# Group M - Deep MedIA Technical Report

ANDREW DRESSER, Bucknell University, Department of Computer Science

Additional Key Words and Phrases: Mask R-CNN, PyTorch Database

## 1 INTRODUCTION

As machine learning continues to become more and more prevalent, the medical image field is starting to see some potential benefits from these advancements. Alongside the explosive growth in healthcare imaging which has led to steady growth in the amount of time physicians must spend interpreting this data. The medical field is left with a couple options. Either hire more physicians in order to be able to handle the current throughput of medical images or look towards machine learning alternatives to be able to increase the throughput that any given physician can handle. While hiring more physicians is enticing in the sense that it would create more jobs, it can be very expensive for hospitals and private practices to have to do this. And it can be difficult to find individuals who have the necessary qualifications. With this being said, an alternative approaches utilizing machines learning is becoming increasingly appealing.

The product we designed has the capability of loading in a Chest X-Ray image, classifying the diseases in the image (from a specific set of diseases). And then localizing the disease(s) within the image. Then outputing the properly labeled disease(s) to the user.

For this project, the Chest X-Ray dataset from the National Institute of Health [5] will be utilized. Chest X-ray exams are one of the most frequent and cost-effective medical imaging techniques available. However, can make it difficult for doctors and physicians to properly diagnose diseases that are present. Imaging techniques such as CT scans can be much more effective examinations to diagnose from. Due to the lack of large publicly available datasets with annotations makes it very difficult, if not impossible, to achieve clinically relevant computer-aided detection and diagnosis (CAD). The lack resources for labeling so many images has been one of the major hurdles in creating a chest X-ray dataset of this scale.

This NIH Chest X-ray Dataset is comprised of 112,120 X-ray images with disease labels from 30,805 unique patients. To create these labels, the authors used Natural Language Processing to text-mine disease classifications from the associated radiological reports. The labels are expected to be >90% accurate and suitable for weakly-supervised learning.

While this does not mean that doctors no longer need to look at chest x-rays at all, it could expedite the process of them. And also make it so less chest CT scans are necessary.

Over the course of the development of the product, my contributions consisted of researching the topic in order to understand how our implementation would be approached. Setting up an image loading system. So, designing and creating the file structure and then designing and writing a DataLoader. Then, in addition to this, working with Mask R-CNN to localize the diseases that were present in the image.

## 2 BACKGROUND RESEARCH

Before diving into my individual contributions, it is necessary to understand all of the working components of the project. Meaning what components are a part of the product and understanding why they were chosen to be designed and implemented in the easy they were.

Author's address: Andrew Dresser, aed018@bucknell.edu, Bucknell University, Department of Computer Science, , Lewisburg, PA, 17837.

As mentioned earlier, the product is designed to be a tool that trainers, doctors or nurses could use in order to classify and locate diseases in X-ray images. So our implementation is comprised of the following components. Disease classification, data loading, object localization and finally a UI to bring all of this together.

## 2.1 Disease Classification

In completing the tasks of classifying the diseases inside of an image, there were a variety of different tools that could be used. In this section the various options will be discussed, accompanied by the rational for choosing the best option.

*2.1.1 Convolutional Neural Network.* Convolutional neural networks (CNN) are a machine learning structure that analyzes local image regions to detect features in the overall image. To start, they take in an image as a two-dimensional array of pixel color values in some colorspace, providing a three-dimensional tensor as input. This process is repeated until a flat layer is ultimately used, which is then either reduced to a single output (e.g. classification) or recombined into a more complicated output as needed by the problem. This allows for the analysis of complex images without prior understanding. However, like all machine learning techniques, this approach requires a sizable data set from which to develop its internal model.

A CNN could be used in a few different ways for bounding boxes, such as segmentation or coordinate output. For segmentation, the network would act to create a mask that marks which pixels are part of the region of interest, while coordinate output would provide the coordinates of a rectangular bounding box's corners. Both of these approaches would provide an output that could be visualized and compared to existing data samples.

*2.1.2 Generative Adversarial Network.* Generative adversarial networks (GAN) are a different machine learning mechanism that could be used to determine bounding boxes. Since we are looking to generate these boxes, a GAN seems like an interesting possibility. GANs use competing neural networks to replicate images that seem indistinguishable from those in the source data set. The generator network is tasked with creating source-like images, and the discriminator tries to determine which images are real or generated. This approach has both the generator and discriminator grow in tandem, ultimately providing a system that can generate an output contextually similar to the training data.

In our use case, the network would be trained on pairs of images: the original image and the image with a bounding box drawn on top of it. This would create a system that can convincingly generate bounding boxes around areas of interest in the image, which is what we are aiming to create.

