Deep MedIA

MITCH GAVARS, Bucknell University, USA

Medical image diagnosis is, and always will be, important for treatment of patients. Processing of such images ensures that doctors have the possibility to double-check their diagnosis and to monitor their patients as treatment proceeds. In this paper, we explore how to use Convolutional Neural Networks and Gradient Class Activation Maps to determine disease classes and location of diseases on a chest x-ray, respectively.

CCS Concepts: • **Data Science → Medical Image Processing**.

Additional Key Words and Phrases: datasets, neural networks, object detection

1 INTRODUCTION

The goal of our project is to use machine learning to create a model that can identify diseases in chest x-ray images by drawing a bounding box around them. This model's primary application is to assist doctors with making medical diagnoses, allowing for fast and accurate localization of diseased areas. However, what we learn can also be applied in other ways, including providing athletic trainers more information on their athletes when doctors are unavailable and expanding the breadth of research into these kinds of problems for future developers to build from.

Object localization is a subset of machine learning that has a variety of possible solutions. In general, models are created using systems that can be updated in accordance with feedback from how well they achieve their goals.For neural networks in particular, the concept of backpropagation allows a model to update its weights over time to minimize the difference between the outputs that it should have, and the output it is currently generating. In our particular case, we are working to evaluate how well the model is able to place a bounding box around the correct part of a given image.

Our goal is to use the National Institute of Health Chest X-Ray Dataset to train a model that can accurately draw bounding boxes around diseased areas, as well as determine if there is a disease in the first place. To accomplish this, we intend to preprocess the image files and metadata into a usable format so that our model can learn to properly identify disease areas, and then work over time to refine its values and improve its accuracy. We will then develop and train a model to create the bounding boxes, training it on the limited data set that we are provided. Once our model identifies a bounding box, we will display this region on the input images, creating a useful representation that our users can process in a meaningful way. Figure 1 represents a general view of what our goal is for this project.

We considered the use of convolutional networks and related architectures, generative adversarial networks, and spatial pyramids to achieve accurate object localization. Convolutional networks are effective in extracting features of different sizes across a set of images, which could be quite useful in finding bounding boxes. Generative adversarial networks pit a generator and discriminator network against each other, teaching the generator to create more believable synthetic images while the discriminator learns to more effectively tell what is a fake image; this can be applied to generate "realistic" bounding boxes. Spatial pyramids are not neural networks, but work in their own right by using histograms of features to gain a spatial understanding of an image's components in a global scale. After evaluating these possibilities on our listed criteria and with the assistance of our client and technical advisor, we determined that a convolutional neural network was the correct approach.

Author's address: Mitch Gavars, mag051@bucknell.edu, Bucknell University, 704 Moore Avenue, Lewisburg, Pennsylvania, USA, 17837.

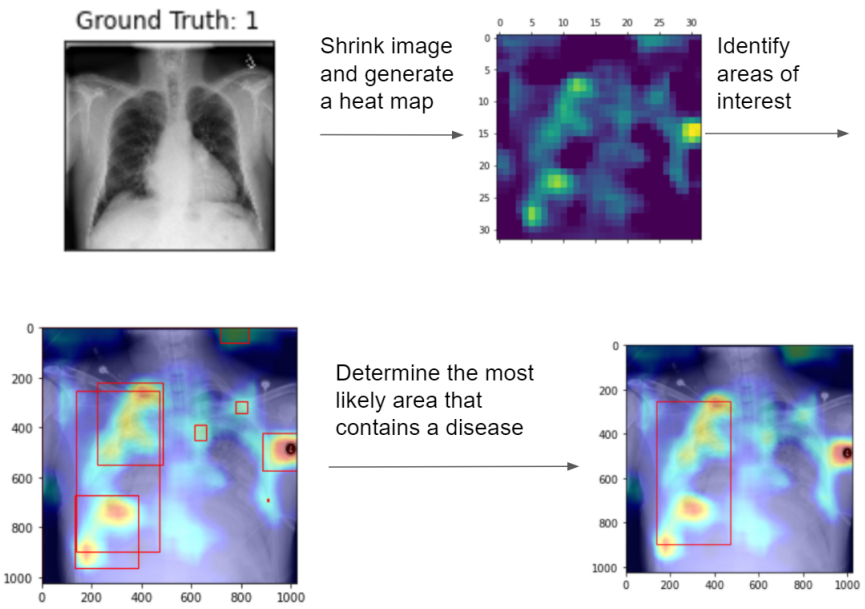

Fig. 1. Flowchart of our design

After researching the variety of approaches that can be used to establish a bounding box around a region of interest in an image, we are confident that we will be able to effectively accomplish our goals with the use of a convolutional neural network. This tool will provide value to all of our potential users, including athletic trainers, medical professionals, and future researchers.

