pooling	56 × 56	3 × 3 max pool, stride 2			
Dense Block (1)	56 × 56	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 6$
Transition Layer (1)	56 × 56	1 × 1 conv			
	28 × 28	2 × 2 average pool, stride 2			
Dense Block (2)	28 × 28	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 12$
Transition Layer (2)	28 × 28	1 × 1 conv			
	14 × 14	2 × 2 average pool, stride 2			
Dense Block (3)	14 × 14	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 24$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 64$
Transition Layer (3)	14 × 14	1 × 1 conv			
	7 × 7	2 × 2 average pool, stride 2			
Dense Block (4)	7 × 7	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 16$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 32$	$\begin{bmatrix} 1 \times 1 \text{ conv} \\ 3 \times 3 \text{ conv} \end{bmatrix} \times 48$
Classification Layer	1 × 1	7 × 7 global average pool			
		1000D fully-connected, softmax			

Figure 2: DenseNet-121

is applied. In the original paper [Rajpurkar et al., 2017], CheXNet is trained on the ChestX-ray 14 dataset. However, here we will train the network on the bigger dataset CheXpert.

6.3 Implementation and Training Environment

Our implementation is written in Python. Due to the large size of the dataset, we choose to use PyTorch and Keras to utilize GPU to accelerate the training process. We initialized the model with the pre-trained DenseNet121 provided in PyTorch. We used the Adam with $\beta_1 = 0.9$, $\beta_2 = 0.999$, and learning rate 0.0002 to train our model, which are identical to the values used in [Irvin et al., 2019].

The model was trained on a desktop with an i5-3470 CPU, 16 GB RAM and a GTX 1070 GPU with 8 GB on-board memory. The GPU supports single-precision floating-point number training only. The number of trainable parameters in DenseNet121 is close to 7 million. During training, back-propagation can take up to 3 times of the size of the memory required by forward-propagation. Given $\texttt{batch\_size} = 32$, the parameters alone can take $7000000 \times 4 \times 32 \div 1024 \div 1024 \times 3 = 2563$ MB memory. The remaining overhead is largely implementation-dependent. Through trial and error, we found that mini-batches of size 64 could not be fit into 8 GB on-board memory, so we opted for $\texttt{batch\_size} = 32$.

We use AUROC to evaluate the performance of the model.

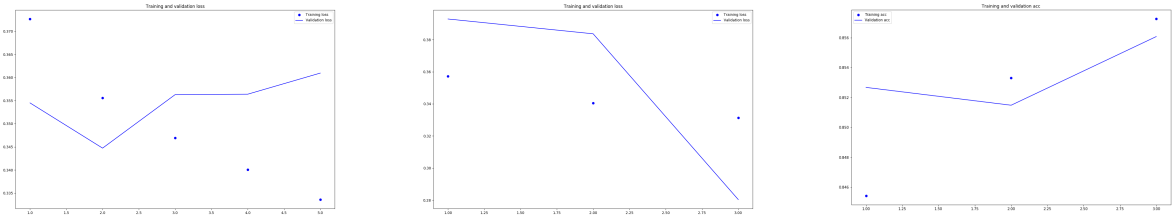
6.4 Training Procedure and Pitfalls

With the given model architecture, we tried three different approaches and used AUROC to evaluate the resulting models.

First we tried training the model from scratch, given that the model was loaded with pre-trained weights on ImageNet dataset. Then we save the model and generated the AUROC data for each of the pathologies.

In this approach, we implemented the procedure in PyTorch. Before feeding data to the model, we randomly flip the data as a way of doing data augmentation. We also resized the images to 224×224 so that the size matched the input size of the model. We also did standardization to the images. We trained the model for 5 epochs, each epoch took 6981 batches and each batch contained 32 images.

In the second approach, we tried *transfer-learning*. We first load the base model and adapted it to the customized form as we mentioned before. Then we froze the layers that were not customized by us, which had the weights trained on ImageNet dataset. The only layer that was trainable was the last classification layer we added to the model, which was



(a) First approach, training and validation loss **(b)** Second approach, training and validation loss **(c)** Third approach, training and validation accuracy

Figure 3: Selective training plots

initialized randomly.

The data preprocessing step was done slightly differently in this approach. We just resized the images to 224×224 and shift all pixel values to the range between 0 and 1. Other than that everything else including parameter values and training steps was the same as the first approach.