*2.1.3 Spatial Pyramid.* While neural networks are one of the more common and popular systems for understanding and potentially classifying an image, they are not the only approach. Before the computational power required for neural networks was as prevalent as it is today, alternative approaches were used for these types of problems. As previously discussed, segmentation is a common approach for analyzing images, but this is not how the human mind approaches these kinds of problems. Instead, it processes high-level features at a holistic level without the need to segment the image into smaller components, which is the procedure that a spatial pyramid works to replicate. The spatial pyramid approach uses histograms to evaluate the contents of the image as well as of smaller and smaller subimages before combining the feature analysis with a weighting system. By weighting the multiple layers of detail differently, different levels of features can be identified and understood without the need for a localized breakdown of different features. This differs from CNNs in its globalized approach to image analysis, providing context of the overall image that can be used to identify local regions.

*2.1.4 Decision.* Through the above analysis and rational it was believed that a Convolutional Neural Net would be the best option in order to complete the task of disease classification.

## 2.2 Data Loading

In order to provide the simplest and most effective way to load in data, a PyTorch Dataloader [2] is the industry standard. With this solution, you must first write a custom Dataset object for your data. In this Dataset, you define a __getitem__() function that can retrieve data to your specifications. And then use the Dataloader object to actually load in your data. This allows to make the process far more efficient.

Due to the customability of the Dataset object, this approach could be taylored to the specific needs of our dataset. This means that the files could be restructured based on metadata and the dataset could be modified to use this to it's advantages.

## 2.3 Object Localization

*2.3.1 Mask R-CNN.* In trying to accomplish Object Localization, there was less of a clear option. The two potential solutions, that we came across were Mask R-CNN and Gradient Activation Maps.

Mask R-CNN is an approach that uses semantic segmentation to produce a bounding box. The goal of semantic segmentation is to classify every single pixel and them to a certain group. As the pixels are grouped, the image is recolored. 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 as well as medical imaging.

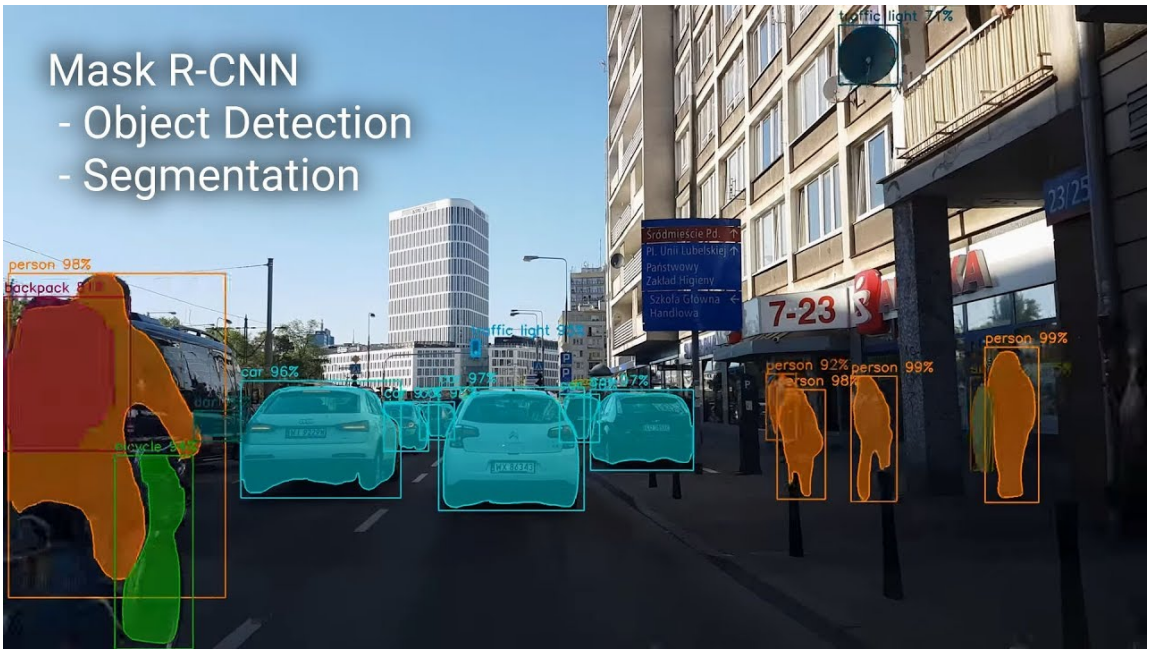


Fig. 1. Diagram of the layers involved with Mask R-CNN

While Mask R-CNN is frequently used in the way we are using it, our dataset did not quite provide the right infrastructure in order to accomplish Mask R-CNN in proper way. Typically Mask R-CNN requires training images that have every single pixel of the image classified. In this dataset, all that was available was the mask boxes. So we could roughly classify every pixel in the image, however, this would classify pixels around the disease as the disease. However, there were other studies that had done this only using the bounding box coordinates. Specifically this study [6] that used Mask R-CNN on chest X-Ray images for Pneumonia.

