Deep MedIA Senior Design Technical Report

NICHOLAS PASSANTINO, Bucknell University, Department of Computer Science

In this report we present a machine-learning model for disease classification and lesion detection in chest x-ray images. Our program uses neural network that performs multi-label classification, which given an input images can output confidence values regarding the prevalence of each possible disease. Our program also uses two different approaches to locate the regions of the image that contain the predicted diseases, which are gradient activation maps and the MASK R-CNN architecture. Both of which rely on information gained from our general classification neural network. All of these modeling learning techniques were trained on the NIH Chest X-Ray Image Data Set[1].

1 INTRODUCTION

With the increase in medical imaging data available, physicians must spend more time determining diagnoses. This project seeks to train a computer to analyze and quantify medical imaging data in a way that is consistent with the techniques of medical professionals. More specifically, our work uses machine learning to create a model that can identify diseases in chest X-ray images by drawing a bounding box around them.

This project is intended to assist doctors by improving the accuracy and reducing the time of diagnosis, to provide athletic trainers with more information about their athletes health when doctors are unavailable, and to expand the breadth of research into this type of problem for future researchers and developers to build on.

2 BACKGROUND RESEARCH AND RELATED WORK

The use of machine learning in medical imaging continues to expand. New ways of applying related learning techniques to problems in the medical field are developed on a continual basis, ranging from cancerous lung cell detection to interpreting mammograms[2]. In addition to this, new medical imaging data sets continue to be created by medical professionals, allowing for more possible forms of research to be conducted. Below we detail the most prominent methods that are relevant to the implementation of this project.

Convolutional Neural Network (CNN) The CNN[3] is one of the most popular modeling techniques used in image processing. The CNN computes a dot product between an input image and a small 'kernel', usually around 4 pixels in length and width, to eventually generate confidence outputs for the classification of an image. This process allows the CNN to learn features of varying sizes, in locations throughout the image. The prevalence of the features within the model informs its decision making process, affecting the output values. Since we propose a multi-label classifier, the structure of the model's outputs is an array of values ranging between 0 and 1, representing individual confidence in positive diagnosis for each of the possible diseases in the data set, which can then be interpreted as a positive or a negative diagnosis for each disease. There are many examples of successful image classification achieved using this method. Some of the most notable models are ResNet[7] and VGG[8]. Both of these models can classify the images that it intended to with over 90% accuracy.

Author's address: Nicholas Passantino, npp002@bucknell.edu, Bucknell University, Department of Computer Science, , Lewisburg, PA, 17837.

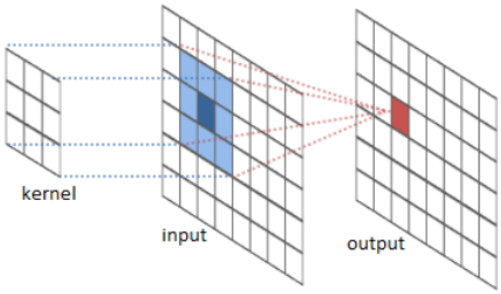

Fig. 1. The convolutional layer operation[4].

Additional Neural Network Layer Types Many neural network structures make use of convolutional layers, but those alone do not create a strong neural network. Other layer types also serve specific purposes that improve the model's accuracy and overall capabilities. Below are the most relevant components of the model proposed in this paper.

(1) **Batch Normalization** This layer normalizes the outputs of the previous layer, most commonly following a convolutional layer. Typically the normalization affects the outputs by scaling a layer's weights so that their mean equals zero and their standard deviation is one, but most machine learning libraries allow for customization of these values. By rescaling values that can potentially grow quite enormously as a machine learning model is trained, the time required to train is dramatically reduced, as smaller numbers require less computation to manipulate and evaluate.[5]

(2) **Fully Connected Layer** This layer, also known as a dense layer, transforms an input vector into an output vector of specified size. It does this by creating a weight value for each possible input and output pair, which are then multiplied across all of the inputs and summed across all of the outputs. In CNNs, this layer type is typically used for the last layers after the convolution operations are completed. Intuitively, the weights of this layer can be seen as determining how important the different features of the convolutional layers were and which features matter most for which classification labels.