2 BACKGROUND

The focus of this project is object localization on images through learning-based models. In our research, we discovered and investigated quite a few distinct ways to approach this problem. One such method is using a dynamic decision process [1] which considers the whole image to be a bounding box. A combination of eight transformations is then used to manipulate the bounding box: the box can be shifted left, right, up, or down, scaled larger or smaller, widened, or heightened. Markov's decision process then works by holding a tuple (O,h), where O is the feature vector of the current box that will be put through a pre-trained CNN, and h is a history vector of the previous 10 actions. When the feature vector gets put through the CNN, its results are valued and then used to see if the next state of the box would be more valuable than the current state; if not, then another transformation can be tried.

An alternative approach to generate a bounding box involves semantic segmentation [2]. The goal of semantic segmentation is to classify every single pixel to a certain group. As it groups the pixels, it assigns them a value and recolors an output image. Once the classification step is completed, semantic segmentation also works to provide a localization for the groups and give a spatial location. This has seen a lot of work be done in the autonomous vehicle industry [3] as well as medical imaging [4]. Our final investigated approach to this problem was with the using a sliding window, which takes a fixed size, rectangular window within the original image and moves it across an image checking for objects of interest. While this still is an effective method, its computational cost puts it at much more of a

disadvantage [5] due to how many times it has to run over the input image. Learning about these approaches focused our efforts when thinking about potential solutions.

3 DESIGN

The design is centered around three main components: Image Preprocessing, Classification, and Object Localization. In order to make use of our database of x-ray images, we must preprocess them so that they can be input into a model. We achieve this by converting each image from a .png style format to a multidimensional array of pixel color values. For our classifier, we want it to take an inputted image and return a label of which disease that image contained, if any. We take our arrays of pixels and train them on a pre-trained ResNet-18 model, and again on a split train/test disease vs. no disease dataset. We do this for each subset of diseases in order to obtain working binary classifiers for each disease. Lastly comes localization. Not only do we want our model to be able to tell us if there is a disease in a specific x-ray, but where it is as well. After locating the disease, our model will draw a bounding box on the image surrounding the disease.

The main steps for our entire design are:

1. Load in the dataset
2. Create a data loader and call it on individual diseases as preparation for binary classification.
3. Load in a pre-trained ResNet-18 model using Torch.
4. Create separate deep copies of the model and run each individual disease set on a separate copy.
5. Create a function that loads in an image from its path and uses Torch to create a heatmap from the activations of that class' disease model.

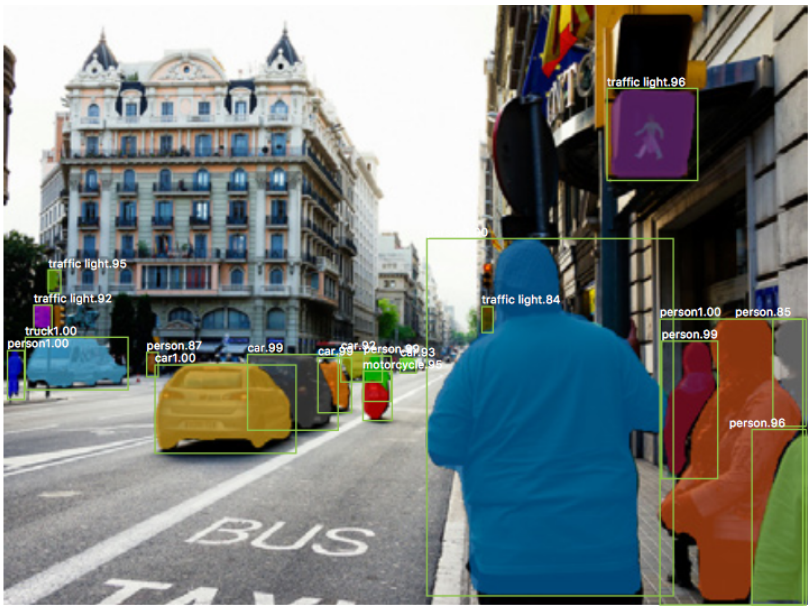

Fig. 2. A depiction of object detection being used in an autonomous vehicle [6]