*2.3.2 Gradient Activation Maps.* Another potential solution was Gradient Activation Maps [7]. The idea behind these, is that it uses network "focus" to identify regions that may be affected. One potential benefit from this over Mask R-CNN is that it can help evaluate overfitting from the classifier. Meaning that if the classifier was having troubles, this tool could potentially be used to investigate what is causing the classifier to err. And similarly to Mask R-CNN there were studies out there that had used Gradient Activation maps that were used in order to localize diseases in an X-ray

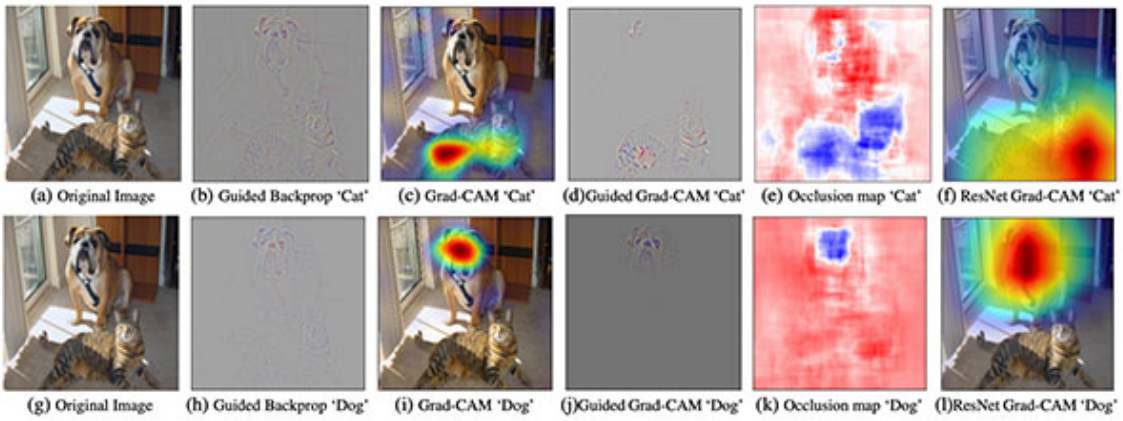


Fig. 2. A Gradient Activation Map Examples

*2.3.3 Decision.* Due to the positive benefits posed by both Mask R-CNN and Gradient Activation maps. There was no sense in eliminating either as an option. There had been different studies that had been successful using both of the different solutions. Given the size of our team it made more sense to pursue both, ultimately resorting to the solutions with the better outcome.

### 2.4 UI Design

While not mitigating this part of the project, it was not the main focus or most important aspect of the product. A good UI is only useful if the functionality is there. So the goal was to create a UI that was capable of the following:

(1) Loading in an image,
(2) Classifying the diseases present in the image,
(3) Localizing the diseases that are present in the image,
(4) Outputing an image to the user that draws bounding boxes around the disease(s) and labels them.

In order to accomplish this, it would be best to proof the functionality in a terminal UI. In order to minimize outside error, first showing the functionality is there would be easiest to do in a terminal UI. And then from here, this can be easily transformed into an application.

## 3 DESIGN

For each piece of my contributions, there were specific plans as to how they should be completed. However, as the year went on and the group did more research, it was understood that this was subject to change.

### 3.1 High-Level Description

*3.1.1 Data Loading.* In trying to develop an efficient system to retrieve images, there were two major components. The first being to design and create a file structure. This file structure needs to work optimally with the Dataloader object that then needs to be created.

*3.1.2 Mask R-CNN.* The approach here was different, as the topic first needed to be researched. Then it needed to be applied to a dataset that was confirmed to work with it. Essentially running a tutorial to ensure that Mask R-CNN will work on the Academic East Room 116 Machines. Before jumping into all of the diseases, it would be beneficial to first test Mask R-CNN on a singular disease and then translate this to all of the diseases.

### 3.2 Detailed Design Process

*3.2.1 Data Loading.* When designing our image retrieval process, our images needed to be organized in a way that would be most optimal for our loader. This required understanding the metadata that came along with our images, as well as knowing exactly what the end goal was.

Some of the most important attributes of the dataset were the orientation of the data [AP, PA], and disease [Atelectasis, Consolidation, Infiltration, Pneumothorax, Edema, Emphysema, Fibrosis, Effusion, Pneumonia, Pleural thickening, Cardiomegaly, Nodule, Mass, Hernia]. In addition, the data would then need spliting into training and testing sets. Having the data preliminarily split into orientation. This is because when doing our classification training and object localization, 'AP' or 'PA' orientations would need to be trained separately as it was unclear if the models being used would view the different orientations in the same or different ways. In addition to orientation, breaking down the data by disease is essential. So when the data is being pulled to train or test, once the orientation and disease(s) are defined, the remainder of the task is easy. The last desired data loading feature is a train/test split. However, it was unclear what the test/train data ratio should be. So, instead of defining a test and train set, through a proper construction of the Dataset object, the images could easily be split into training/testing sets with any desired ratio, while remaining in the same directory.

There was no real priority between the orientation or disease because the Dataloader could handle it with ease either way. So the file structure ended up being split by disease and then by orientation. After this, it was not worth considering the training/testing split as the Dataset object could do this for us.

