arXiv:1901.11210v1 [cs.CV] 31 Jan 2019

Chester: A Web Delivered Locally Computed Chest X-Ray Disease Prediction System

Joseph Paul Cohen   JOSEPH.PAUL.COHEN@MILA.QUEBEC
Paul Bertin   BERTINPA@MILA.QUEBEC
Vincent Frappier   FRAPPIEV@MILA.QUEBEC
Montreal Institute for Learning Algorithms
University of Montreal

Abstract

Deep learning has shown promise to augment radiologists and improve the standard of care globally. Two main issues that complicate deploying these systems are patient privacy and scaling to the global population. To deploy a system at scale with minimal computational cost while preserving privacy we present a web delivered (but locally run) system for diagnosing chest X-Rays. Code is delivered via a URL to a web browser (including cell phones) but the patient data remains on the users machine and all processing occurs locally. The system is designed to be used as a reference where a user can process an image to confirm or aid in their diagnosis. The system contains three main components: out-of-distribution detection, disease prediction, and prediction explanation. The system open source and freely available here: https://mlmed.org/tools/xray/

Keywords: Chest X-Ray, Radiology, Deep Learning

1. Introduction

Deep learning has shown promise to augment radiologists and improve the standard of care globally. Two main issues that complicate deploying these systems are patient privacy and scaling to the global population. To deploy a system at scale with minimal computational cost while preserving privacy we present a web delivered (but locally run) system for diagnosing chest X-Rays.

This paper describes a web based (but locally run) system for diagnosing chest X-Rays. Code is delivered via a URL[1] where they are presented an interface shown in Figure 1 while the patient data remains on the users machine and all processing occurs locally. The system is designed to be used as a reference where a user can process an image to confirm or aid in their diagnosis. The design of the system allows it to be available via a web browser for effortless access while also processing all data locally to ensure confidentially of patient data. The goals of this tool are as follows:

1. The medical community can experiment with it to figure out how it can be useful. It can provide "research gradient" so the machine learning community can identify challenges to work on.

2. The existence of this tool is an example of what is possible when data is open. It only exists because the NIH released 100k images (Wang et al., 2017) for unrestricted use. This tools can serve as an example to push for more unrestricted public dataset creation projects.
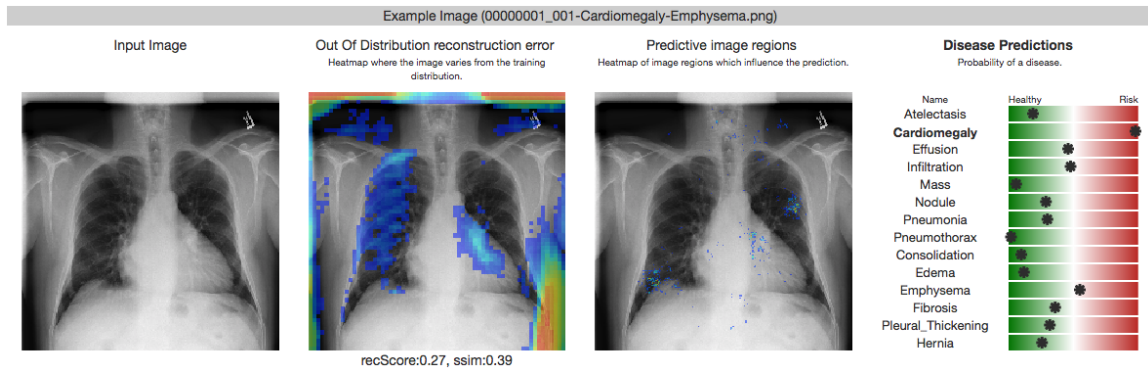
---

1. https://mlmed.org/tools/xray/

Figure 1: The web interface is designed to be simple tool which provides diagnostic information as well as a visual explanation of the prediction.

3. Across the entire world we can establish a lower bound of care. Every radiologist with a web browser should be no worse than this tool. We can transfer the knowledge of the NIH to everyone.

4. The design of the system is a model that can be copied to globally scale medical solutions without incurring significant server costs. No need to configure a computer with deep learning libraries. It can even run on a cell phone or from a thumbdrive without the internet.

5. Also, this system demonstrates that patient privacy can be preserved while using a web delivered system because the patient images never leave the client machine.

The system (Figure 1) is composed of three main parts: out of distribution (OOD) detection, disease prediction, and prediction explanation. Together these parts form a complete tool which can be used to complement the skills of a radiologist, doctor, or student. The idea that Deep Learning (LeCun et al., 2015) can be used to create tools to diagnose disease in chest X-Rays has been discussed recently (Litjens et al., 2017; Chartrand et al., 2017; Lakhani & Sundaram, 2017; Bar et al., 2015). However in this work we create such a tool in a way that it can be used by anyone for free and scale to millions of users globally. In this work we have identified many challenges which make systems like this difficult to create.

One difficult challenge is knowing when to not make a prediction. If you asked a radiologist to diagnose an image of something that is not their speciality they should not provide a diagnosis. For neural networks their *speciality* is the domain of the classifier which is defined by their training distribution. We cannot evaluate the accuracy of the model on these examples so we choose not to process them. This is illustrated in Figure 2. We discuss our solution in §3.

Explaining the prediction is critical to provide confidence in the prediction as well as allow users to come to their own conclusion with insight from the tool. Although many methods exist we have a limited computational budget. We discuss this in §4.

