RSNA Pneumonia Detection Challenge

Suhird Singh, Adit Shah, Jagdeep Mann, Pratik Sanghvi
School of Computer Science
University of Windsor

{singh1nt, sanghvip,shah19a,mann12h}@uwindsor.ca

***Abstract*--** In the challenge named RSNA Pneumonia Detection Challenge, presented by Kaggle.com, we are trying to build an algorithm to detect visual cues for pneumonia in medical images specifically termed as CRX. By presenting this challenge RSNA community are thinking to leverage the Machine Learning tools to improve their initial diagnosis and provide better services. Based on the data provided we need to predict whether the patient has pneumonia or not. Pneumonia is an inflammatory condition of the lung affecting primarily the small air sacs known as alveoli. [1].

Diagnosing pneumonia is a difficult task and requires specific radiographs known as chest radiographs (CXR). These kinds of radiographs are taken by a highly trained specialist. Also, it is difficult to interpret the CRX images as a number of conditions obscures most the pertinent area in the images. [2]. One thing that can be inferred that comparison of multiple CRX images of the patient can be done thereby leading to detection of many other abnormalities.

By presenting our solution we are trying to provide early insight so as to improve the diagnosis of pneumonia in the affected patients. Our take on this problem involves Mask RCNN which is very helpful in object detections and Object Segmentation. These techniques will help to narrow down our search for whitish patches inside the radiographs. We aim to present some interesting insights within the dataset through our solution.

1. INTRODUCTION

Kaggle is an online community comprised of data scientists, Machine learning engineers, and many other professionals. This online community is owned by Alphabet Inc with its parent as Google. The community provides a platform for hosting challenges, publishing datasets, providing an online workbench for data science and crash courses on AI. Kaggle initially hosted only Machine Learning competitions but currently, it is providing many other services like a public platform for hosting datasets, job postings through careers and many more as listed before. As a whole Kaggle as an online data science platform has fostered advancement in many fields ranging from transport, politics, medical research, physics and many more to come. Through Kaggle competitions participants are compelled to improve beyond the existing practices thereby providing new and efficient methods to solve the problems. Many of the findings found during the competitions are turned into paper and many other have taken the form of research questions. [3]

In the modern world, the field of Image processing and Natural Language processing has advanced to very great heights. Detecting objects within images and in video clippings is now an easy task, thanks to the current development and the availability of powerful resources. Many fields like Computer Vision, Artificial Intelligence, Self-driving cars have grown exponentially thanks to the advancements in the existing methods and an increase in the availability of the resources.

Certain specialized methods are required for identifying the objects within the photos which were extant even in the 1900s. But why these ideas are now gaining much traction are compared to earlier years. One of the main reasons, which is already pointed out, is the current technological advancements in the computation field and also the availability of huge data required to perform experiments and provide insightful findings through existing methods. Dozens of methods are listed for the activity of object detection which is classified into Machine Learning methods or Deep Learning methods.

Machine Learning methods are [4] :
1.Viola-Jones object detection framework based on Haar features
2.Scale-invariant feature transform (SIFT)
3.Histogram of oriented gradients (HOG) features

Following are the Deep Learning techniques used for object detection [4]:
1.Region Proposals (R-CNN, Fast R-CNN, Faster R-CNN)
2.Single Shot MultiBox Detector (SSD)
3. You Only Look Once (YOLO)

For this paper, we are using Mask RCNN, a technique which is an extension of the Faster RCNN Deep Learning technique.

Most of the time we are trying to identify the objects within the images which can be any person, animal or any objects for the given instance. But understanding the semantics of the given image and identifying the classes of the objects present on the level of each pixels is a daunting task but it can be achieved through Semantic Segmentation. [5]. The Semantic Segmentation technique is very helpful for fields like Robotics, Computer Vision, Self-driving cars, where it is necessary to understand the

semantics of the environment in which the action is performed so as to achieve the desired outcome. These techniques automate the tasks at hand thereby opening many opportunities for the development of many solutions.

Currently, deep learning techniques are providing the much greater result as compared to the other methodologies. In case of object detection too techniques like Fast RCNN, YOLO, Faster RCNN are providing great results but as compared to the listed methods there is one other existing method which provides great results for pixel level detection as compared to others, which is Mask RCNN. It is an extension of the Faster RCNN, which uses a different branch or a fully connected network to predict the regions in comparison to Faster RCNN which uses a region proposal network to generate proposals for objects inside the images.