From here all that needed to be done was to specify the Dataset parameters in a way where it was very easy to pull any section of diseases of a specific orientation for either training or testing. In order to do this, I defined the Dataset with the following parameters.

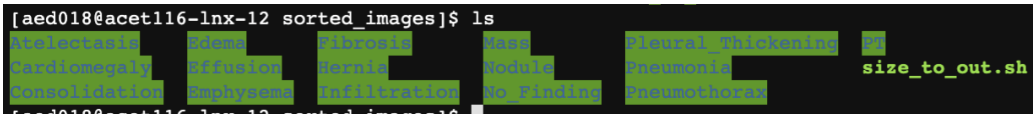


Fig. 3. Images organized by disease

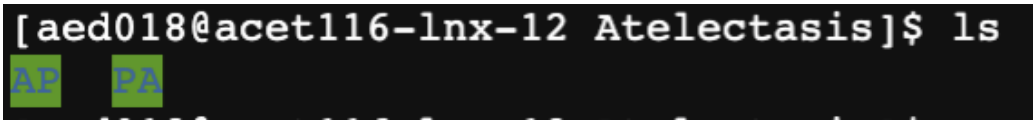


Fig. 4. Images organized by orientation

```
Params: data - the data dictionary
        view - the orientation you want to look at
        diseases - the diseases you would like to look at
        num_imgs - the number of images of each disease you would like
        factor - the ratio of training and testing data
        typ - 0 for training, 1 for testing
```

Fig. 5. Parameters set for the Dataset object

*3.2.2 Mask R-CNN.* In researching Mask R-CNN the most helpful resource was *Computer Vision and Pattern Recognition* [3]; one if not the most widely recognized papers regarding Mask R-CNN. In addition to this, there are other examples of how Mask R-CNN can be used with Chest X-Rays [6].

Our dataset has thirteen different diseases, but only eight of the diseases contain bounding box metadata. To eliminate the possibility of environment issues, first a demo called PennFudan was set up in our space [1]. And got outputs such as the following:

Fig. 6. Output localization image from the PennFudan Dataset

In order to confirm functionality of Mask R-CNN in our dataset, it was deemed a better idea to tackle a singular disease instead of all eight at the same time. This is just to confirm that Mask R-CNN would work, before getting to far in the implementation. Since not all of the images had bounding box metadata, those that did were filtered out to a separate folder. The disease with the most number of images was Atelectasis with 180 images, so it was most fitting to try the single disease Mask R-CNN with this this disease

Since there was a limited amount of images with bounding boxes, a pretrained model from the coco dataset was used. Then the images along with their masks, generated from the bounding box data, were fed into the model to update the edge weights. This translates the coco model to a model that can localize Atelectasis.

To translate this to a multi-disease model, involved a few modifications to the initial training process. One being to change training configuration to now represent the nine classes (the eight disease classes and no finding). Then the disease data subset needed to be updated in order to accommodate all of the disease that contain bounding boxes. In addition to this, the csv files for the metadata had to be updated to correspond to the multi-disease dataset. Aside from these modifications, producing the trained model ran identically to the singular disease class.

## 4 IMPLEMENTATION

### 4.1 Python Notebooks in Jupyter Lab

For the implementation of these components, all of it was done in python notebooks using Jupyter Lab. Jupyter Lab made remotely accessing the Academic West room 116 machines to do our work very easy. On these machines, there is a shared space where all of the files are stored. Jupyter Lab makes it very easy for us to be able to open up these notebooks regardless of our teams location. For reference COVID-19 is happening.

In addition to this, Python notebooks allow easy visualization of output. And provides the functionality to run only certain pieces of a document while leaving others idle. In order to run the notebooks, each cell can be ran individually by pressing 'ctrl-enter' or hit the run button at the top of the window.

### 4.2 Libraries

Indside of python, there are a few libraries that were used. The most notable of these being from torch.utils.data, the Dataset and Dataloader classes were used to Implement the Data Loading. And the Mask R-CNN library was used to implement Mask R-CNN.

In order to use the Dataset and Dataloader classes, all that needed to be done install all of the necessary dependencies and then insert the import statements. Then these two classes were used to define the dataset and also to load the data in. The dataset is a class that was extended, and then redefined to a class that met the specifications of our data, For the DataLoader nothing needed to be redefined.

Mask R-CNN was more complicated to get to a functioning state. The Mask R-CNN repository first had to be cloned into the work space [4]. Then a conda environment had to be created with all of the proper dependencies installed. And then finally, a new kernel has to be created specifically for this workspace. Once this is set up, Mask R-CNN is ready to be used.