Computing locally in the browser was the most technical challenge. With the recent creation of tools such as (ONNX, 2017) and TensorFlow.js (Abadi et al., 2015) models
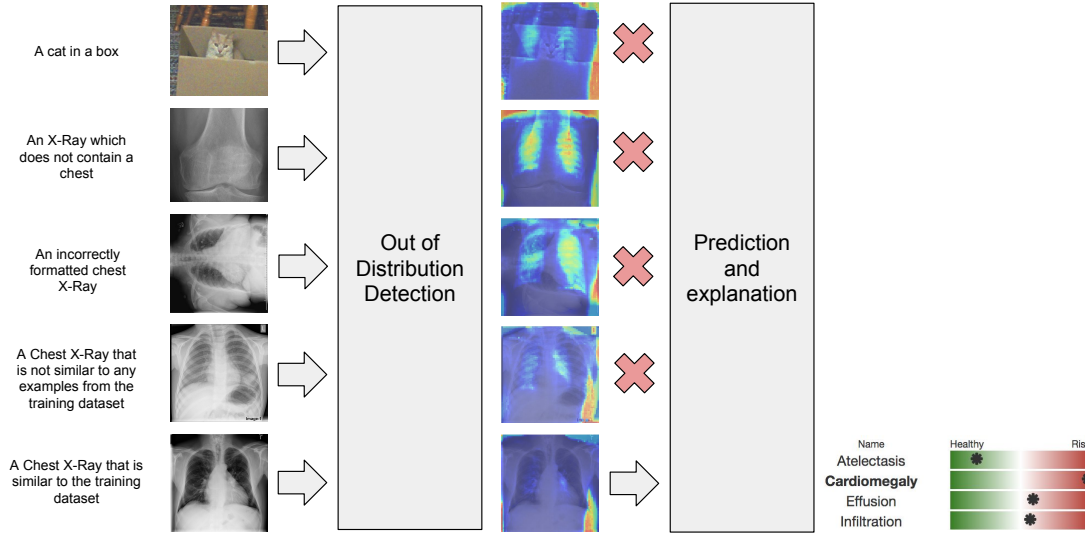
Figure 2: The use cases of an out-of-distribution detection method. Using a reconstruction loss we can visualize where the error occurs on the image. Only images which are similar to those in the training distribution are allowed to be processed.

trained in PyTorch (Paszke et al., 2017) can be converted to work in the browser and compute using WebGL. This is discussed in §5.

An open challenge continues to be generalization. While many images should qualify to be diagnosed from a human point of view, they are not represented by the training distribution even with data augmentation and will be rejected by this system. For example an image acquired from a different population or from a machine which is not represented in the training data could be incorrectly processed. Automating the transformation using domain adaptation approaches could have issues of hallucination (Cohen et al., 2018) where samples are projected into the training distribution without regard for maintaining diagnosis. Without a clear way to evaluate the transformation we don't use methods like this.

1.1. Data

Chest-Xray8 dataset To train our models we used the Chest-Xray8 dataset released by the NIH (Wang et al., 2017). This dataset contains 108,948 frontal-view X-ray images of 32,717 unique patients with 14 disease image labels: Atelectasis, Cardiomegaly, Effusion, Infiltration, Mass, Nodule, Pneumonia, Pneumothorax, Consolidation, Edema, Emphysema, Fibrosis, Pleural Thickening, and Hernia.

There have been claims that this dataset contains many errors in labelling (Oakden-Rayner, 2017). However this is the only dataset of its size and usage rights.

Pneumonia dataset: For evaluation of out-of-distribution examples we would like the model to correctly predict a dataset from Guangzhou Women and Childrens Medical Center is used (Kermany et al., 2018). This dataset contains 5,232 chest X-ray images from children.

3,883 labelled as depicting pneumonia (2,538 bacterial and 1,345 viral) and 1,349 normal cases. This dataset was only used after all models had finished being trained to provide an unbiased evaluation.

MURA dataset: For evaluation of out-of-distribution examples we would like the model to reject the MURA (musculoskeletal radiographs) dataset is used (Rajpurkar et al., 2018). It contains 40,561 bone X-Ray images from 14,863 studies, where each study is manually labeled by radiologists as either normal or abnormal. The X-Ray images contain a finger, wrist, elbow, forearm, hand, humerus, or shoulder.

We also experiment using the CIFAR-100 (Krizhevsky & Hinton, 2009) dataset of real world images as well as the MNIST (LeCun & Cortes, 1998) dataset of hand drawn digits.

2. Disease Prediction

We used an implementation of the CheXnet DenseNet-121 model (Rajpurkar et al., 2017) which is based on the DenseNet-121 architecture (Huang et al., 2017). It is trained with the train-validation-test split as in the initial paper $(70\%, 10\%, 20\%)$, using Adam with standard parameters $(\beta_1 = 0.9$ and $\beta_2 = 0.999)$, learning rate of 0.001, and learning rate decay of 0.1 when validation accuracy plateaus. We used the weights from (Weng et al., 2017), which achieve comparable performance compared to the original paper shown in Figure 1.