(3) **ReLU** The Rectified Linear Unit (ReLU)[6] is an activation function that is most commonly placed after a convolutional layer, and after the associated batch normalization layer if that is present as well. There are many different activation functions; each of which serve the purpose of allowing a neural network to learn non-linear relationships between inputs and outputs. This particular function sets a value to zero if it is less than zero, else the value stays the same.

(4) **Sigmoid and Softmax** These two activation functions are most commonly placed at the very end of a convolutional neural network, with one particular difference. The Sigmoid function transforms all inputs into outputs between 0 and 1. This is where a model's probability outputs for classification are finally created. The Softmax function performs a similar operation, but all output probabilities for Softmax will sum to 1. Softmax is best suited for problems that have a single correct answer, whereas Sigmoid is useful when there are multiple possible answers.[6]

Gradient Activation Maps The gradient activation map[1] allows one to take a convolutional neural network and display which portions of the image contributed most to the output predictions it generated. It does this by taking the last convolutional layer and multiplying the gradient of each output times the associated weight value in the following

dense layer. By generating a heat map for a disease based on the multi-label classification model, one can then use it to create bounding boxes around the disease regions within the image.

Fig. 2. Gradient Activation map example for classifying a dog[11].

Semantic Segmentation This machine learning method takes in an image and outputs a pixel-wise classifier. This is a matrix of the same size as the original values, with each value representing a class that the corresponding pixel belongs to. In this way an image can be broken into different segments and objects can be detected. The idea behind incorporating this strategy into our project regards using the bounding box images of the NIH data set as training data, where we could use the bounding box location information to generate squares of pixel-wise labels with which we could train a model. There are many different architectures that use this technique such as the Mask R-CNN[12] architecture which is used in this project.

Fig. 3. An example output of a semantic segmentation model, assigning different classes (represented as colors) to each pixel[13].

Transfer Learning Transfer learning[14] is not a model architecture in itself, but a method of recycling successful models and adapting them to data sets different from what they were initially trained on. Models tend to learn similar pixel patterns in low level features, therefore using successful models as a starting point saves a considerable amount of processing power and time during the training period.

3 DESIGN

The design of this project is two-fold. The first component is the general architecture that maps a user's input to the outputs, informing the user of the contents of the image. The second, more specific component of the design is the architecture used for the machine learning model itself.

Project Structure The image below shows a flow diagram of the processes at work in this project. It is designed so that a person without a background in machine learning can use it, as that is one of the primary purposes of our work. A user inputs an image into the program, the program sends the image to the model, the model generates its predictions, and those predictions are interpreted to give answers back to the user. Those answers are of two kinds: the presence or lack thereof for each disease that the model is aware of, and the associated bounding box image that determines the region of the chest x-ray where the disease was detected. Bounding boxes are only displayed for diseases that are predicted to be present in the image.

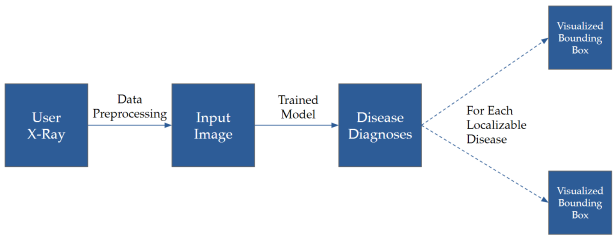

Fig. 4. The flow diagram of this project.