### 4.3 Our Data

The dataset being used, is from the National Institute of Health Chest X-Ray Dataset. This dataset contains 112,120 Chest X-Ray images. These X-Rays consist of different diseases, that are thought to have an accuracy of greater than

90 percent. Included with this data are a few documents containing metadata for the images. The first is a document called 'Data_Entry_2017.csv'. Which contains the following metadata about all of the images: 'Image Index', 'Finding Labels', 'Follow Up#', 'Patient ID', 'Patient Age', 'Patient Gender', 'View Position', 'OriginalImage Width', 'OriginalImage Height', 'OriginalImagePixelSpacing_x', and, 'OriginalImagePixelSpacing_y'. Then along with this, there was a file called 'BBox_List_2017.csv' which consisted of the following categories for images that had bounding boxes (which is a pretty small percentage of them): 'Image Index', 'Finding Label', 'x', 'y', 'w', and 'h'.

Here is an example image from our dataset:

Fig. 7. NIH X-Ray Image

### 4.4 Evaluation

In order to evaluate the results, a few tactics were used. In order to verify that the dataloader was working properly, the most effective way was to simply see if the proper images were being loaded in.

On the other hand the performance of Mask R-CNN is more difficult to evaluate. To ensure a thorough evaluation, a few different metrics were used. The first being a validation set that will output a predicted bounding box on the image. This output can be seen in figure 8. The visual feedback can be very helpful to understand where Mask R-CNN is doing well and where it is doing poorly. For example if the predictor was consistently mislabeling images but consistently mislabeling them in the same way, then this may become very apparent in the outputed images. However, actual performance metrics also need to be used in order to evaluate the success of Mask R-CNN. So in order to this, Intersection of Union (IoU) will be used given by the following expression. Where A is the given bounding box and B is the predicted bounding box.

$$IoU(A, B) = \frac{A \cap B}{A \cup B}$$

Essentially this is calculating the intersected area and dividing it by the total area that the two boxes are covering. The NIH published some metrics on their IoU outputs that can be compared to the results from Mask R-CNN.

## 5 RESULTS

### 5.1 Data Organization

For the data organization the results are a correct file system and for the loader, it is a loader that properly loads in the data. This can be most easily seen in the the classifier loading in the proper images properly. There is no other way to demonstrate the effectiveness of this implementation.

### 5.2 Mask R-CNN

From training the single disease classifier, the following validation dataset in figure 8 was created created.

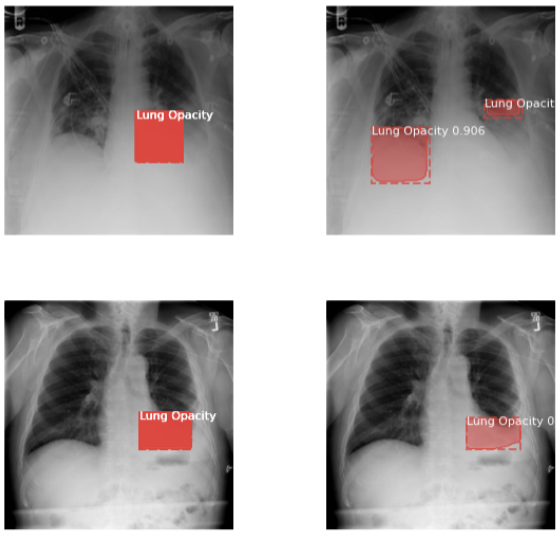


Fig. 8. X-Ray Validation Dataset

In the above figure, the left column represents the image with the given mask pasted over it. Then on the right is the image with the predicted bounding box over the image. This is obviously not perfect, but proof that pursuing with multiple diseases was viable.

With the single disease localization, IoU was never implemented as this was just to see if Mask R-CNN was worth pursuing further. This was enough to determine that it was. Then for multidisease localization, the validation output in figure 8 acquired.

As before the left images are the original image with their given masks. And then the right side is the image overlayed with the predicted mask using Mask R-CNN. This is just a sample of three images from the entire dataset. And similarly to before this is not perfect, but as you can see most of the predicted masks have a relatively good overlay.

The better metric for the performance however is the IoU. In order to evaluate IoU, the overall average IoU was calculated as well as the average IoU by disease. The output in figure 9 lays out the IoU by disease as well as the weighted average over all diseases.

The overall weighted average of the IoU was 0.505. This ranged from 0.227 with Pneumothorax to 0.818 with Cardiomegaly. Results that appeared to be good considering the IoU that was being aquired from the Gradient Activation