We implemented the procedure in Keras and trained it for 3 epochs. Because only the last layer was trainable, the bottleneck during the training was the image loading to GPU and GPU was idle from time to time.

In the third approach we continued training based on the model we obtained from the second approach. In theory, this can avoid the flaw in the first approach. That is when some of the layers have fine-tuned weights and others have random weights, the gradients generated and propagated by the random weights tend to be too large to overtune the fine-tuned weights, cause the model to take longer time to converge.

The data preprocessing and training steps were the same as the second approach. We trained for 3 epochs. This time the GPU became the bottleneck of the training.

In all three approaches we saved the best checkpoint based on some metric. In the first approach we used validation loss as the metric while in the second and third ones we used validation accuracy as the metric.

In Figure 3 we present some of the plots we collected during training the model using the three approaches.

While building the model we have difficulty to understand how different libraries constructed the base model DenseNet121. In Keras, the single layers are encapsulated in blocks, which are essentially custom layers. By removing the last classification layer we actually removed the last fully connected layer and the global average pooling layer. To keep the model the same as the one used in [Rajpurkar et al., 2017] we carefully added back the pooling layer.

But in PyTorch things are done differently. When we looked into the structure of DenseNet121 in PyTorch we did not find the last global pooling layer. It turned out to be the case that for layers without trainable parameters one could use static layers provided in a particular package in PyTorch. This kind of layers are not visible by printing the model or using third party visualization tools. We found out by looking into the PyTorch source code.

We used to train as many as 20 epochs to see how much time did it take the training to overfit the model. It turned out to be that after 5 epochs the gap between training and validation loss increased monotonically. Later we adjusted it to 5 or fewer epochs to save time. This is validated by the number of epochs used by authors of different papers we cite.

7 Experimental Results

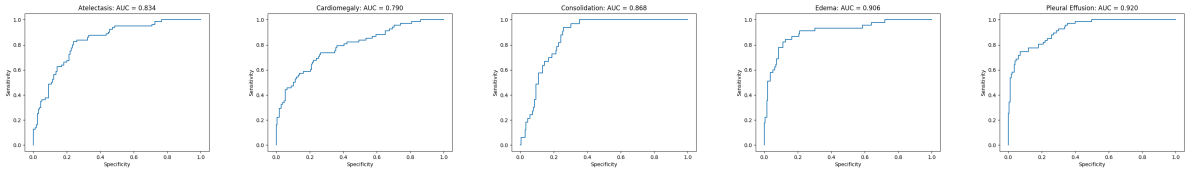
The AUROC scores of our model on the five selected pathology are in Table 1. Those are the diseases presented in [Irvin et al., 2019]

In Figure 4 we show the AUROC curves for the five listed diseases collected from the model trained directly from scratch.

As we can see, transfer learning based on the pre-trained model produced a model that is not competitive. In general

Pathology	Transfer Learning	Trained from Scratch	Trained from TF	CheXpert
Atelectasis	0.720	0.834	0.788	0.858
Cardiomegaly	0.673	0.790	0.831	0.832
Consolidation	0.658	0.868	0.857	0.899
Edema	0.779	0.906	0.911	0.941
Pleural Effusion	0.724	0.920	0.933	0.934

Table 1: AUROC Scores: All models are compared under U-Ones encoding



(a) Atelectasis **(b)** Cardiomegaly **(c)** Consolidation **(d)** Edema **(e)** Pleural Effusion

Figure 4: AUROC curves for selected diseases from trained from scratch model

transfer learning is suitable for situations where data for the interested task is hard to get. For those scenarios transfer learning can produce robust models with very few data. Also it takes little time to train. In our case, the available training images are over two hundred thousand which is not the best case to apply transfer learning.

The other two approaches produced models with similar performance in terms of AUROC. In training from scratch, we did more data augmentation steps. It did not bring noticeable advantage over the model trained on transfer learned one. It could be that we need more aggressive data augmentation which would take more time to train.

On average, the transfer learner based model slightly outperforms the one trained from scratch. It is difficult to tell if the third approach is truly effective in this scenario because their performance are close to each other.

We applied *class activation map* technique to the raw images and found that it performed well localizing the lesion. We used the same implementation employed by [Irvin et al., 2019], which was proposed by [Selvaraju et al., 2016].

8 Discussion

With the same label encoding U-Ones two of our trained models achieved performance that is comparable to the one presented in [Irvin et al., 2019].