The Classification Model We ultimately chose to use the convolutional neural network as our classification model type. We believe that because of its enormous popularity, and large quantities of research from which to base our work on, it was the optimal choice. We additionally incorporated the idea of transfer learning by using ResNet-18 as our starting point. ResNet-18 uses 18 main convolutional layers, as its name implies, in combination with other supporting layer types to generate outputs. There are larger forms of this model such as ResNet-34, ResNet-50, and so on, but these larger models took much longer to train on the limited hardware that we had access to. We chose ResNet over other pretrained models because it performed the best in the multi-label classifier on the NIH Chest X-Ray data set described in the paper where this data set was first published[1].

Each convolutional layer in ResNet is followed by a batch normalization layer and then a rectified linear unit layer (ReLU). As was mentioned in the background information section, the batch normalization layer improves training time by normalizing the weight values, minimizing processing power and time needed to train the model. The ReLU layer teaches the model to learn non-linear patterns within the data.

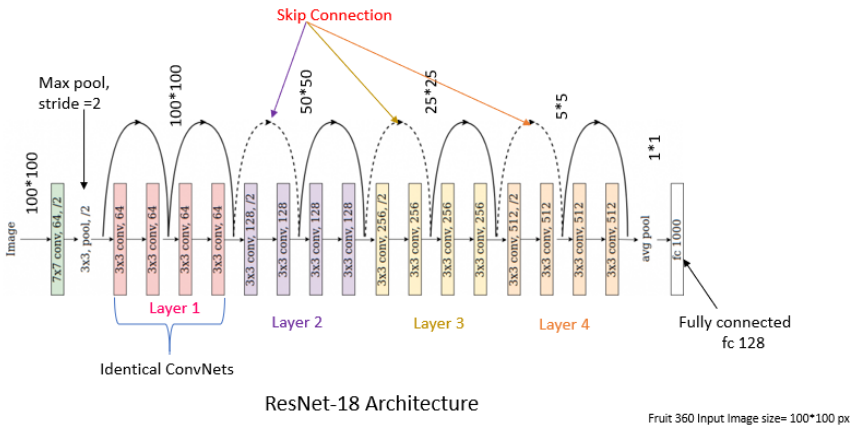

Fig. 5. The ResNet architecture we used for disease classification[7].

ResNet models initially have 1000 output classes, whereas we were only using 9 disease classes. We therefore modified ResNet so that the last two fully connected layers transform the convolutional layers output into 9 output values, each a value between 0 and 1. We initially created a binary-classifier for each disease with a single output value, but soon realized that while binary classifiers were good at detecting if a disease was present, they had no way of distinguishing if they were noticing their intended disease or a different one that looked similar. Because of this, the multi-label model was used instead. The outputs of the classifier is interpreted by using a threshold that is unique to each disease. If the value is higher than the threshold, for example if the output value was 0.8 and the threshold was 0.5, the program would determine that the disease is present. The process by which the threshold value is chosen is described in the Implementation section.

The Bounding Box Model We used two independent methods which were then compared against one another to localize diseases within the X-Ray images. Specifically, they are the gradient activation maps and semantic segmentation. A gradient activation map is generated for each disease that the multi-label classifier labels as positive within an image, which has a resolution of 32x32 pixels. This small matrix comes from the outputs of the last, smallest convolutonal layer in the classification network. This image is then up-scaled to match the resolution of the input image, which is 1024x1024 pixels. After this, the activation map is normalized between 0 and 255 to represent color values that makes visualization easier. A threshold value is chosen for which pixels are considered most relevant to the classification result, and then bounding boxes are drawn around each of the shapes within the image. To improve results, a set of heuristic functions were used to choose the best bounding box. Lastly, the bounding boxes are overlaid on the original image, and then displayed to the user of the program.

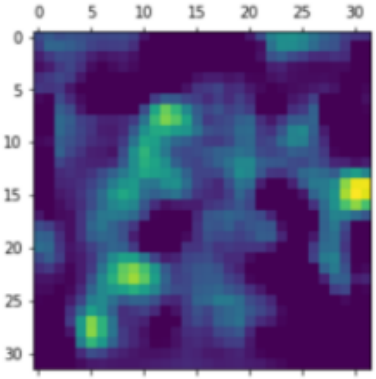

Fig. 6. Gradient activations from the last convolutional layer.