3.1 Object Recognition

There are three subsections to object recognition: Classification, Object Localization, and Object Detection. We already covered the first two so this will mainly be about object detection. Object detection takes it a step further. Instead of being able to classify images and locate the presence of objects/diseases, it can identify, locate, and label many different objects and diseases. Figure 2 is from the perspective of a self-driving car - a technology that loves using object detection. In a matter of seconds, the car is able to classify, locate, and label up to hundreds of objects at once. In this image, we can see bounding boxes labeled with objects such as cars, trucks, people, and even traffic lights. This is the same idea we want to apply to our model. We are currently only classifying one disease at a time. Our model compares pictures of one single disease to undiseased images. As we finalized our product, instead of just binary classification we were able to correctly classify any image from the entire dataset.

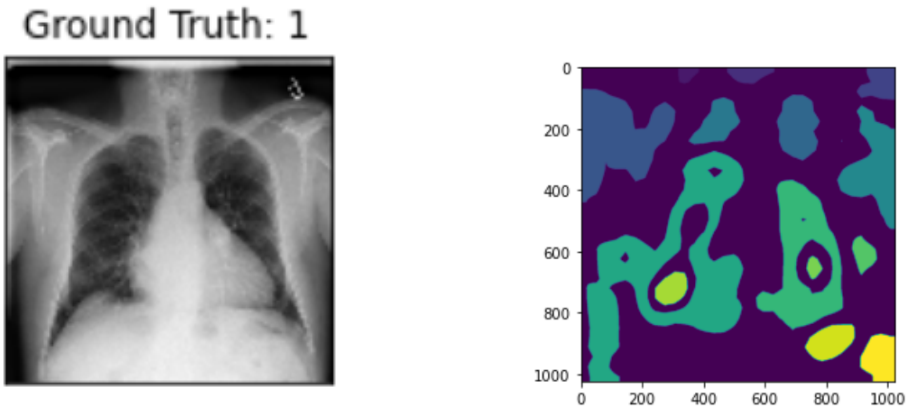

Fig. 3. Using Grad-CAM on an image to develop a heat map

3.2 Gradient Class Activation Maps

When dealing with deep learning models, it is often hard to tell how your model arrives at a conclusion for a label. This is where Gradient Class Activation Maps(Grad-CAM) come in handy. A Grad-CAM creates a heat map visualization from the last layer of a Convolutional Neural Network (see Figure 4) for a specific label. This heat map allows us to see which parts the model is learning to identify. There are two main uses for this. One is to use the heat map to finetune your model if you see that it is not detecting the right thing; the other is to use a well-learned visualization for post classification results. We can see an example of heatmap generation on our images in Figure 3. The model uses that heatmap to obtain coordinates for placing bounding boxes on the areas it is most interested in. We can then trace these coordinates back to the original x-ray to decrease cluttering and obtain a final product.

4 IMPLEMENTATION

The main platform used to create our design is Jupyter Notebooks, an interactive environment that allows us to input code into individual cells. This simplifies the process of data collection, preprocessing, model testing, and statistical

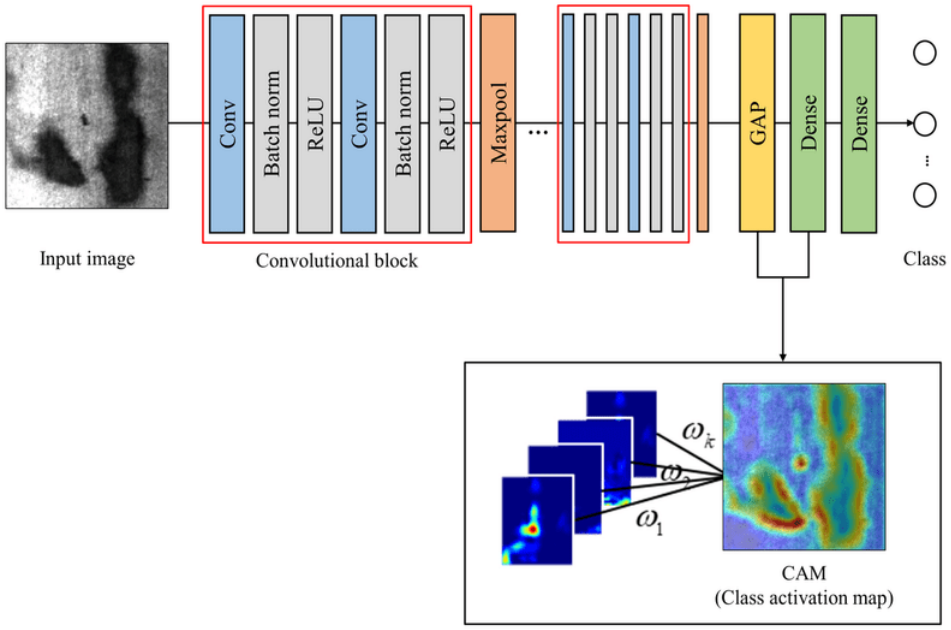

Fig. 4. GradCAM learning from the last layer of a CNN.
[7]