The Mask RCNN provides great results as compared to Faster RCNN because it extends the same techniques to pixel level segmentation which speeds up the process. Also, the techniques can be applied to many general tasks. Mask RCNN extends Faster RCNN by just adding a branch of fully convolutional network parallel to the existing execution flow of Faster RCNN, which identifies the bounding boxes. It adds only a small overhead to the existing methods which are 5fps. [6]. Mask RCNN can also be applied to the problem of human pose estimation and detection of many other general actions within the image.

2. RELATED WORKS

There are numerous ways to solve the problem of object detection may it be using machine learning of using deep learning. But in the current scenario employing deep learning techniques to solve the problem is the best solution.

Following are the related works which are used as a base to establish the context for Mask RCNN and it also includes some other image classifier model.

A. Resnet

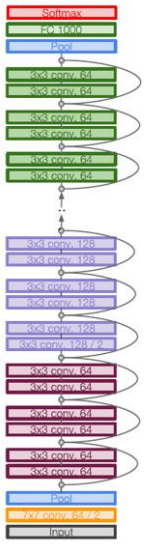

Figure 1. The gives figure shows the high-level Resnet Architecture [17][18]

Resnet is an image classification model which was an ILSVRC'15 winner with 152 layers for the ImageNet dataset. It swept all the competition under 2 well-known competition which is ILSVRC and COCO in 2015. The Resnet uses a residual network which employs skip connections to directly skip one layer. Why this architecture works because initially, the neural network runs on a few layers and then it expands into many layers when it learns more feature space. As the network expands it will explore more feature space and the residual network will help the main architecture to not get swayed away by some discrepancies in the data.

A. RCNN

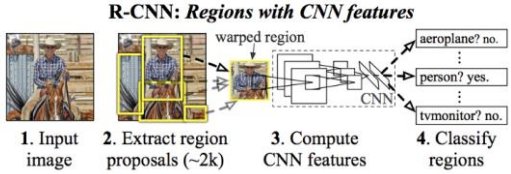

Figure 2. This figure shows a high-level overview of Typical RCNN Architecture with Region proposals and the classification at the end of the network [9]

Initially, it was difficult to devise an algorithm using conventional methods as there would be numerous regions of interest within the image. To overcome this problem we use RCNN with region proposal methods. The aim of RCNN is to create the Region Bounding Boxes using region proposal methods namely Selective Search. The selective search uses different window sizes and other characteristics to

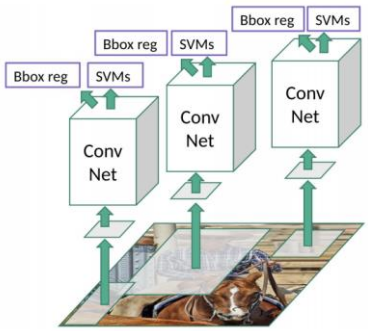

Figure 3. This figure shows an overview of RCNN architecture with Bbox reg and SVMs [9]

find the bounding boxes. The cropped images of these bounding boxes are passed through a pre-trained Convolutional neural network (CNN) to identify the objects and assign labels which include Support Vector Machine (SVM). [7]. Although this approach provides a solution to the object detection problem, it is very slow and inefficient. The region proposal method generates a huge amount of proposal which needs to be classified. The region proposal activity hence adds time to the overall training time of the model. Also, the selective search algorithm is fixed thereby there is no learning happening here. We need some other method to optimize the RCNN approach or provide some other better approach.

B. Fast RCNN

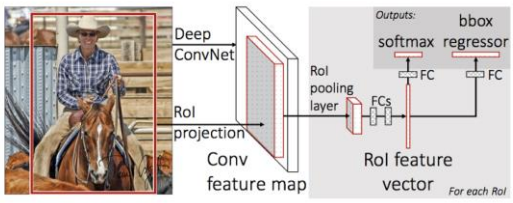

Figure 4. The given figure shows High level overview of Fast RCNN Architecture [9]

In order to solve the problems of RCNN, Fast RCNN was proposed. In RCNN the proposed regions over an image were feed to the Convolutional Network, which was the overhead during training. This particular issue is solved by Fast RCNN by first generating convolutional feature map [9] from which the Region of proposals are identified. Now we have the ROI's but there might be cases where the boxes might not be fitted properly over the object. In order to reshape the object, the ROI's are passed through the ROI pooling layer. The feature vector [9] obtained from the ROI pooling layer is passed to the softmax layer for predicting the classes and generating the update values for the bounding boxes. When Fast RCNN was compared with RCNN during training time, it proved to be quicker. But it was observed that Fast RCNN with region proposal takes much time as compared to the time taken during testing without the region proposal network. We can infer that the region proposal [9] method becomes the point of contention here.