In order to output the final estimation of disease probability, we apply a piecewise linear transformation Eq. 1 to the predictions of the classifier. The transformation is chosen so that the best operating point corresponds to a 50% disease probability while ensuring that predictive probabilities lie in the set $[0, 1]$. The ROC curves and chosen operating points are shown in Figure 3. For each disease, we computed the optimal operating point by maximizing the difference (*True positive rate - False positive rate*). It corresponds to the threshold for which the ROC curve is furthest away from the diagonal.

$$f_{opt}(x) = \begin{cases} \frac{x}{2opt} & x \leq opt \\ 1 - \frac{1-x}{2(1-opt)} & otherwise \end{cases} \tag{1}$$
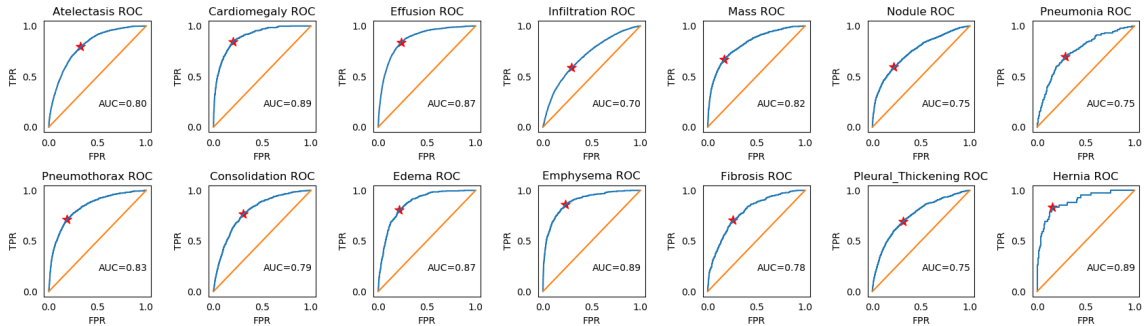

Figure 3: ROC of 14 diseases on the NIH ChestXRay holdout set. The red star corresponds to the operating point. We decide on an operating point and scale the output of the model to reflect this performance.

Table 1: Here we compare the performance (AUC) of models on the ChestX-ray14 dataset. We report baseline results from the CheXNet (Rajpurkar et al., 2017) and compute performance of our own model. Our standard deviation is computed using 10 random splits of the test set. Each split is half the size of the test set

Disease	ChestX-ray14 Wang et al. (2017) DenseNet-50	CheXNet Rajpurkar et al. (2017) DenseNet-121	CheXNet Weng et al. (2017) DenseNet-121
Atelectasis	0.716	0.809	0.810 ± 0.004
Cardiomegaly	0.807	0.924	0.900 ± 0.005
Effusion	0.784	0.863	0.874 ± 0.003
Infiltration	0.609	0.734	0.701 ± 0.004
Mass	0.706	0.867	0.824 ± 0.009
Nodule	0.671	0.780	0.745 ± 0.006
Pneumonia	0.633	0.768	0.760 ± 0.017
Pneumothorax	0.806	0.888	0.834 ± 0.006
Consolidation	0.708	0.790	0.794 ± 0.007
Edema	0.835	0.887	0.866 ± 0.005
Emphysema	0.815	0.937	0.894 ± 0.011
Fibrosis	0.769	0.804	0.789 ± 0.011
Pleural Thickening	0.708	0.806	0.751 ± 0.005
Hernia	0.767	0.916	0.889 ± 0.035

3. Predicting out of distribution

Work by (Zenati et al., 2018) found that using a GAN (Goodfellow et al., 2014) with an encoder such as ALI performs as well or better than VAEs (Kingma & Welling, 2014) and energy based models for outlier detection. We find that instead of estimating the density with ALI, a reconstruction loss works best to identify out of distribution samples. Here we train the encoder E and decoder D of an ALI model where the encoder output conforms to a Gaussian $E(\text{img}) \sim N(\mu, \sigma)$. Expecting the latent space to fit a Gaussian we explored using the distance of latent point z to the center of the Gaussian. This approach assumes that the model can be mapped to a single mode. Another approach is to look at the reconstruction distance after being projected into the latent space and reconstructed as follows:

$$\text{dist}(D(E(\text{img})), \text{img}) \tag{2}$$

There are a few choices for the dist function. An $L1$ or $L2$ norm was shown to be feeble when presented with real world images. Using SSIM (Wang et al., 2004) we can obtain a distance which accounts for maintaining spatial structure to prevent an offset of the reconstructed image to result in a large error.

3.1. ALI Training

An overview of the ALI model is shown in Figure 4. Images were resized to 64×64 pixels. 106,381 images were used for training, 1000 images were used for validation for
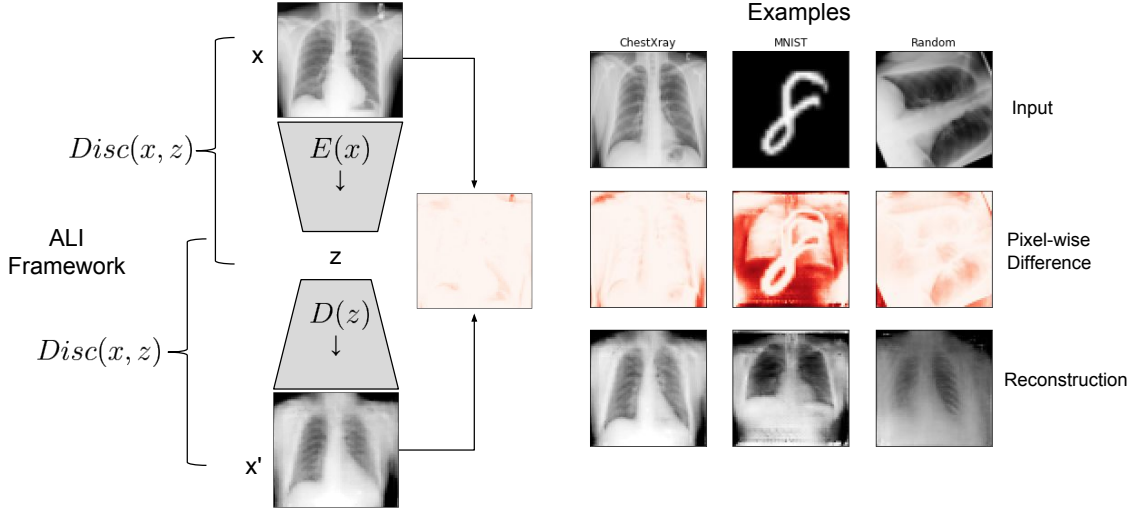
Figure 4: Example reconstructions using the model. The pixel-wise error is shown in red.