In [Irvin et al., 2019] the model is the ensemble of 30 checkpoints from 3 epochs. The checkpoints are the best 10 checkpoints from each of the 3 epochs. Checkpoints are saved every 4800 batches. Batch size is 16.

From the performance of our models on labels ”Atelectasis” and ”Cardiomegaly” we can see that ensemble of our best two models might be beneficial. If we ensembled the checkpoints from the training we might get even better performance.

Figure 6 shows the AUROC values for the selected diseases by the ensemble models using different label encodings. Note that even our coarse single model outperforms all the ensemble models using encodings other than U-Ones in ”Atelectasis”. This validates the experimental results in [Irvin et al., 2019].

Another thing we may learn from Figure 6 is different encodings perform differently for different labels. Thus we can encode data with more than one encoding strategies. For example, for the label column that corresponds to ”Atelectasis” we may encode the data with U-Ones and for the label column that corresponds to ”Cardiomegaly” we may encode the data with U-MultiClass.

9 Conclusion

We implemented the CheXNet and trained it using the CheXpert dataset. The overall performance of our model is on par with the ones proposed in the original paper.

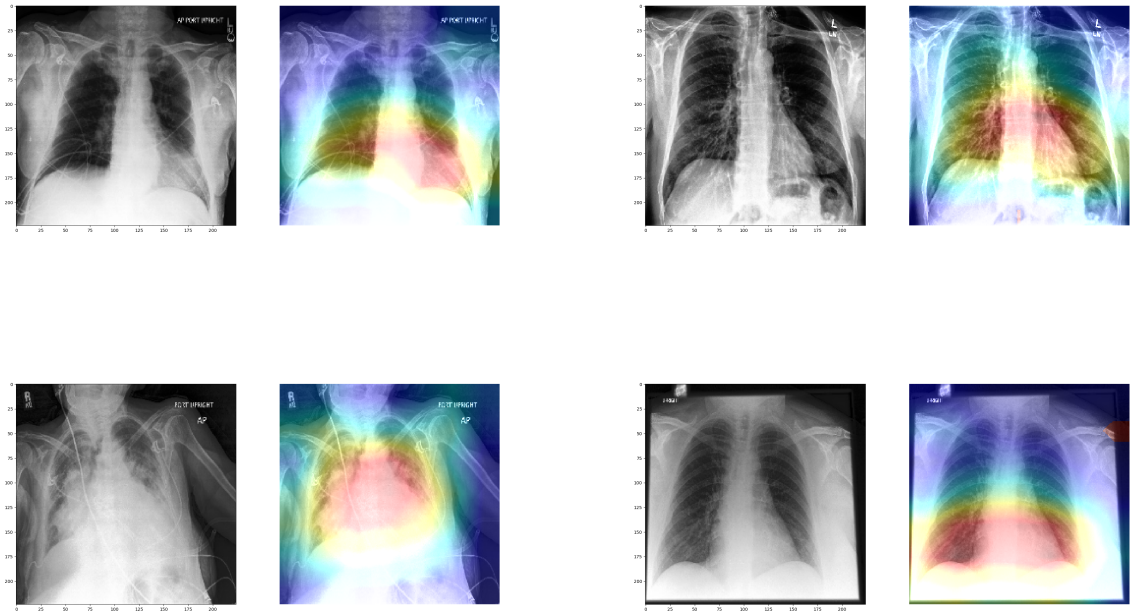


Figure 5: Our model localizes findings in images via Gradient-based localization.

	Atelectasis	Cardiomegaly	Consolidation	Edema	Pleural Effusion
U-Ignore	0.818 (0.759,0.877)	*0.828 (0.769,0.888)*	0.938 (0.905,0.970)	0.934 (0.893,0.975)	0.928 (0.894,0.962)
U-Zeros	*0.811 (0.751,0.872)*	0.840 (0.783,0.897)	0.932 (0.898,0.966)	0.929 (0.888,0.970)	0.931 (0.897,0.965)
U-Ones	**0.858 (0.806,0.910)**	0.832 (0.773,0.890)	0.899 (0.854,0.944)	0.941 (0.903,0.980)	0.934 (0.901,0.967)
U-SelfTrained	0.833 (0.776,0.890)	0.831 (0.770,0.891)	0.939 (0.908,0.971)	0.935 (0.896,0.974)	0.932 (0.899,0.966)
U-MultiClass	0.821 (0.763,0.879)	**0.854 (0.800,0.909)**	0.937 (0.905,0.969)	0.928 (0.887,0.968)	0.936 (0.904,0.967)

Figure 6: Results Obtained in [Irvin et al., 2019]

There are a few techniques that we could apply to the model to potentially improve the model performance but did not due to time constraints.