C. Faster RCNN

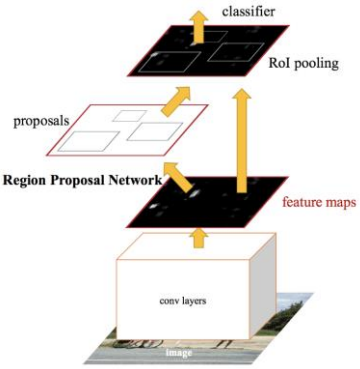

Figure 5. The following figure shows the Faster RCNN model overview [9]

In the case of Fast RCNN the bottleneck was the region proposal methodology. To eliminate there is another model known as Faster RCNN. Faster RCNN starts in the same way as Fast RCNN with Convolutional neural network and generate a feature map. But in the next step, the feature map is not provided as an input to the third-party region proposal method instead it is passed to a separate network which proposes the regions. These predicted regions from the network are passed to the ROI pooling layer for reshaping and then to the classification layer for predicting the label for the regions and the update values for the bounding boxes.

D. YOLO

You only look once (**YOLO**) is an object detection algorithm with is much efficient as compared to the existing algorithms. When compared to another algorithm, YOLO does do not use third-party region proposal algorithm but it uses single convolutional network working on the whole image instead of feature maps. The first step in YOLO is to divide the image into an SxS grid. We also take m bounding boxes per grid. Then for every bounding box, the network outputs a probability class and offset coordinates for the boxes. Only those bounding boxes are selected which has a class probability above the given threshold.

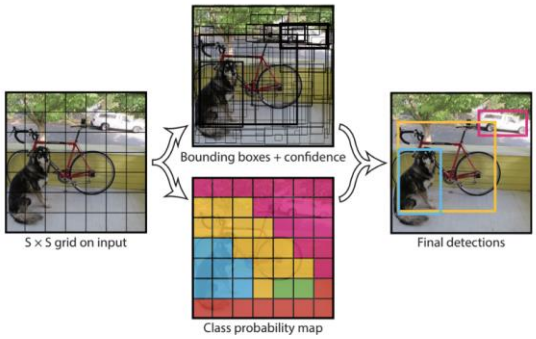

Figure 6. The following figure shows You Only Look once **(YOLO)** abstract ideology. [9]

In contrast to other object detection algorithm, YOLO works comparatively faster when the objects to be detected are not small objects. YOLO struggles when the objects to be detected are very small, as compared to other algorithms.

E. Mask RCNN

The first few techniques listed under the heading related works were used for object detection but when it comes to segmentation, which means identifying the pixels responsible for the classification of the objects, we need to look for some other algorithm. Mask RCNN, an algorithm which does segmentation at the pixel level. Along with predicting the labels and the bounding boxes, it also does image segmentation. It does this by extending the existing algorithm of Faster RCNN by adding a branch which generates a binary mask predicting whether the gives pixel contributes to the given part of an object or not.

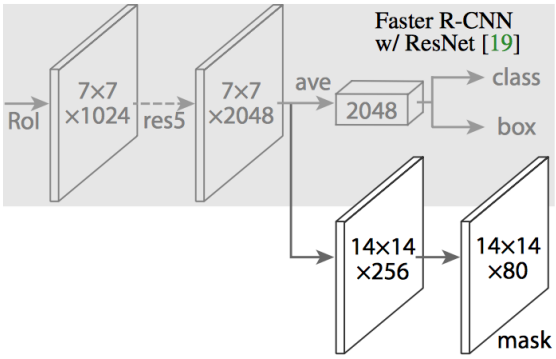

Figure 7. The following figure shows Mask RCNN extending Faster RCNN model for a generation of a mask (segmentation output) [7][8]

Later there were some inaccuracies observed during prediction which were tackled using ROIalign [10]. It uses bilinear interpolation [10] to avoid the misalignment caused by ROI Pool. At the last step, these masks are merged with the predictions

generated through the traditional network to generate image segmentation with object detection.

3. IMPLEMENTATION

For solving the problem of the RSNA Pneumonia challenge on Kaggle.com we had used certain libraries and other online platforms for running the computation and recording the results.