analysis/presentation. Our chosen programming language for this project is Python. Python is not only easy to use and understand, but it is very useful in data science because of its available libraries.

4.1 Libraries

There are quite a few libraries we used to develop our design and previous models we built off of. Torch allowed us to import many functionalities that helped with building out Convolutional Neural Network. ResNet-18 is a pre-trained model we used to initially train our data on. Once we ran our data on the pre-trained model, it made the learning process of the data a lot simpler. NumPy allowed us to use python to store any measures we had into numpy arrays. This included prediction labels, actual labels, pixel arrays, and involvement with data preprocessing. SciKitLearn provided us with utilities that binarized our labels for preprocessing, splitting our data into training and testing, classification, and our ROC curves (more in Results). MatPlotLib allowed us to plot all of our plots for evaluation and our Grad CAM images.

4.2 Data

The dataset provided to us is a Kaggle dataset available here [9]. This NIH dataset of over 110,000 images also provided two key .csv files. The first one contains important data entries of every single image with metadata of the patients (gender and age), what disease the image has, and whether the image is anteriorposterior (AP: patient facing machine) or posterioranterior (PA: patient facing away from machine). The second one contains disease labeled data entries of images that already have a bounding box placed on them in the form of (x,y,width,height) coordinates.

4.3 Code

The main standard we tried to adhere to while building our code was just to make sure we did it ourselves. The reason for this is because we are using a dataset that already has an official paper [10] published about it. This means that the problem we are solving has already been done. So we want to build a classifier and object detector that works with the materials and resources that we find.

5 RESULTS

For our results, the two areas we need two evaluate are classification and bounding box locations. For our classifier, we use an ROC (Receiver Operating Characteristic) curve, which graphs the true positive rate with the false positive rate. This shows us how well our classifier is working and correctly labeling images with the correct disease. We are still working on our bounding box evaluation. In order to evaluate where our model places the bounding boxes, we have to train our box location on images from the NIH dataset that already contain bounding boxes. Unfortunately, there are not that many images within each disease that are like that.

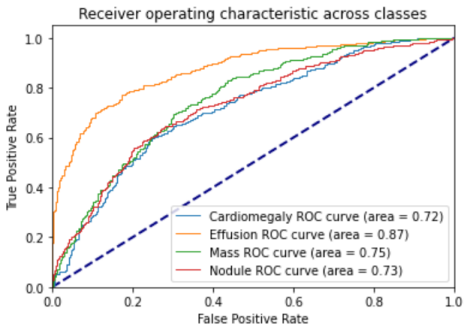

Fig. 6. ROC DeepMedia curve

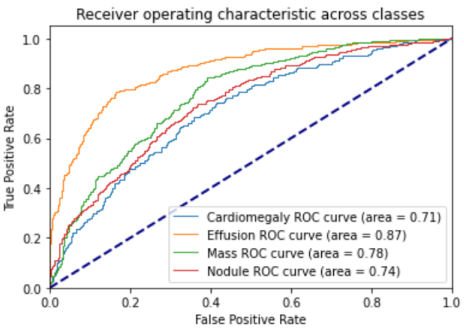

Fig. 7. ROC DeepMedia curve after metadata improvements

5.1 Classifier Evaluation

The goal of our classifier is to correctly label images with their corresponding disease(s). Any image that is correctly labeled with its disease is a true positive, whereas any image labeled with a disease that it does not have is a false positive. When we graph these two variables, it allows us to evaluate our classifier. Figure 4 represents the basic trends that ROC curves can take. A perfect ROC curve has 1.0 true positive at 0.0 false positive. A line with a slope of one shows a random classifier, 50% of the time it will be correct. As the curve begins to concave downward, that means the classifier is improving. The results from the current binary classifications we have are seen in Figure 6.