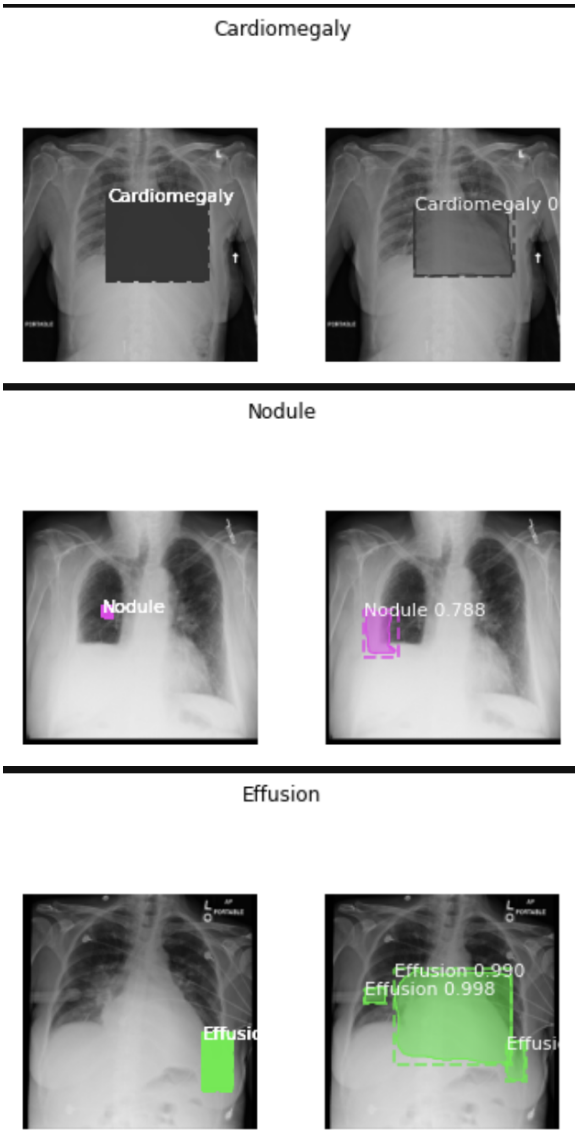


Fig. 9. Multidisease X-Ray Validation Dataset

Maps. From the Grad Maps there was an average IoU of VALUE. Ranging from 0.048 with Nodule to 0.391 with Cardiomegaly.

However, the real question was how these results stacked up against the results that the NIH acquired from their models. Table 2 illustrates how our IoUs stack up those of the NIH at different threshold values.

This table shows five different IoU thresholds, [0.1, 0.25, 0.5, 0.75, 0.9]. For each of these threshold it shows the percentage of tests that value or higher. In general our IoU had better performances for lower thresholds (0.1, 0.25

Table 1. IoU Analysis.

| IoU | |
|---|---|
| Atelectasis | 0.46 |
| Cardiomegaly | 0.818 |
| Effusion | 0.393 |
| Infiltrate | 0.554 |
| Mass | 0.478 |
| Nodule | 0.425 |
| Pneumonia | 0.579 |
| Pneumothorax | 0.227 |
| **Weighted Average** | **0.505** |

Table 2. IoU Comparison to NIH.

| 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** |

and 0.5). Then for the 0.75 threshold it still performed slightly better overall, but at the threshold of 0.9 it tended to to slightly worse for most diseases.

What this means, is that the Mask R-CNN model that we trained is better at getting at least some overlap of the localized diseases. But the NIH was better at getting very closely overlapped bounding boxes. In this case it is very difficult to definitely say whether one or the other had better results. However, given that our model was better at localizing at least part of the disease, this suggests that through better hyperparameter tuning, the model may have been able to more finely overlay image. On the other hand however, this could suggest that our model is creating large masks that are overlapping the disease, but are also highlighting large areas that are not affected and sort of getting 'lucky'.

Lastly the distributions for the IoU's by disease are highlighted in Figure 10

There are a few distributions here that are worth pointing out. The first is Cardiomegaly. The outlier threshold for this dataset sat around 0.75 IoU. However there are clearly a few IoUs that are sitting around 0.2-0.3. This means that without these few outliers the average IoU would be even higher that 0.818. Cardiomegaly is most likely like this,