We are using Resnet 50 as a backbone for classification of the images. We are particularly using Resnet 101 as it is employed for building deeper networks. Next, we get the feature maps on which the Faster RCNN steps are performed. Initially, we run the feature maps through the Selective Search method which gives us the Regions of Interest boxes. These are then passed to the Linear Classifier and in Parallel, we also create a Mask using Mask RCNN.

At last, we get the predictions which contain the bounding boxes and also the probability scores on the images.

Furthermore, we will specify the technical details of the implementation.
For the computation of the results and gathering predictions, instead of running locally we had used google colab. Colab is a cloud environment and provides a workbench for running the Python 2 or Python 3 kernels.
Colab also was known as Collaboratory was started as a project of Jupyter and then it was taken over by Google. As of the current year, Google's collab does not support kernels from Julia and R. The collab environment provide 12 hrs free GPU runtime per session which is very good to get things going as compared to other environments where the plans are on pay per usage method.

Initially, we import the keras with tensorflow as backend and other dependent libraries. Next to fetch the dataset of Kaggle we are using Kaggle API which requires a custom API key to be generated for that particular user. In total there are 26684 images for the training of which 2669 are included in the validation set. For prediction, we have a total of 3000 images inside the dataset.
Also, one other library is the Mask RCNN library from git hub, which provides the prediction for the images use.

A. Image Preprocessing

Before starting the whole process of prediction, we need to make sure that the data which is being used is consistent so that during the computation the results obtained does not have any discrepancy. The size of the original image is 1024x1024 which was reduced to 256x256 in case of image preprocessing. The given Dicom images are on the gray scale which we are transferring it to the RGB scale in this step.

B. Image Augmentation

In the case of our dataset, there is only some variation in the images. But if we want variation within the data set, we need to augment the images which can be done using various operations like scaling, rotating, sharing and flipping the image. In the model specified for solving the problem mostly all the listed techniques are used for augmenting the image data. As Keras is used for processing, there are some specific parameters for image augmentation which can be set up.

C. Batch Size

For solving the problem, we have to send the images in batches as the model architecture is unable to handle them. So, we had divided the batches each into the size of 32 images at a time.

D. Parameters

There are certain parameters whose configuration is set initially to run the model. These are kept configurable as they can change over the time based on the output of the model.

Some particular parameters which are kept configurable are a number of images per GPU, backbone which specifies that whether we are using Resnet or VGG net, dimensions of the images.

E. HyperParameters

They are the parameters which need to be learned by running the algorithm and then setting them. Some of the hyperparameters are learning rate which is 1e3, momentum is set to 0.9, a number of epochs are set to 30 where each epoch contains 100 steps.

F. Training Model

For training the model, there are total 26684 label image data. During prediction, we are taking 2 classes into consideration which are No Lung Opacity and Lung Opacity. In our network, there is a total of 235 layers including convolutional 2D, Batch Normalization Layer and activation layer(relu), classifier as soft max and fully connected layer size is 1024.

While training the selection criteria used for finalizing hyperparameter values was a random search which involves moving the different values of the parameter and using those values which works. This approach has proved to be a good option in some cases providing meaningful insights. For the optimizer being used is SGD and the regularization being used is L2. The initial image classification network which we are using is Resnet 101 and next,

we are using the Faster RCNN with Mask RCNN for region classification and prediction of the bbox.

The configuration of the hyperparameters is used as specified in the previous section. During training, there are multiple losses namely the rpn loss, mask rcnn loss, bbox loss, validation loss, training loss. The total training time over 30 epochs with each of 100 steps, is 4 hours where the model was run on the google colab environment using GPU. When doing multi-class classification cross entropy error is used to calculate the loss between the true label and the predicted label. In our model which uses Keras for the implementation and tensorflow as backend has support for the Categorical Cross Entropy. But in our implementation, we are using another variant of the Categorial entropy which is Sparse Categorical Cross Entropy.s