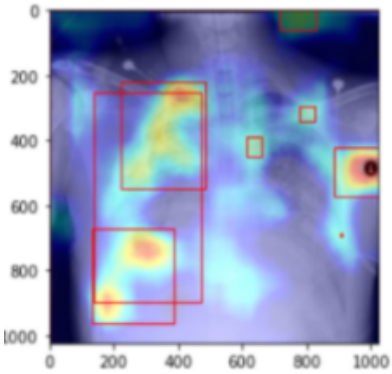

Fig. 7. Overlay of heatmap, bounding boxes, and image

In regards to semantic segmentation, our model doesn't require any information from the classification model other than which diseases it needs to be looking for. In this way, the classifier informs the segmentation model, and the segmentation model then converts inputs into output masks via its neural network structure. This output then simply has a box drawn around the segments within the image. As for the Mask R-CNN architecture, it is similar to ResNet in that it is also a residual-network, or a specific type of convolutional network that utilizes skip connections between convolutional layers, and differs from it in that its output is a layer mask that predicts what classification each pixel belongs to, instead of a classification for the image as a whole.

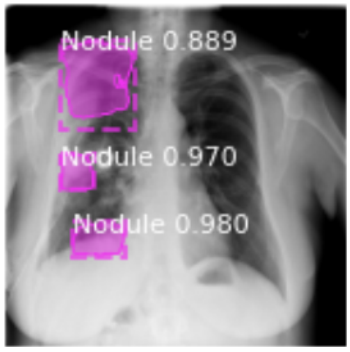

Fig. 8. Segmented chest X-Ray containing a nodule.

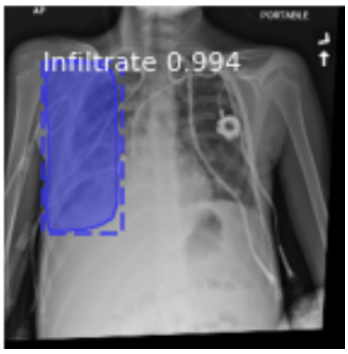

Fig. 9. Segmented chest X-Ray containing Pneumonia

4 IMPLEMENTATION

Tools The code for this project is written entirely in Python 3. The classification and gradient activation models were completed in PyTorch, while TensorFlow was used for the Mask R-CNN model. All were written and executed in Jupyter Notebooks on a shared space in the Bucknell servers. The main project file that combines the machine learning models with the user interface is an executable python file. We trained the model on the machines that Bucknell University had available for us in their Academic East building, featuring Intel(R) Core(TM) i9-9900 CPU @ 3.10GHz processors and one GeForce RTX 2080 Ti each that we could all concurrently access.

Data Preprocessing The NIH Chest X-Ray Data Set contains 112,120 images from 30,805 unique patients. Each image is labelled with a variety of data points, with the most relevant being the diseases displayed in each image. There are 9 different disease classes present, one of which is 'No Finding', representing an image without any disease present. It is estimated that the accuracy of these labels is above 90%, as they were text mined from legitimate radiological reports. Of those images, only 880 have bounding box labels, specified by x, y, width, and height values that are associated with those images. Since gradient activation maps do not require training data, all of these images are used for testing accuracy measurements of the model, whereas for MASK R-CNN, the training data was quite limited. This limitation in size should be considered in regards to the accuracy values that our model generates.

To keep the data organized, we created a nested dictionary data structure that maintained a strong level of organization among the images. The main data set object contained a dictionary with two key-value pairs. One key is 'AP' for anterior chest X-ray images, and the other is 'PA' for posterior chest X-ray images. Each key is associated with its own dictionary, where the keys are the disease types and the values are lists of the data points for each disease type. An entry in one such list is a dictionary that contains key-value pairs for the location of an image file, the diseases present, and other meta-data regarding that particular image. At runtime, our model grabs entries from one such list, uses the image file name to load in the actual image as a tensor, and perform the computations necessary for classification.