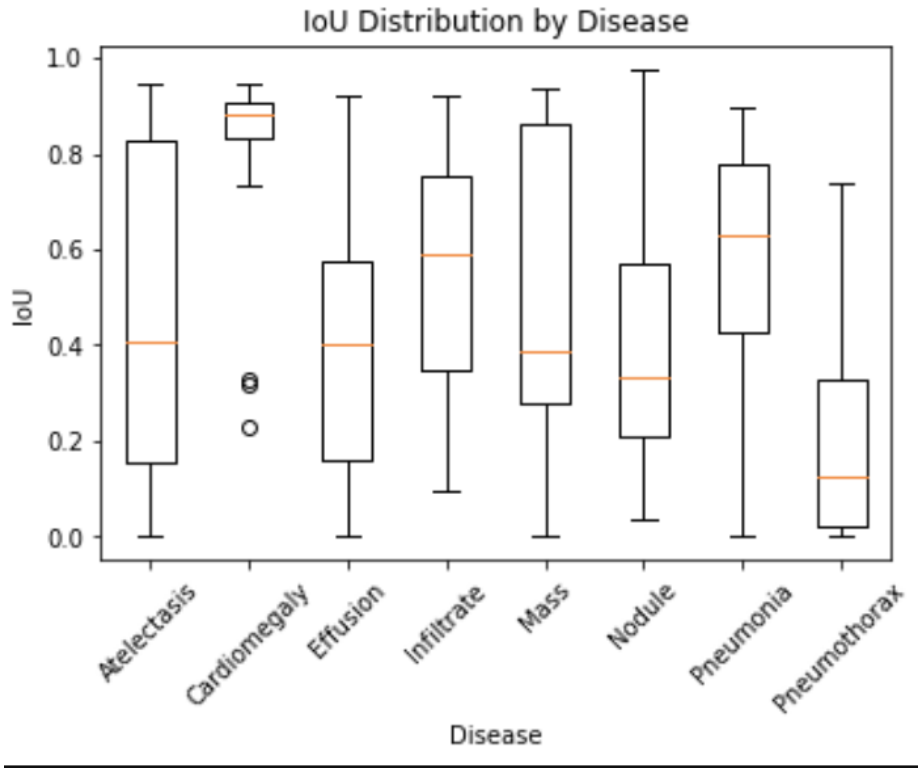


Fig. 10. These are the distribution of the IoUs for each disease

because it always occurs in the same location of the chest X-Rays. Because of this it is very easy for Mask R-CNN to localize the disease.

Now looking at Pneumothorax, it is clear that a very opposite trend is occuring. The upper outlier bound is at around 0.55 IoU. Essentially showing that it is difficult to even get close to localizing this disease. The other six diseases tended to have a more standard range of IoUs that went from 0.0-0.9.

From looking at these two examples it is clear that our product may be much more useful in localizing in some diseases than others. In the case of Cardiomegaly it would be useful, however, a doctor would know that if an X-Ray had this disease that it would alwasy be in the same spot. And in the case of Pneumothorax it would not be as useful to use our tool as it is not very accurate in localizing the disease.

Based on these results, this is definitely a viable way to localize diseases in an X-ray. it may not be perfect, but it is understood that this may not be feasible. The goal is to aid doctors and physicians into being able to increase the throughput of X-rays they may examine and also to potantially decrease the number of CT scans that are needed. This would do just that.

## 6 CHALLENGES

Throughout the process of this project, numerous challenges have crossed my path. However, thankfully none of them had to do with our team dynamic and effort put in by the other members of my team.

### 6.1 Jupyter Lab Collaboration

Early on in our project, we experienced an issue where only one member of our team could access and modify documents in the workspace. Other members could not even make new Python Notebooks to work in. So I contacted with ECST to discuss the issue that was occuring. The solution that we came up with was to modify the following line we were using to access Jupyter Lab:

```
jupyter lab --ip=0.0.0.0 --no-browser
```

By adding the following tag to this line, we were able to solve the issue at hand:

```
--FileCheckpoints.checkpoint_dir=/home/[buck username]/jl-checkpoint
```

The issue was that when we were saving our notebooks all group members were saving them to the same checkpoint directory. So in order to fix this we simply had to all define our own personal checkpoint directories to avoid this issue. After adding this tag we never experienced this issue again. And as a side note, [buck username] is replaced with your bucknell username. So in my case that would be aed018.

### 6.2 Experience

Coming into this project, I had little to no experience with the task at hand. I had some machine learning knowledge, but knew nothing about any of the classes that I was working with. However, I anticipated this coming in and was up for the extra challenge. And all this really meant is that I had to do a little bit of extra research on the topics we were covering in order to catch myself up to speed. This mostly meant looking into the classes that we were using a little bit extra. So the libraries such as Mask R-CNN and the DataSet and DataLoader classes.

### 6.3 Waiting Time

Since we are working with such a large dataset, many of the tasks I needed to complete would require a lot of waiting while they are running. For example when trying to organize the files, locating and copying over 100K images takes time. And since we initally were unsure how we wanted to structure the files, I ended up having to do this multiple times. This was frustrating as I felt I was spending significant amounts of time waiting when instead of working.

Even worse than this was training the Mask R-CNN models. In order for a model to do sixteen epochs on about one hundred images it takes roughly 10 hours. This is difficult since my computer is remotely accessing the Academic East machines. So during the training I had to leave my computer on, and therefore I was forced to train these models over night when my computer can remain in the same location for an extended period of time undisturbed.

### 6.4 Bucknell VPN

This year I am living in an off campus house, so in order to access the Academic East machines, I have to log onto Bucknell's VPN. For some unknown reason Bucknell's VPN is incredibly slow with the WiFi I have in my house. My WiFi on it's own is fast and reliable, but when I log onto the VPN it becomes very slow. In addition to this, it also frequently will lose connection for a minute or two and then reconnect. This makes it very difficult for me to get any sort of work done when I am trying to work remotely. As much as possible I try to go to campus and get my work done there. However, during COVID times this is not always so easy to do.

Lastly, the VPN times out after 12 hours, and so when my models take 10 hours to train, I typically am pushing the edge of that time frame. In the event that the VPN does disconnect, the kernel of the Python Notebook shuts down and

I can lose any progress that was stored in the document. There are ways around this, such as consistently saving my data, but it is still a frustrating reality.