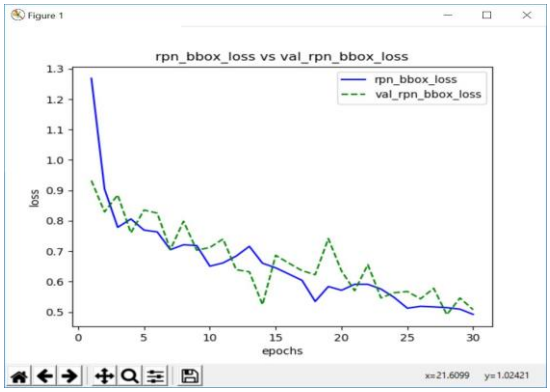

Figure 7. Following image shows a line graph of RPN Train Loss vs RPN Val loss

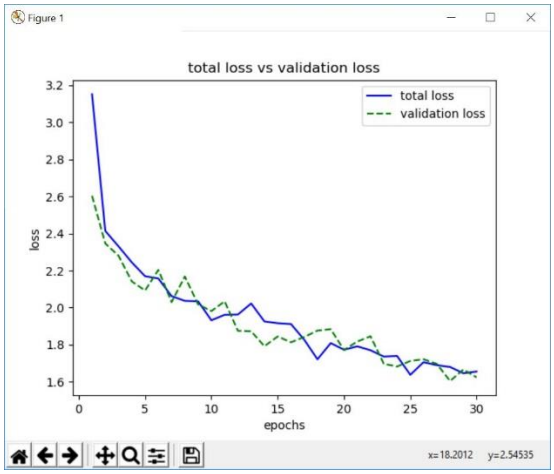

Figure 8. Following image shows a line graph of Total Loss vs Validation loss on the model.

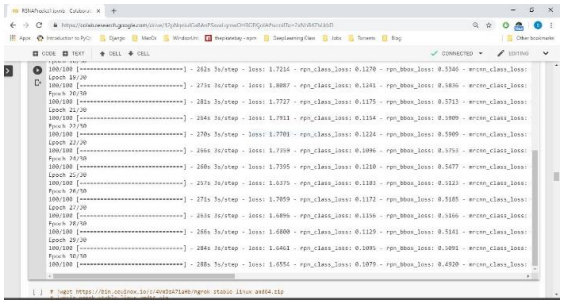

Figure 9. The given image shows snapshot od last few Epoch on the models.

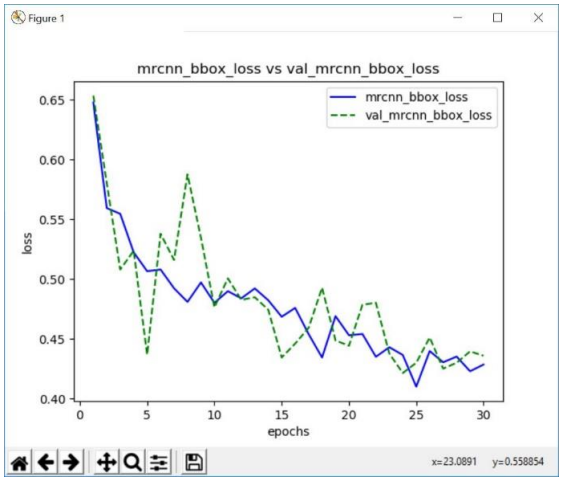

Figure 10. The given image shows MRCNN train and validation loss for the given model.

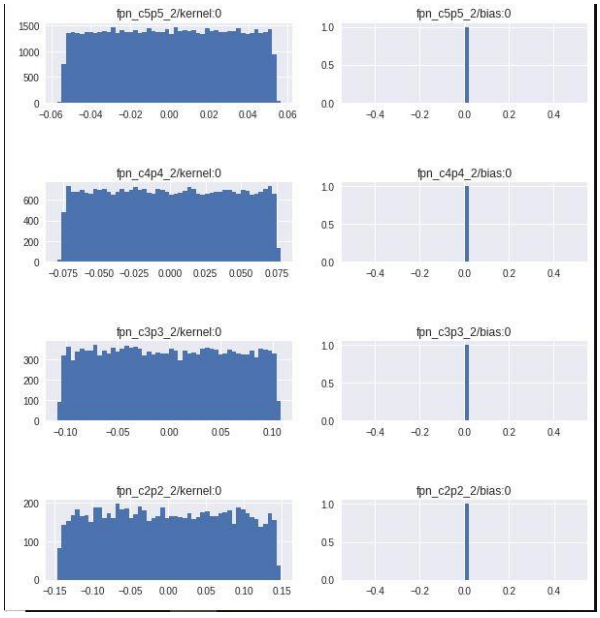

Figure 11. The following figure shows the weight distribution at particular layers