the ALI model. When training the ALI model most hyperparameters resulted in the same performance. We tried latent space sizes of 64, 128, 256, and 512 and learning rates of 0.000001 and 0.00001. To stabilize training we used two tricks, first we halt training of the generator when its performance becomes too good. Also, label smoothing is used where true positive labels are modified to between 0.7 and 1.1 and true negative labels between -0.1 and 0.3 (Salimans et al., 2016). We used an Adam optimizer with $\beta_1 = 0.5$ and $\beta_2 = 1e-3$. Each ALI model during the hyperparameter search was trained for 2,121,000 iterations or until the server crashed using a batch size 100.

3.2. Outlier Metrics

We experimented with multiple metrics to rank outlier images. First we try to use the latent variable of the ALI model because this variable should be a Gaussian centered at 0. We should be able to use the distance from the mean as an estimate of how likely the sample is in the training distribution. We also explore reconstruction error of the input images using L1, L2, and SSIM distances between the original and reconstructed image in Figure 5. We observed the latent variable distance failed to identify most of the outliers, while the L1, L2, and SSIM distance gave similar performance on the CIFAR, MNIST and MURA datasets, shown in Figure 7. Distance from the mean of the Gaussian distribution in the latent space also fail at separating CIFAR examples from ChestXray images. We observed that the CIFAR dataset is the most challenging data. This is likely due to its diversity, where it has more examples so by chance some examples can fool our model versus MURA or MNIST that are more homogeneous datasets.