### 6.5 Mask R-CNN

With Mask R-CNN I have had a few issues. Those being with the environment and masks.

*6.5.1 Environment.* For much of this project, we have been using the python/3.7-deeplearn environment. This has been great for much of the project, but when it came to Mask R-CNN, it was not so great. The version of tensorflow that is in this environment is not compatible with mrcnn. The solution to this was to create a conda environment. This environment had the proper version of tensorflow along with all of the other necessary dependencies. Then with this, a new kernel had to be created that used the conda environment we specified. Here is the detailed list of instructions that I used in order to set up this environment:

```
# steps to create the python environment
conda create --name myproject python=3.7 conda
source activate myproject
git clone https://www.github.com/matterport/Mask_RCNN.git
pushd Mask_RCNN
sed -i 's/tensorflow>=1.3.0/tensorflow==1.15/' requirements.txt
sed -i 's/keras>=2.0.8/keras==2.0.8/' requirements.txt
sed -i 's/h5py/h5py==2.10.0/' requirements.txt
pip install -r requirements.txt
python setup.py install
popd
python –m ipykernel install --user --name=myprojectPython


# steps to run jupyter lab after the python environment has been created
# run these steps on one of the ACET 116 computers
export LD_LIBRARY_PATH=/usr/remote/cudnn-10.1-7.6.5.32/lib64:/usr/remote/cuda-10.
>>0/lib64:$LD_LIBRARY_PATH
# next step only needed if you exited environment above
source activate mypojrect
jupyter lab
# Select the kernel in the notebook and you are good to go
```

After creating this environment and kernel Mask R-CNN worked with no issues.

*6.5.2 Masks.* When using Mask R-CNN, typically the masks that you have are exact outlines of the objects you are looking at. However, we did not have this so instead we had to use the bounding box as the mask. This is not ideal, because Mask R-CNN works by classifying every pixel. And by making the mask a rectangle than contains the object, we are misclassifying some of the pixels in the image. At first we were going to abandon the idea of using Mask R-CNN

because we believed that it would not work the way we were doing it. But then we found a research paper where they did it the way weer did it, and achieved reasonable results so we decided to move forward with it.

The other difficulty here is that it is unclear whether or not this is actually causing problems or not. Since we do not have images with exact masks to train on and then compare to it is very unclear whether or not the masks we are using are going to be problematic.

## 7 CONCLUSION

The goal of our project was to create a program in which you could load in an image, classify the diseases that are in the image and then draw the bounding boxes around the diseases and then print out the resulting image. This is supposed to help physicians expedite the process of diagnosing Chest X-Rays for patients and therefore allow them to increase the throughput of diagnosed images. Also this couls potentially limit the number of alternative scans such as CT scans that doctors need to acquire from their patients.

The contributions I made to this project mainly revolved around research, image file structure/retrieval and Mask R-CNN disease localization. In terms of research I read a handful of papers on topics pertaining to our project. Then for our images, I developed the file structure and implemented a Dataloader in order to have an easy way to load in the desired images sets. Then for disease localization, Mask R-CNN was used for the program in order to create bounding boxes for our X-Rays.

Through the obtained results of this, it is feasible to use these tools in our product. And with these tools, our product could be of use to physicians, doctors, or trainers that wish to increase the throughput of images they can diagnose.

Through this process I experienced a handful of issues. These included problems with Mask R-CNN and the environment I was working, but also the masks that we were using. In addition to this, I also experienced issues with the Bucknell VPN and Jupyter Lab. And lastly I had issues with waiting time and a lack of experience with the tools we are working with.

## 8 ACKNOWLEDGMENTS

Thank you to professor Stough for his guidance throughout this year long endeveor as a Professor and as our client. And also to Professor Nickel for his insight and advice along the way.

## REFERENCES

[1] 2017. Torchvision Object Detection Finetuning Tutoria. *PyTorch* (2017). https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html

[2] Sasank Chilamkurthy. 2017. Writing Custom Datasets, Dataloaders AND Transforms. *PyTorch* (2017). https://pytorch.org/tutorials/beginner/data_loading_tutorial.html

[3] Piotr Dollar Ross Girshick Kaiming He, Georgia Gkioxari. 2018. *Mask R-CNN.* Technical Report.

[4] matterport. 2018. Mask R-CNN. Mask R-CNN repository, https://github.com/matterport/Mask_RCNN.

[5] The National Institute of Health. 2018. NIH Chest X-rays. data retrieved from the National Institute of Health, https://www.kaggle.com/nih-chest-xrays/data.

[6] Kevin Widholm. 2019. Semantic Segmentation Part 3: Transfer Learning with Mask R-CNN. *Novatec* (Sept. 2019). https://www.novatec-gmbh.de/blog/semantic-segmentation-part-3-transfer-learning/

[7] Le Lu Zhiyong Lu Mohammadhadi Bagheri Ronald M. Summers Xiaosong Wang, Yifan Peng. [n.d.]. *ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases.* Technical Report.