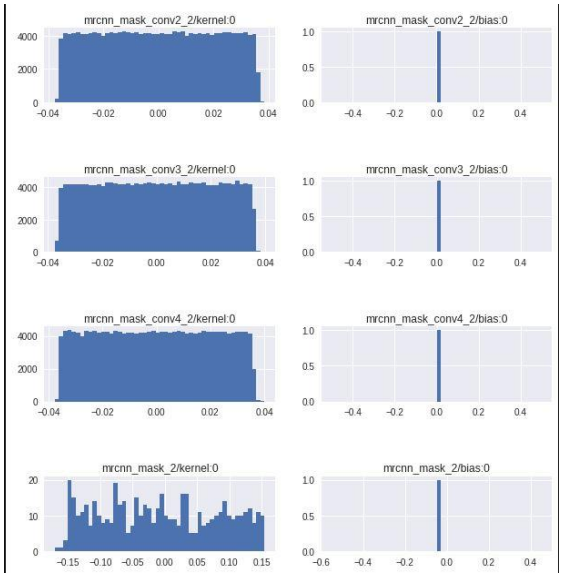

Figure 12. The following figure shows the weight distribution at particular layers.

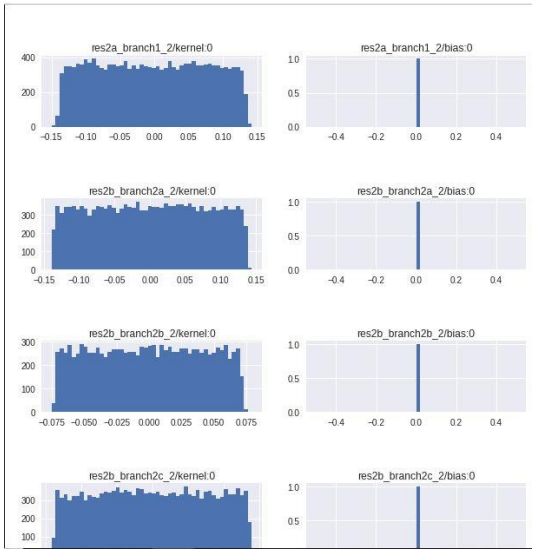

Figure 13. The following figure shows the weight distribution at particular layers.

G. Prediction

For predicting the bounding boxes and the labels on the images we had run a total of 3000 through the model which took around approximately 8 minutes on the google's colab environment. If in certain cases there are no lung opacities as compared to the ground truth classes, we then we display no instances included.

Further, we have included the images generated during the prediction phase which includes object localization and image segmentation with specific class probabilities.