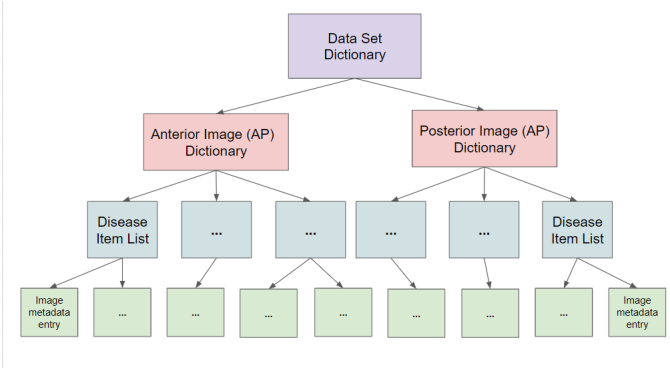

Fig. 10. The data set object structure

Training The classification model was intended to be trained on 5000 images of each disease present, as well as on an additional 5000 images with no disease present. Some diseases did not have 5000 positive case images, such as Cardiomegaly which only had 1600, though this does not seem to have impacted the accuracy results for those classes in a significant way. We used 80% of the images to run the training and backpropagation algorithm on the model, 10% of the images on testing the model's accuracy and determining the best iteration of the model to save, and 10% of the images for validating the results of the model and generating accuracy statements.

In a single training iteration, or epoch, the model grouped the images into randomized batches of size 4, computed output predictions, and then used backpropagation to alter the weights in a way that minimizes the loss calculation. If after 20 epochs the model did not improve in decreasing the test set loss, the training algorithm was terminated and the best model was saved. On average, each sequential attempt at training the model took 30 to 40 epochs over the course of about 10 hours.

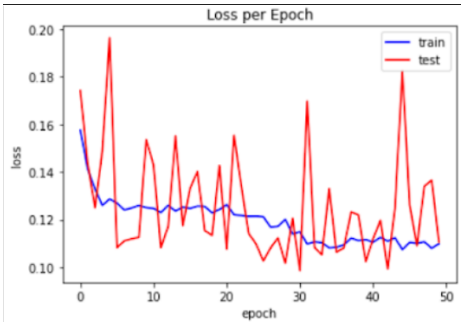

Fig. 11. Loss per epoch of training for the multi-label classifier

The implementation of the gradient activation maps involved the use of several heuristic values since there would often times be 10 or more bounding boxes drawn on an image. First, the average bounding box size for each disease was calculated using the bounding box images in the data set. Second, the 5 boxes that were closest in size to the average size for such disease is selected. Third, the box with the highest average heat map value was selected. In this way a balance between boxes that were of the wrong size but showed high gradient activation, and boxes of the correct size but showed less gradient activation, were able to be balanced against one another. The process of using these heuristics was very much trial and error based upon what would increase the intersection of the bounding box predictions with their ground truth values in the data set.

5 RESULTS

For classification, we used the area under the curve of the receiver operating characteristic as our accuracy measurement. We chose this metric because it allows us to determine the best threshold value for each model by finding the optimal balance between the true positive classification rate and the false positive classification rate. Additionally, it is the same metric used in the NIH data set's publication to evaluate the results of their multi-label classifier, allowing for direct comparison. The results are shows in the graph below, which outperform the NIH paper by varying degrees for all diseases.