- Ensemble
- Aggressive data augmentation
- Mixed label encoding

Despite that we did not apply all the techniques that we were aware of, we were able to achieve results that were close to the state of the art research using a single model.

10 Team Contributions

Xiao-Hong Deng: Improved PyTorch model. Implemented Keras models. Trained and tuned all the models. Implemented some io and visualization code to help with the training and analysis. Contributed Dataset and Approach sections in the proposal and was responsible for organizing and putting sections from all team members together to form the final proposal. Wrote the final report based on the draft.

Gerry Meixiong: Integrated the CheXPert dataset for use with PyTorch, including data transformations and preprocessing. Added functionality to train and test a PyTorch model, including calculating and plotting AUROC. Responsible for the Project Presentation.

Chenxu Wen: Worked with Lan Yao to implement the main PyTorch model of the project. Implemented the gradient-based heatmap for the model . Wrote an initial version of the project proposal. Wrote the project draft based on results on the first iteration.

Lan Yao: Implemented the code that generates the label distribution graph. Implemented the PyTorch model and the training process used in our first approach. Hook up the heatmap generation code with the main program to generate the heatmap graphs used in the paper. Wrote a version of project proposal and mainly contributed the Related Work section to the final proposal.

References

[Ge et al., 2018] Ge, Z., Mahapatra, D., Sedai, S., Chakravorty, R., and Garnavi, R. (2018). Chest x-rays classification: a multi-label and fine-grained problem. *arXiv:1807.07247v3*.

[Gündel et al., 2019] Gündel, S., Grbic, S., Georgescu, B., Liu, S., Maier, A., and Comaniciu, D. (2019). Learning to recognize abnormalities in chest x-rays with location-aware dense networks. pages 757–765.

[Huang et al., 2017] Huang, G., Liu, Z., Van Der Maaten, L., and Weinberger, K. Q. (2017). Densely connected convolutional networks. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2261–2269.

[Irvin et al., 2019] Irvin, J., Rajpurkar, P., Ko, M., Yu, Y., Ciurea-Ilcus, S., Chute, C., Marklund, H., Haghgoo, B., Ball, R. L., Shpanskaya, K. S., Seekins, J., Mong, D. A., Halabi, S. S., Sandberg, J. K., Jones, R., Larson, D. B., Langlotz, C. P., Patel, B. N., Lungren, M. P., and Ng, A. Y. (2019). Chexpert: A large chest radiograph dataset with uncertainty labels and expert comparison. *Proceedings of the AAAI Conference on Artificial Intelligence*.

[Li et al., 2019] Li, Z., Wang, C., Han, M., Xue, Y., Wei, W., Li, L., and Li, F. (2019). Thoracic disease identification and localization with limited supervision. *Advances in Computer Vision and Pattern Recognition*.

[Liu et al., 2019] Liu, H., Wang, L., Nan, Y., Jin, F., Wang, Q., and Pu, J. (2019). Sdfn: Segmentation-based deep fusion network for thoracic disease classification in chest x-ray images. *Computerized Medical Imaging and Graphics*, pages 66–73.

[Rajpurkar et al., 2017] 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*.

[Selvaraju et al., 2016] Selvaraju, R, R., Das, A., Vedantam, R., Cogswell, M., D, P., and Bartra, D. (2016). Grad-cam: Why did you say that? visual explanations from deep networks via gradient-based localization. *CoRR, abs/1610.023917*.

[Wang et al., 2017] Wang, X., Peng, Y., Lu, L., Lu, Z., Bagheri, M., and Summers, R. M. (2017). Chestx-ray8: Hospital-scale chest x-ray database and benchmarks on weakly-supervised classification and localization of common thorax diseases. *IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI*, pages 3462–3471.

[Yao et al., 2018] Yao, L., Prosky, J., Poblenz, E., Covington, B., and Lyman, K. (2018). Weakly supervised medical diagnosis and localization from multiple resolutions. *arXiv:1803.07703v1*.

[Zhou et al., 2016] Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., and Torralba, A. (2016). Learning deep features for discriminative localization. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2921–2929.