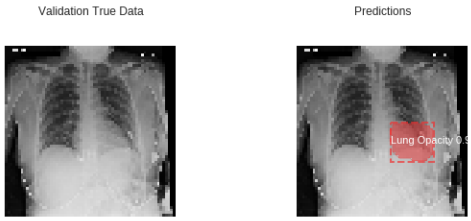

Figure 14. Prediction result sample 1

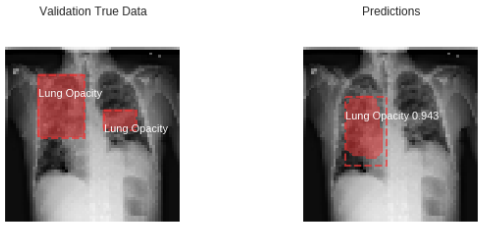

Figure 15. Prediction result sample 2

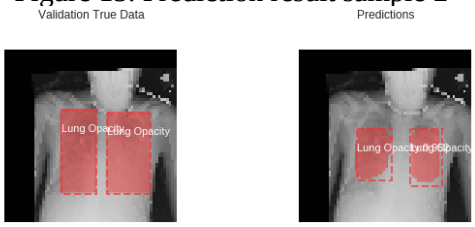

Figure 16. Prediction result sample 3

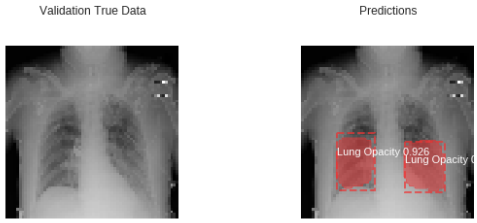

Figure 17. Prediction result sample 4

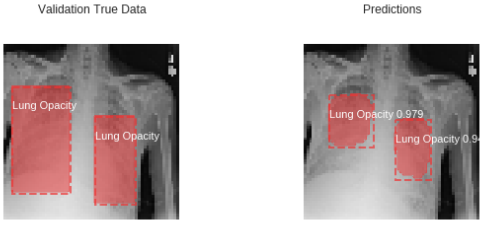

Figure 18. Prediction result sample 5

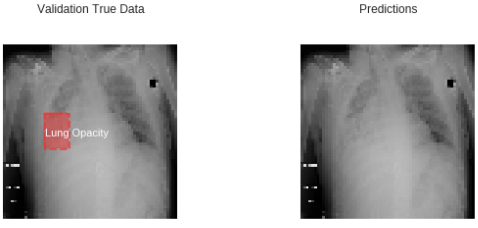

Figure 19. Prediction result sample 6

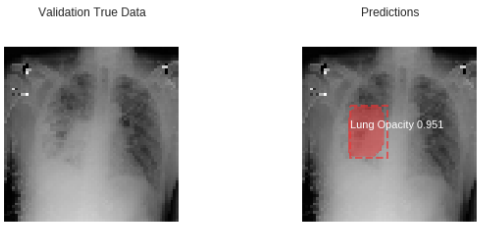

Figure 20. Prediction result sample 7

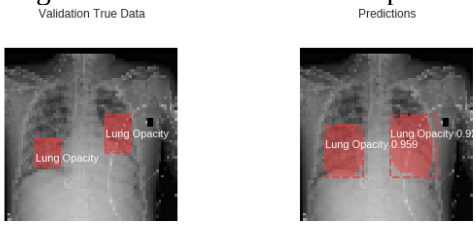

Figure 21. Prediction result sample 8

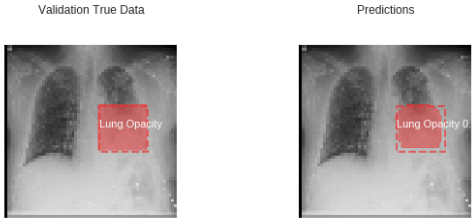

Figure 22. Prediction result sample 9

4. Research questions and methodology

While solving the problem and predicting labels and bounding boxes on the RSNA dataset we observed certain things which can be researched further.
Following are the questions which we encountered and researched it during our implementation.

A. Why are we using mask RCNN and Resnet 101 as a backbone for object detection and segmentation?

Mask RCNN is an algorithm which does segmentation at the pixel level. Along with predicting the labels and the bounding boxes, it also does image segmentation. It does this by extending the existing algorithm of Faster RCNN by adding a branch which generates a binary mask predicting whether the given pixel contributes to the given part of the object or not. Our Mask RCNN model does pixel-wise segmentation along with object detection by drawing bounding boxes around the pneumonia prone area in lungs. This is better than Faster RCNN because we get more accurate pixel-wise sematic segmentation for the region of pneumonia inside the rectangular bounding box. For example, refer figure 14 to see the pixel level segmentation inside the box.

Our model uses Mask RCNN with a backbone of Resnet 101 as the Resnet is an efficient image classification network with a total of 235 layers thereby going deep and finding greater insights into the data under consideration.

In Alex net and VGG 16, no of layers are very less in the architecture, memory consumption is also high as a number of parameters increases gradually while training.

GoogLeNet has about 22 layers. Also, it covers the drawback of VGG and AlexNet by using the inception module in its architecture. But this architecture is computationally expensive due to multiple convolution layers in each inception module. Resnet is used for deeper networks using residual connections. For deeper networks, it is very reliable as it improves the efficiency and minimizes the loss over data while training.

B. What are the changes in efficiency and loss when we used Resnet 101 instead of Resnet 50 and changed parameters/ hyper-parameters?

Initially, we trained our model using Resnet 50 with the 132 layers including (Mask RCNN layers) and following configuration for the hyperparameters:

i. *Epochs = 1*

ii. *Training Images = 26684*

iii *Testing Images = 3000*

iv *Batch Size = 8*

v. *Image Resize = 64x64*

Using the above-mentioned configuration for parameters/hyperparameters, the total loss after one epoch was 3.9.
Time estimation for training was ~15 mins (with a huge loss). Refer figure 14 and 20.
To minimize the loss, we change the backbone network to Resnet 101 and also the configuration of some of the parameters/hyperparameters as mentioned below:
i. *Epochs = 30*

ii. *Training Images = 26684*

iii. *Testing Images = 3000*

iv. *Batch Size = 32*

v. *Image Resize = 256x256*

Using the above-mentioned configuration for parameters/hyperparameters, the total loss after 30 epochs was 1.6. Refer to Figure 9 for more details.
Time incurred for training was ~4 hours (with the reduced loss as compared to Resnet 50).