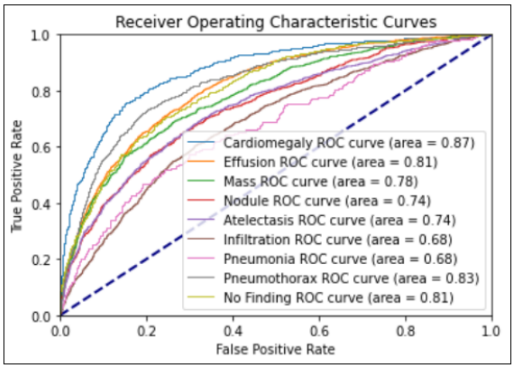

Fig. 12. Deep MedIA AUC of ROC results.

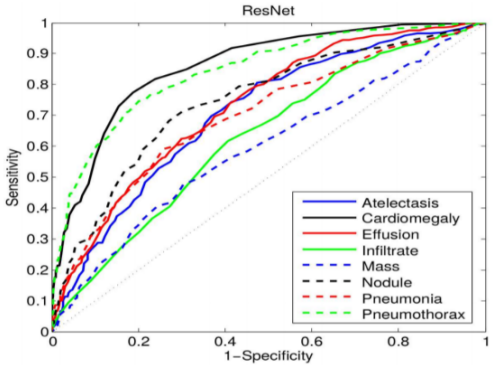

Fig. 13. NIH ROC curve results[1].

The following chart shows a side by side comparison that illustrates the success of our model. It yields anywhere from 0.03 to 0.22 higher in area under the curve of the receiver operating characteristic values than did the best performing model used by the NIH.

AUC of ROC Comparison		
Disease	**DeepMedIA**	**NIH)**
Cardiomegaly	**0.87**	0.81
Effusion	**0.81**	0.73
Mass	**0.78**	0.56
Nodule	**0.74**	0.71
Atelectasis	**0.74**	0.71
Infiltration	**0.68**	0.61
Pneumonia	**0.68**	0.63
Pneumothorax	**0.83**	0.79
No Finding	**0.81**	xx

For disease localization, we used the intersection over union (IoU) metric to determine accuracy. We used this metric to compare the percentage of overlap between the labeled bounding boxes in the NIH dataset and the bounding boxes that we generated using gradient activation maps and Mask R-CNN. The gradient activation maps technique yielded decent results for some disease classes, but Mask R-CNN outperformed overall. The boxplots below show the distribution of IoU values across the different disease classes.

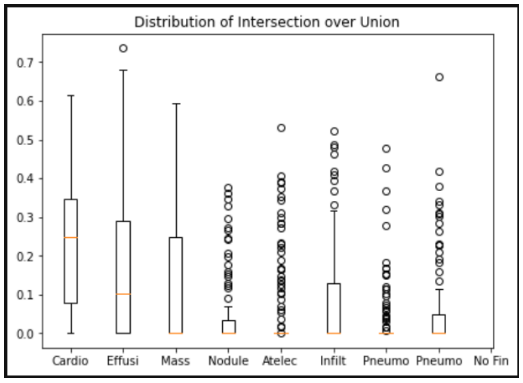

Fig. 14. Gradient Activation Map IoU values.

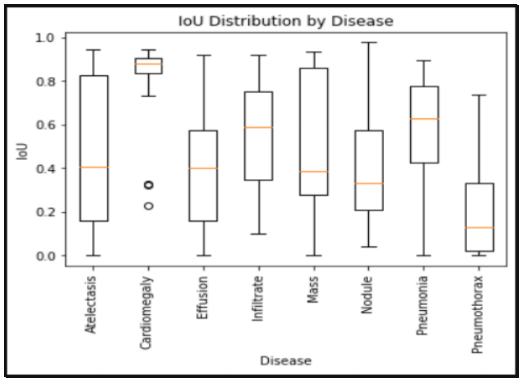

Fig. 15. Mask R-CNN IoU values.