After running the initial classification on the image sets, I then used the metadata to try to improve the classification. The csv file containing the metadata has information about each image's Patient Age, Patient Gender, Image View, Follow Up Visit Number, and Image Size. The main info I used to try to increase classification accuracy was Patient Age, Patient Gender, and Follow Up Visit Number. I did not expect much improvement because it metadata could correctly classify, then we would not need the images at all. However, as seen in Figure 7, the metadata increased the

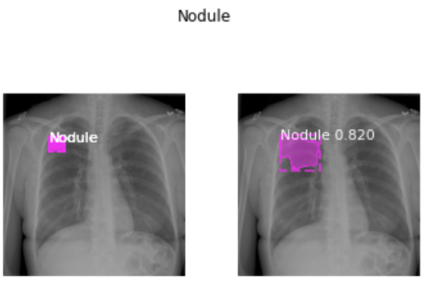

Fig. 5. Image depicting MaskRCNN evaluation and Intersection over Union calculation for an image in the Nodule class.

classification of classes by up to .03. It may not seem like a lot for a sampling of the dataset, but .03 increase of about 110,000 images is another 3,300 correctly classified images just from the metadata.

5.2 Bounding Box Evaluation

When our Grad CAM heatmap provides many areas of interest on an x-ray, it can cause a lot of confusion when we interpret the bounding boxes. Figure 6 is a good example of this. There are so many bounding boxes on the image that it just looks cluttered. To further understand which boxes to use, we first evaluate these using Intersection over Bounding Box (IoBB). To put it simply, if we take an image that already has a bounding box and overlay where our model put one, then the IoBB is simply the area of overlap over the area of union. Looking at Table 1, we can see the direct comparison of our evaluation to the NIH paper. There are a good amount of areas where our evaluation is better than the NIH paper so these results are something our team in content with.

Another way we were able to evaluate our bounding boxes was with MaskRCNN [8]. We used a similar evaluation method, but instead of bounding boxes, it just used the ratio of how close the model was to encasing the correct area. Figure 5 shows an example of MaskRCNN being used on an image in the Nodule class and its corresponding IoU score. A total summary of some diseases IoU scores can be seen in Figure 7.

6 DEVELOPMENT PROCESS AND CHALLENGES

The senior design project as a whole went very well. Meeting twice a week with our client and more with the team kept us up to date with what we were working on. The agile process itself always makes an assignment feel like a real life project. However, It did not go so cleanly the whole year. We ran into a handful of issues that slowed down our

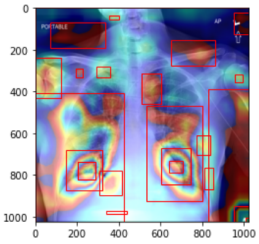

Fig. 6. Image from our localizer that shows how many areas of interest can be generated from one x-ray.

T(IoBB)	Atelectasis	Cardiomegaly	Effusion	Infiltration	Mass	Nodule	Pneumonia	Pneumothorax
T(IoBB) = 0.1								
Deep MedIA	**0.806**	**1.0**	**0.806**	**0.96**	**0.883**	**0.938**	**0.958**	**0.55**
NIH	0.7277	0.9931	0.7124	0.7886	0.4352	0.1645	0.7500	0.4591
T(IoBB) = 0.25								
Deep MedIA	**0.611**	0.967	**0.677**	**0.92**	**0.765**	**0.625**	**0.875**	**0.4**
NIH	0.5500	**0.9794**	0.5424	0.5772	0.2823	0.0506	0.5583	0.3469
T(IoBB) = 0.5								
Deep MedIA	**0.472**	**0.9**	**0.419**	**0.64**	**0.353**	**0.375**	**0.583**	**0.2**
NIH	0.2833	0.8767	0.3333	0.4227	0.1411	0.0126	0.3833	0.1836
T(IoBB) = 0.75								
Deep MedIA	**0.333**	**0.867**	0.129	0.28	**0.294**	**0.188**	**0.333**	0.0
NIH	0.1666	0.7260	**0.2418**	**0.3252**	0.1176	0.0126	0.2583	**0.1020**
T(IoBB) = 0.9								
Deep MedIA	0.083	0.3	0.032	0.12	**0.118**	**0.125**	0.0	0.0
NIH	**0.1333**	**0.6849**	**0.2091**	**0.2520**	0.0588	0.0126	**0.2416**	**0.0816**