Finally, we observed that images from the Pneumonia dataset tended to be classified as outliers and we wondered if they were truly out of the training distribution or if it was caused by a failure of our outlier detection methods. We scored each of these images using the DenseNet-121 model and observed that the predictive power increased upon increasing the stringency of the outlier classification cutoff in Figure 6.
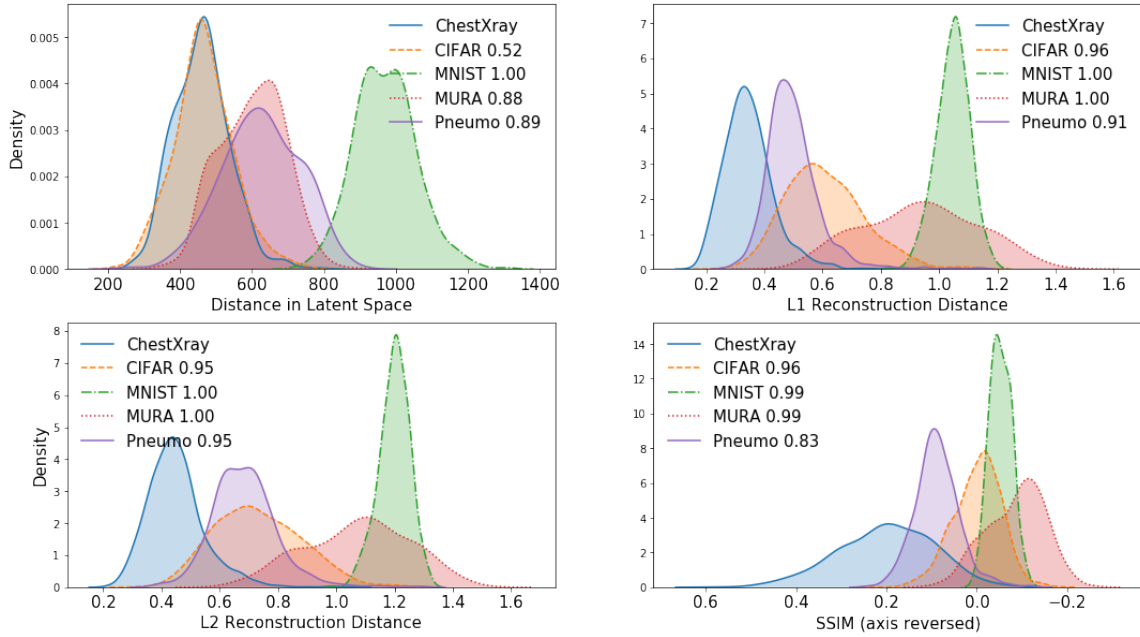
Figure 5: Distribution of OOD distances using different metrics. Euclidean distance in the latent space and three types of pixel-wise reconstruction losses: L1, L2, and SSIM.
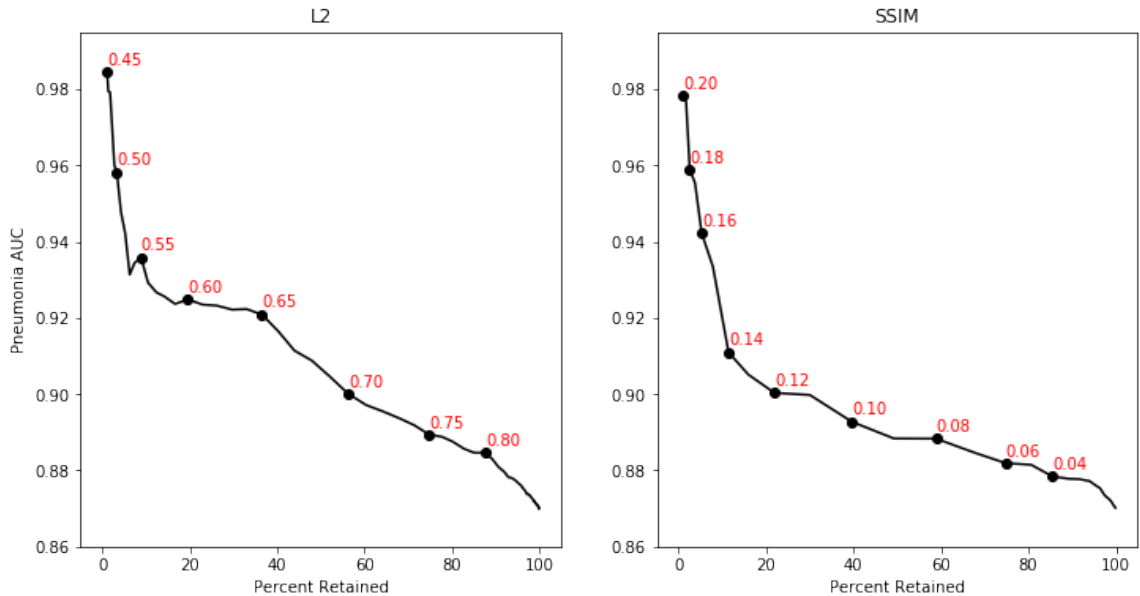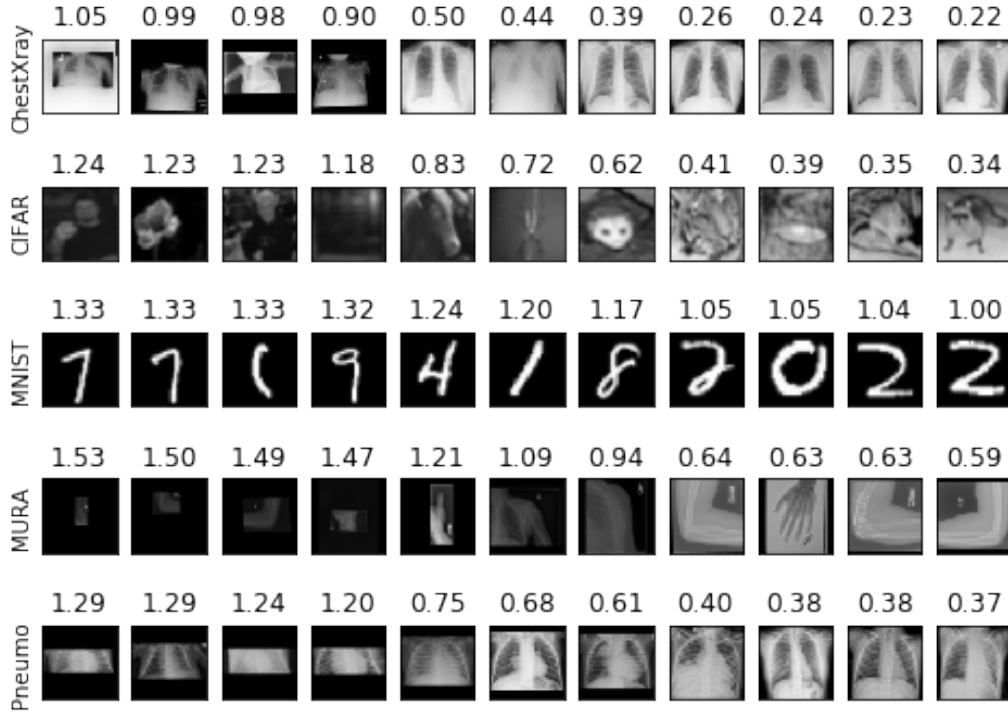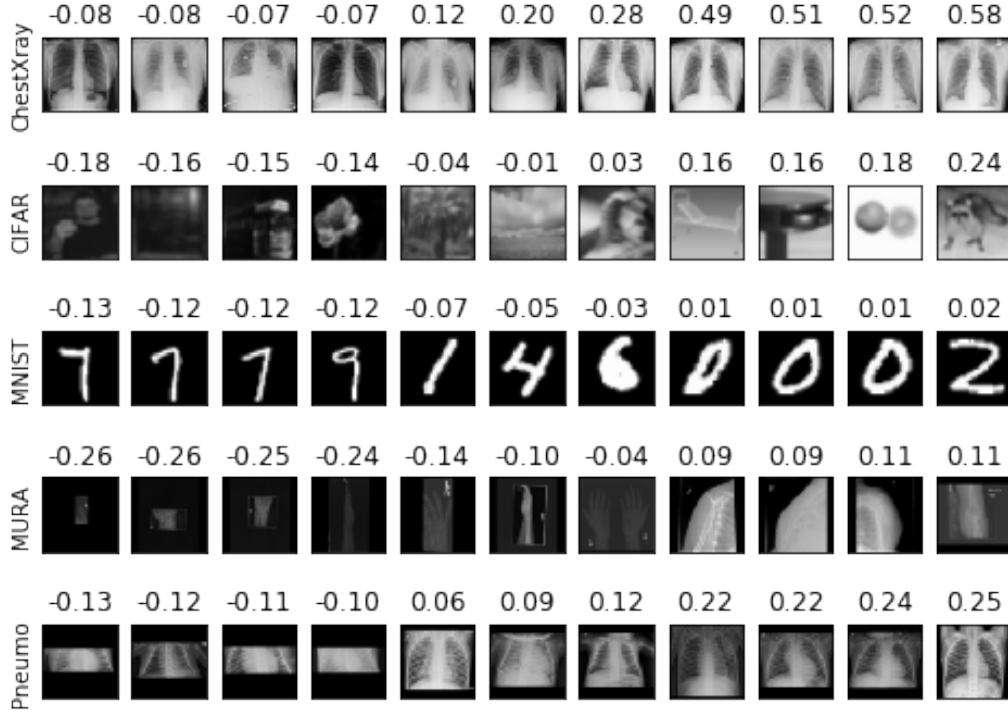