5.Summary

The aim of our project by using deep learning for Vision is to provide observations on the dataset under study which is provided by RSNA and hosted by Kaggle on Kaggle.com. We download the dataset from Kaggle, referred MD.ai for model creation and Mask RCNN for creating the masks. We also used MatterPot's mask RCNN library from git hub for pixel level segmentation. Our backend python libraries is Keras and TensorFlow.

The code for training and testing is executed on Google Cloud (Colab.Research.Google notebook) with 1 GPU. The initial sample code we referred from MD.ai was running with Resnet 50 and 1 epoch which resulted in a huge loss for the model. We added code for image augmentation and changed the hyperparameters from Resnet 50 to Resnet 101 as the backbone. Moreover, we changed the batch size, a number of epochs, image resize dimensions to reduce the loss. The code was also added to visualize the loss on graphs, weight distribution using histogram for each layer and x-ray images for ground truth vs prediction. The model training was executed on google cloud with 1 GPU taking around 4 hours. The test data predictions were added to an excel file with 3000 entries for Patient Ids along with respecting bounding box coordinates and pneumonia classification.

6. CONCLUSION

The loss was significantly reduced by using ResNet 101 and by changing some parameters/hyperparameters. The epochs were also increased to 30. Using activations function like Softmax and Relu we get better results for our problem. This model reduces the time for initial diagnosis of pneumonia in affected patients.

For future improvements, the accuracy/precision can be increased by improving network architecture and backbone.
This can be achieved by adding more layers or changing the backbone network entirely. The precision can also be increased by adding more training images. This can be achieved by improving our image augmentation parameters. For increasing the time efficiency we can add more computational power by adding more GPUs.

REFERENCES

[1] https://en.wikipedia.org/wiki/Pneumonia

[2] https://www.kaggle.com/c/rsna-pneumonia-detection-challenge

[3] https://en.wikipedia.org/wiki/Kaggle

[4] https://en.wikipedia.org/wiki/Object_detection

[5] http://www.cs.toronto.edu/~tingwuwang/semantic_segmentation.pdf

[6] https://ieeexplore.ieee.org/document/8237584

[7] https://blog.athelas.com/a-brief-history-of-cnns-in-image-segmentation-from-r-cnn-to-mask-r-cnn-34ea83205de4

[8] https://arxiv.org/pdf/1703.06870.pdf

[9] https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e

[10] https://medium.com/@umerfarooq_26378/from-r-cnn-to-mask-r-cnn-d6367b196cfd

[11]https://en.wikipedia.org/wiki/Project_Jupyter#Colaboratory

[12] Rui P, Kang K. National Ambulatory Medical Care Survey: 2015 Emergency Department Summary Tables. Table 27. Available from: www.cdc.gov/nchs/data/nhamcs/web_tables/2015_ed_web_tables.pdf

[13] Deaths: Final Data for 2015. Supplemental Tables. Tables I-21, I-2 Available from: www.cdc.gov/nchs/data/nvsr/nvsr66/nvsr66_06_tables.p

[14] Franquet T. Imaging of community-acquired pneumonia. J Thorac Imaging 2018 (epub ahead of print). PMID 30036297

[15] Kelly B. The Chest Radiograph. Ulster Med J 2012;81(3):143-148

[16] Wang X, Peng Y, Lu L, Lu Z, Bagheri M, Summers RM. ChestX-ray8: Hospital-scale Chest X-ray Database and Benchmarks on Weakly-Supervised Classification and Localization of Common Thorax Diseases. IEEE CVPR 2017, http://openaccess.thecvf.com/content_cvpr_2017/papers/Wang_ChestX-ray8_Hospital-Scale_Chest_CVPR_2017_paper.pdf

[17]https://en.wikipedia.org/wiki/Residual_neural_network

[18] http://cs231n.stanford.edu/slides/2017/cs231n_2017_lecture9.pdf

[19] http://colab.research.google.com/

[20] https://github.com/matterport/Mask_RCNN

[21] https://cwiki.apache.org/confluence/display/MXNET/Multi-hot+Sparse+Categorical+Cross-entropy

[22] https://towardsdatascience.com/an-overview-of-resnet-and-its-variants-5281e2f56035