The charts above demonstrate the greater degree of success of Mask R-CNN in comparison to the gradient activation maps. Using this as our preferred localization method, results were then formulated to match the metric used by the NIH for evaluating bounding box success. Their method, called IoBB, defines results in terms of the percentage of each disease's IoU values that are greater than some set threshold. As an example, an output value of 0.25 for a threshold of T(IoBB)=0.1 would mean that 25% of the bounding boxes for a particular disease yielded at least a IoU=0.1 value. Our results using this method are compared against those of the NIH in the table below, showing that in the majority of cases, excluding the T(IoBB)=0.9 case, the Mask R-CNN model generated better results.

Table 1. Intersection over Union of each Disease[16].

T(IoBB)	Atelec.	Cardio.	Effusion	Infil.	Mass	Nodule	Pneum.	Pneuthx.
T(IoBB) = 0.1								
Deep MedIA	**0.806**	**1.0**	**0.806**	**0.96**	**0.833**	**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	0.4277	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**

6 DEVELOPMENTAL CHALLENGES AND PROCESSES

We faced a variety of challenges throughout the completion of this project, though we worked well in overcoming each setback. Most had to do with hardware limitations that slowed our ability to make progress, while others came as the result of such problems.

Challenges Our greatest setback in the completion of this project stems from a Bucknell University service known as BisonNet. This application is meant to allow students to run programs that require large amounts of processing power on sophisticated machines by submitting 'jobs' to the program via the command line. Unfortunately for us, BisonNet does not interact well with PyTorch, and even with a batch size of 1, the remote machine would either run out of RAM or execute indefinitely, never completing any computations in our benefit. We did our best to debug the systems involved, as well as contacted the system administrators who are responsible for maintaining it, but utilizing this service properly was never achieved. We spent a great part of the fall semester trying to get this resource to work, but unfortunately were never able to. Because of this, many hours that could have been spent on researching and implementing our model were spent on debugging BisonNet instead.

```
#!/bin/bash
#SBATCH -p short # partition (queue)
#SBATCH -N 1 # (leave at 1 unless using multi-node specific code)
#SBATCH -n 1 # number of cores
#SBATCH --mem-per-cpu=8192 # memory per core
#SBATCH --job-name="myjob" # job name
#SBATCH -o slurm.%N.%j.stdout.txt # STDOUT
#SBATCH -e slurm.%N.%j.stderr.txt # STDERR
#SBATCH --mail-user=username@bucknell.edu # address to email
#SBATCH --mail-type=ALL # mail events (NONE, BEGIN, END, FAIL, ALL)
for i in {1..100000}; do
  echo $RANDOM >> SomeRandomNumbers.txt
done
sort -n SomeRandomNumbers.txt
```

Fig. 16. The BisonNet scripting instructions

In addition to the time lost pursuing BisonNet, we were now limited by the processing power of the machines outlined in the Tools section. The average run time for the multi-label classifier was 10 hours. This bottleneck reduced the amount of variations in hyper-parameters and techniques we could realistically experiment with. In addition to this, we had to limit the size of the network in terms of the number of its parameters. While the results from ResNet-18 are promising, they would possibly be higher had we used a network with more layers, such as ResNet-34 or ResNet-50[7].

Processes of Teamwork The use of the SCRUM methodology was largely successful in keeping our team on track and making effective progress toward our goals. Initially, it was challenging to designate specific tasks for each sprint, as estimating the time requirements for research was difficult to complete. Though, as our project became more defined, we were able to effectively use sprint objectives and key results in combination with SCRUM artifacts such as user stories and issue boards to make well defined, incremental progress toward our goals.

The downsides of the agile development process derived mostly from GitLab's SCRUM interface. Particularly, the website does not allow one to group tasks by user story other than adding a label to each task, which was not effective or helpful. Additionally, there is no inherent tool that can be used to generate a burn down chart, either per sprint or across the entire project's lifetime. This was particularly frustrating as we had to duplicate all of the recorded hours for completing each task into an Excel file one by one to generate a chart. In regards to our team itself, everyone actively contributed and helped in the completion of this project.