Figure 6: The DenseNet-121 performance on Pneumonia dataset increase with stringency of outlier cutoff value (shown in red) while the percent of retained images decrease

(a) Sample images using the L2 Reconstruction Loss

(b) Sample images using SSIM

Figure 7: Reconstruction error shown over different datasets. The input image is shown with the reconstruction overlaid in red. The model is not able to reconstruct X-Rays other than chest X-Rays so the reconstruction error is high.

4. Prediction explanation

Explaining the prediction of a neural network has had progress in recent years. Currently Grad-Cam (Selvaraju et al., 2016) is regarded as the go-to method when a computational budget is not an issue.

For simplicity we use the basic gradient saliency map approach discussed by (Simonyan et al., 2014; Lo & Cohen, 2015). Here for an input image I and the pre-softmax output of a neural network y we can compute the pixel-wise impact on a specific output y_i or over all outputs. The computation time is non-trivial and computing for all classes is the same as computing for a single class. The following saliency map is used for a general explanation of the prediction.

$$\text{saliency map} = \max(0, \sum_i \frac{\partial y_i}{\partial I}) \tag{3}$$

Figure 8 shows example saliency maps. Generally the gradient is high in clusters of pixels which cover regions which are predictive of the disease. One issue when interpreting gradients directly like this is that the gradient is high not only at the location of the feature (like a nodule) but also at locations which condition another region to have impact (like the area next to a nodule).

We compared this technique against another localization mapping called Class Activation Mapping (Zhou et al., 2015). Let $f_k(x, y)$ be the activation map of the kth filter of the last convolutional layer at location (x, y). A global pooling operation is performed on this activation map before being fed to the last FC layer. Thus, let ω_{ck} be the weight of the last FC layer that links filter k to class c. The Class Activation Map for class c at location (x, y) is computed as follows :

$$M_c(x, y) = \sum_k \omega_{ck} f_k(x, y) \tag{4}$$

This captures the part of the activation map after the last convolutional layer that has a strong impact on the prediction of class c. Figure 8 shows an example of localization of the Cardiomegaly disease using the two different mappings. The class activation mapping does not allow a very precise localization, as the resolution of the deep activation maps has been greatly reduced by successive pooling layers. Note that here the class activation map is upsampled to the size of the original image for visualization purpose.

5. Web Computation

The engineering of the system is designed in a modular way to be used by many projects in the future. To achieve this we create a pipeline using ONNX to transform models trained in PyTorch or other frameworks. From ONNX we can transform the models into frameworks which have browser support such as TensorFlow or MXNet (Chen et al., 2015). For this project TensorFlow.js was the most compatible across browsers and supported the necessary operations to translate the DenseNet and ALI models from PyTorch. The pipeline is as follows:
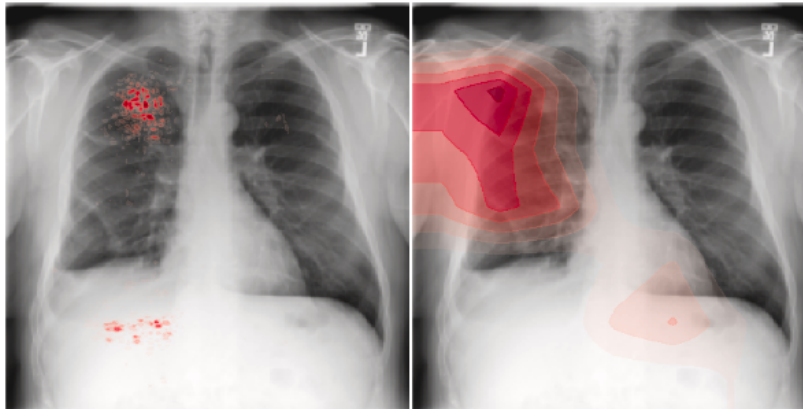
$$\text{PyTorch} \rightarrow \text{ONNX} \rightarrow \text{TensorFlow} \rightarrow \text{TensorFlow.js}$$
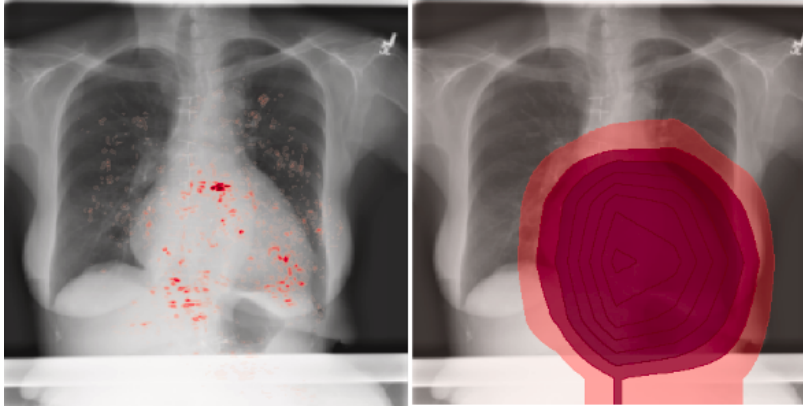
(*a*) Cardiomegaly
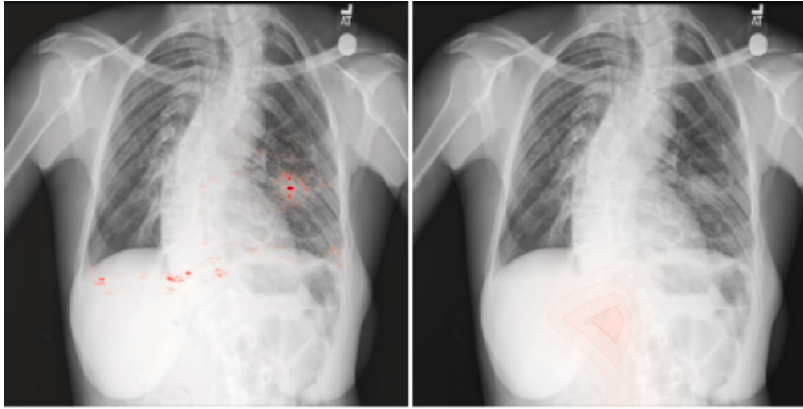
(*b*) Nodule