Table 1. Comparison of our Bounding Box Evaluation to the NIH dataset

whole project for a while. The first issue we encountered was file checkpoints. Our Jupyter Notebooks kept crashing and running into errors until we visited the tech desk and they told us to use file checkpoints in our home directory. The next error we ran into was file permissions. Even though we work in a shared folder, any files created or added to the directory only had permissions given to that individual. If one of us needed access to a file/folder, we would have to send a message and wait for a response from our teammates. We were all at school working on it so it was never a major issue, but it could be a day before a response. We still have this issue, but it is nothing that prevents us from working for too long. The last ongoing issue we have is BisonNet. BisonNet was supposed to improve our training time and make everything run smoothly, but we were never able to get it running. We spent a lot of our first semester trying to get BisonNet functioning, but we have decided to give up on it. We are able to train on GPU, and while it is not as effective as BisonNet, it is still manageable.

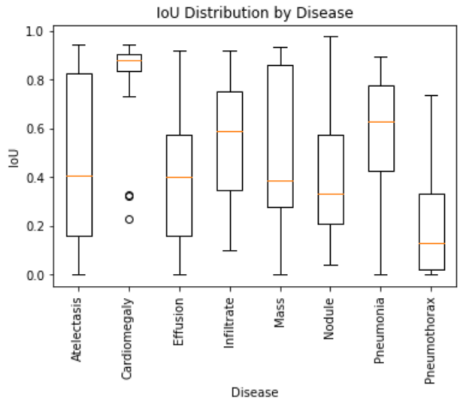

Fig. 7. Graph depicting Intersection over Union calculations in a boxplot for multiple class.

If I could have done this project again, I would have tried to talk with our client and try to get BisonNet working before we really started work. I also would have tried to get an object localizer working earlier. We waited until our classifier began working to begin working with object detection.

7 CONCLUSION

After struggling with many issues during the first semester that set us behind, our progress this semester has increased by a large margin. For the final release, our team is happy with the end result. Our classifier is working properly and our GradCAM and MaskRCNN metrics are allowing us to efficiently generate bounding boxes on all images. We are very lucky that the NIH paper gave us a benchmark for our evaluation goals and our team is glad that our performance improved upon that of the paper.

ACKNOWLEDGMENTS

To Joshua Stough, for leading our project and keeping us updated on what steps to take.
To Robert Nickel, for assistance in machine learning and what models may help our project.
To Zach Stephens, for providing guidance as an athletic trainer on what he would like to see from our project if he could use it.

REFERENCES

[1] Caicedo, J., Lazebnik, S. 2015. Active Object Localization With Deep Reinforcement Learning. *In Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pp. 2488-2496. Computer Vision Foundation from https://arxiv.org/abs/1511.06015

[2] Ulku, I., Akagunduz, E. 2019. *A Survey on Deep Learning-based Architectures for Semantic Segmentation on 2D images.* In Computer Vision and Pattern Recognition. arXiv. from https://arxiv.org/abs/1912.10230

[3] He, K., Gkioxari, G., Dollár, P., Girshick, R. 2017. *Mask R-CNN.* In Computer Vision and Pattern Recognition. arXiv. from https://arxiv.org/abs/1703.06870

[4] Taghanaki, S., Abhishek, K., Cohen, J., Cohen-Adad, J., Hamarneh, G. 2019. *Deep Semantic Segmentation of Natural and Medical Images: A Review.* In Computer Vision and Pattern Recognition. arXiv. from https://arxiv.org/abs/1910.07655

[5] Lampert, C., Blaschko, M., Hofmann, T. Beyond sliding windows: Object localization by efficient subwindow search. *In 2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2008),* 24-26 June 2008, Anchorage, Alaska, USA, 2008. from https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/34843.pdf

[6] Ajay Uppili Arasanipalai. May 20, 2018. *State of the art deep learning: an introduction to Mask R-CNN* from https://www.freecodecamp.org/news/mask-r-cnn-explained-7f82bec890e3/

[7] Ankit Choraria. October 10, 2020. *Class Activation Mapping in Deep Learning* from https://www.loginradius.com/blog/async/class-activation-mapping/

[8] [n.d.] *How Mask R-CNN Works* from https://developers.arcgis.com/python/guide/how-maskrcnn-works/

[9] [n.d.] *NIH-Chest-XRays* from https://www.kaggle.com/nih-chest-xrays/data

[10] [n.d.] *ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases* from https://openaccess.thecvf.com/content$_{c}vpr_{2}017/papers/Wang_{C}hestX-ray8_{H}ospital-Scale_{C}hest_{C}VPR_{2}017_{p}aper.pdf$