7 CONCLUSION

My Contribution My particular contributions to this project are largely defined by the descriptions given in this report, but in summation included creating the classification model, training and iteratively improving this model, generating graphs that visualized the accuracy of the model, implementing and optimizing the gradient activation maps technique, and connecting the gradient activation maps to the classifier for the alpha and beta releases. Additionally, I helped in completing the initial background research on the previous usage of our chosen data set, as well as contributed to the implementation of the command line interface for the project.

Project Overview This project was a success in its ability to train a model so that it could accurately give disease classifications for chest X-ray images, defined by the strong AUC of ROC values presented in the results section. This project was also a success in regards to the bounding box generation, which was comparable and higher in value in some situations that that of the NIH. Our choice in using a convolutional neural network with an effective architecture worked very well, though the consideration of other modeling techniques helped us in arriving at an ideal implementation strategy.

REFERENCES

[1] Wang, Xiaosong, et al. ChestX-Ray8: Hospital-Scale Chest X-Ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. , 2017.

[2] Shi, Zhenghao, et al. "Survey on Neural Networks Used for Medical Image Processing." International Journal of Computational Science, vol. 3, no. 1, 1 Feb. 2009, pp. 86–100, www.ncbi.nlm.nih.gov/pmc/articles/PMC4699299/#:~:text=Neural%20networks%20are%20well%20known. Accessed 21 Feb. 2021.

[3] Gu, Jiuxiang, et al. "Recent Advances in Convolutional Neural Networks." Pattern Recognition, vol. 77, May 2018, pp. 354–377, 10.1016/j.patcog.2017.10.013.

[4] Olah, Christopher. "Understanding Convolutions." Colah.github.io, 13 July 2014, colah.github.io/posts/2014-07-Understanding-Convolutions/. Accessed 25 Mar. 2021.

[5] Ioffe, Sergey, and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift." ArXiv.org, 2015, arxiv.org/abs/1502.03167.

[6] Chigozie, Enyinna, et al. Activation Functions: Comparison of Trends in Practice and Research for Deep Learning. , 2018.

[7] He, Kaiming, et al. Deep Residual Learning for Image Recognition. , 10 Dec. 2015.
[8] Simonyan, Karen, and Andrew Zisserman. VERY DEEP CONVOLUTIONAL NETWORKS for LARGE-SCALE IMAGE RECOGNITION. , 2015.
[9] Goodfellow, Ian, et al. Generative Adversarial Nets. , 2014.
[10] Packt_Pub. "Inside the Generative Adversarial Networks (GAN) Architecture." Medium, 19 Nov. 2019, medium.com/@Packt_Pub/inside-the-generative-adversarial-networks-gan-architecture-2435afbd6b3b. Accessed 25 Mar. 2021.
[11] Rosebrock, Adrian. "Grad-CAM: Visualize Class Activation Maps with Keras, TensorFlow, and Deep Learning." PyImageSearch, 9 Mar. 2020, www.pyimagesearch.com/2020/03/09/grad-cam-visualize-class-activation-maps-with-keras-tensorflow-and-deep-learning/. Accessed 25 Mar. 2021.
[12] "Mask R-CNN." IEEE Transactions on Pattern Analysis and Machine Intelligence, 2018, pp. 1–1, 10.1109/tpami.2018.2844175.
[13] Karagiannakos, Sergios. "Semantic Segmentation in the Era of Neural Networks." AI Summer, 25 Jan. 2019, theaisummer.com/Semantic_Segmentation/. Accessed 10 May 2021.
[14] Tan, Chuanqi, et al. A Survey on Deep Transfer Learning. , 2018.
[15] [n.d] Abstract (summary). Retrieved Feb 13, 2020 from https://en.wikipedia.org/wiki/Abstract_(summary)
[16] [n.d] Tables Generator. Retrieved Feb 13, 2020 from https://www.tablesgenerator.com/
[17] Gary Blake and Robert W. Bly. 1993. The Elements of Technical Writing. Macmillan Publishers, New York.