(*c*) Mass

Figure 8: Example of localization using the two different mappings. *Left* : Saliency Map. *Right* : Class Activation Map. The visualization explains all 14 diseases at the same time, transparent pixels have negligible impact on the prediction. The more red a region is the more of an impact that region has
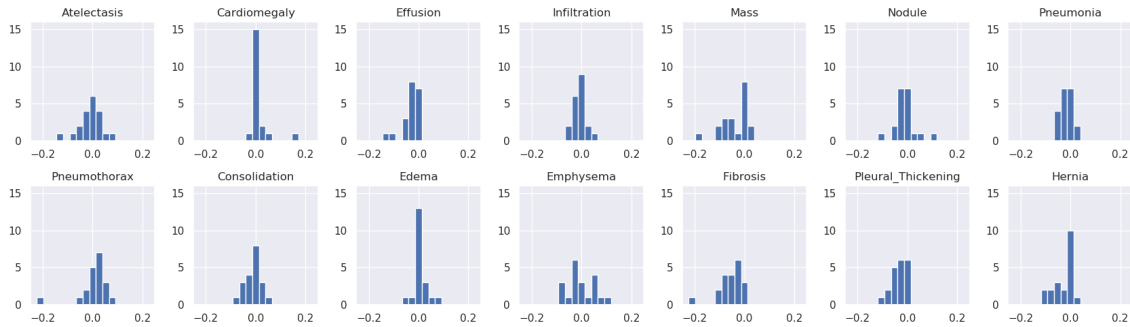
Figure 9: Histogram of differences between predictions using the JavaScript/WebBrowser image preprocessing pipeline and using the PyTorch pipeline, for each disease. This validation is performed over 20 randomly chosen images.

The methodology when deploying to the browser is to package the model's graph and weights into files which are then loaded by a script running in the browser which will reconstruct the graph and load the weights. The script must process images into the format expected by the model, execute the computation graph, and then present the results. The source code is available here: https://github.com/mlmed/dl-web-xray

We verified the computation graph conversion is correct by comparing predictions of the pytorch model to predictions of the tensorflow.js model on three images. Differences between predictions were within a $1e-5$ tolerance margin. Nonetheless, preprocessing methods differ, leading to different downscaled images. We validated on 20 images to assess prediction differences induced by the different preprocessing methods as shown in Figure 9. In extrem cases, predictions differed by up to 20%, but the average difference is $\pm 3\%$, leading to consistent predictions between the two pipelines. This reveals that our model is somehow sensitive to aleatoric noise, a common issue in deep learning.

5.1. Other design considerations

To ensure accurate image representation we used canvas elements instead of creating image objects. This offers exact control over the pixels of an image. When performing image preprocessing we scale and crop instead of stretching images. This is to ensure that structure remains correct.

5.2. Runtime

There are three main processing points: initial loading of the models (12 ± 2 seconds), computing the ALI and DenseNet computation graphs (1.3 ± 0.5 seconds), and computing gradients to explain predictions (17 ± 4 seconds).

There were two ways we can achieve gradient computations in the browser. We used the `tf.grad` method of TensorFlow.js which has the advantage that it can work on any model loaded into the site while having the disadvantage of slower running time. An alternative is to build a gradient computation graph and export that graph as regular model. This has the advantage that the graph could be built offline and optimized instead of being built

in realtime in the browser. However there seem to be limitations in the ability to export the computation graphs of the gradients due to missing backward operations and gradient computation graphs that are not compatible with ONNX.

6. Conclusion

In this work we present a complete tool to aid radiologists in diagnosing chest X-Rays. We address several critical needs for these systems such as: OOD detection, disease prediction, and prediction explanation. This tool can be used as an assistant and as a teaching tool. The system is designed to process everything locally which ensures patient privacy as well as enables it to scale from 1 to 1 Million users with negligible overhead. We hope this prompts radiologists to give us feedback which would help us improve this tool and adapt it to their needs.

Acknowledgements

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, Manjunath Kudlur, Josh Levenberg, Rajat Monga, Sherry Moore, Derek G Murray, Benoit Steiner, Paul Tucker, Vijay Vasudevan, Pete Warden, Martin Wicke, Yuan Yu, Xiaoqiang Zheng, and Google Brain. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems, 2015.

Yaniv Bar, Idit Diamant, Lior Wolf, and Hayit Greenspan. Deep learning with non-medical training used for chest pathology identification. Technical report, 2015.

Gabriel Chartrand, Phillip M. Cheng, Eugene Vorontsov, Michal Drozdzal, Simon Turcotte, Christopher J. Pal, Samuel Kadoury, and An Tang. Deep Learning: A Primer for Radiologists. *RadioGraphics*, 2017. doi: 10.1148/rg.2017170077.

Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. MXNet: A Flexible and Efficient Machine Learning Library for Heterogeneous Distributed Systems. *Neural Information Processing Systems, Workshop on Machine Learning Systems*, 2015.

Joseph Paul Cohen, Margaux Luck, and Sina Honari. Distribution Matching Losses Can Hallucinate Features in Medical Image Translation. *Medical Image Computing & Computer Assisted Intervention (MICCAI)*, 2018.

Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative Adversarial Networks. In *Neural Information Processing Systems*, 2014.

Gao Huang, Zhuang Liu, Laurens van der Maaten, and Kilian Q. Weinberger. Densely Connected Convolutional Networks. In *Computer Vision and Pattern Recognition*, 2017.

Daniel S Kermany, Michael Goldbaum, Wenjia Cai, Carolina C S Valentim, Huiying Liang, Sally L Baxter, Alex McKeown, Ge Yang, Xiaokang Wu, Fangbing Yan, Justin Dong, Made K Prasadha, Jacqueline Pei, Magdalene Y L Ting, Jie Zhu, Christina Li, Sierra Hewett, Jason Dong, Ian Ziyar, Alexander Shi, Runze Zhang, Lianghong Zheng, Rui Hou, William Shi, Xin Fu, Yaou Duan, Viet A N Huu, Cindy Wen, Edward D Zhang, Charlotte L Zhang, Oulan Li, Xiaobo Wang, Michael A Singer, Xiaodong Sun, Jie Xu, Ali Tafreshi, M Anthony Lewis, Huimin Xia, and Kang Zhang. Identifying Medical Diagnoses and Treatable Diseases by Image-Based Deep Learning. *Cell*, 2018. doi: 10.1016/j.cell. 2018.02.010.

Diederik P Kingma and Max Welling. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations*, 2014.

Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical report, University of Toronto, 2009.

Paras Lakhani and Baskaran Sundaram. Deep Learning at Chest Radiography: Automated Classification of Pulmonary Tuberculosis by Using Convolutional Neural Networks. *Radiology*, 2017. doi: 10.1148/radiol.2017162326.

Yann LeCun and Corinna Cortes. MNIST handwritten digit database, 1998.

Yann LeCun, Yoshua Bengio, and Geoffrey E. Hinton. Deep learning. *Nature*, 2015. doi: 10.1038/nature14539.

Geert Litjens, Thijs Kooi, Babak Ehteshami Bejnordi, Arnaud Arindra Adiyoso Setio, Francesco Ciompi, Mohsen Ghafoorian, Jeroen A. W. M. van der Laak, Bram van Ginneken, and Clara I. Sánchez. A Survey on Deep Learning in Medical Image Analysis, 2017. doi: 10.1016/j.media.2017.07.005.

Henry Z. Lo and Joseph Paul Cohen. Prediction gradients for feature extraction and analysis from convolutional neural networks. In *International Conference on Automatic Face and Gesture Recognition*, 2015. doi: 10.1109/FG.2015.7163154.

Luke Oakden-Rayner. Exploring the ChestXray14 dataset: problems, 2017.

ONNX. Open Neural Network Exchange, 2017.

Adam Paszke, Gregory Chanan, Zeming Lin, Sam Gross, Edward Yang, Luca Antiga, and Zachary Devito. Automatic differentiation in PyTorch. In *Neural Information Processing Systems Autodiff Workshop*, 2017.

Pranav Rajpurkar, Jeremy Irvin, Kaylie Zhu, Brandon Yang, Hershel Mehta, Tony Duan, Daisy Ding, Aarti Bagul, Curtis Langlotz, Katie Shpanskaya, Matthew P. Lungren, and Andrew Y. Ng. CheXNet: Radiologist-Level Pneumonia Detection on Chest X-Rays with Deep Learning, 2017. doi: 1711.05225.

Pranav Rajpurkar, Jeremy Irvin, Aarti Bagul, Daisy Ding, Tony Duan, Hershel Mehta, Brandon Yang, Kaylie Zhu, Dillon Laird, Robyn L Ball, Curtis Langlotz, Katie Shpanskaya, Matthew P Lungren, and Andrew Y Ng. MURA: Large Dataset for Abnormality Detection in Musculoskeletal Radiographs. Technical report, 2018.

Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved Techniques for Training GANs. In *Neural Information Processing Systems*, 2016.

Ramprasaath R. Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-CAM: Visual Explanations from Deep Networks via Gradient-based Localization, 2016.

Karen Simonyan, Andrea Vedaldi, and Andrew Zisserman. Deep Inside Convolutional Networks: Visualising Image Classification Models and Saliency Maps. Technical report, 2014.

Xiaosong Wang, Yifan Peng, Le Lu, Zhiyong Lu, Mohammadhadi Bagheri, and Ronald M. Summers. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases, 2017. doi: 10.1109/CVPR.2017.369.

Z. Wang, A. C. Bovik, H. R. Sheikh, and E. P. Simoncelli. Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 2004.

Xinyu Weng, Nan Zhuang, Jingjing Tian, Yingcheng Liu, and Yadong Mu. Python3 Pytorch reimplementation of CheXNet. *GitHub*, 2017. doi: https://github.com/arnoweng/CheXNet.

Houssam Zenati, Chuan Sheng Foo, Bruno Lecouat, Gaurav Manek, and Vijay Ramaseshan Chandrasekhar. Efficient GAN-Based Anomaly Detection, 2018.

Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Learning deep features for discriminative localization. *CoRR*, 2015.