# RSNA Pneumonia Detection

Osaetin Evbuoma

*Machine Learning Nanodegree – Udacity, 2018*

**Abstract**

In this project, I will attempt to build a machine learning model using deep learning that will to detect cases of pneumonia (lung opacity) infections in chest radiographs. The model will leverage state-of-the-art object detection and instance segmentation algorithms that will help accurately detect the ailment.

# 1. Definition

## 1.1 Project Overview

Pneumonia is an infection that inflames the air sacs of one or both lungs and fills it with fluid or pus (purulent material), causing cough with phlegm or pus, fever, chills and difficulty in breathing. It is the leading infectious cause of death internationally for children under 5, killing approximately 2,400 children a day. Pneumonia accounts for approximately 16% of the 5.6 million under-five deaths, killing around 880,000 in 2016 and around 900,000 children in 2015. Most of these victims are under 2 years old.

In the USA, pneumonia accounts for over 500,000 emergency ward visits with over 50,000 deaths in 2015, keeping the ailment among the top 10 deaths in the country. In Sub-Saharan Africa, over 500,000 infants died of the ailment in 2015 and the region accounts for roughly half of pneumonia deaths worldwide. It is imperative by these ridiculously high numbers that pneumonia is a problem that need to be tackled as early and quickly as possible.

This project originates from the Radiology Society of North America (RSNA). The RSNA is an international society of radiologists, medical physicists and other medical professionals with more than 54,000 members from 146 countries across the globe. They see the potential for ML to automate initial detection (imaging screening) of potential pneumonia cases to prioritize and expedite their review.

The dataset is made up of chest radiographs in the form of DICOM images and are separated into testing and training sets. The dataset also contains two CSV files: one contains the identifications of patients (of the DICOM files), the coordinates of the bounding boxes for those patients with pneumonia and a target column indicating if a patient has pneumonia or not. The second CSV file contains detailed information that categorizes the DICOM images into three categories: *No Lung Opacity/Normal, Normal, and Lung Opacity*. Only “Lung Opacity” category indicates that a patient has pneumonia. A patient’s chest radiograph can contain 0 or more cases of lung opacity. Each case is represented by a row in the CSV files.

## 1.2 Problem Statement

Properly diagnosing pneumonia can be a tall order because it requires the review of chest radiographs (CXR) by highly trained specialists. The specialist confirms a case of pneumonia by also examining the patient's clinical history, vital signs and laboratory examination results. Pneumonia usually reveals itself in the lungs as an area(s) of increased opacity on CXR, however, diagnosis on CXR can be complicated by several other lung conditions. These include fluid overload (pulmonary edema), bleeding, volume loss (atelectasis), lung cancer or post-radiation/surgical changes. Outside of the lungs, fluid in the pleural space (pleural effusion) also appears as increased opacity on CXR.

CXRs are the most commonly performed diagnostic imaging study. Several factors such as positioning of the patient and depth of inspiration can alter the appearance of the CXR, complicating interpretation further. In addition, clinicians are faced with reading high volumes of images every shift.

The project will attempt to develop a model that will detect visual signals of pneumonia in medical images. It will automatically locate lung opacities on CXRs.

The model will be built using a deep learning (computer vision) algorithm known as Mask Region-based Convolutional Neural Network (Mask R-CNN). Mask R-CNN is an instance segmentation algorithm that allows in the identification of pixel-wise locations of a desired class (lung opacity). This means segmenting individual cases of pneumonia in a CXR.

To use Mask R-CNN, each patient's radiograph will need to be parsed, that is, instead of having multiple rows for a single patient, each patient will be converted into an object with information such as the image location and coordinates of the bounding boxes (that is, annotations, for those that have lung opacities). The parsed dataset will be split into a training and validation set which will be fed into the model to learn. Inferences will be made on the final trained model, comparing the model's performance (detection/inference) on the validation set against the actual targets of the validation set.

## 1.3 Metrics

The solution model will be evaluated on the mean Average Precision (mAP) at different Intersection over Union (IoU) thresholds. IoU is a measure of the overlap between two bounding boxes containing objects. It calculates the size of the overlap between the two objects, divided by the total area of the objects combined.

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

*A: predicted bounding box (object); B: ground truth box (object)*

The metric sweeps over a range of IoU thresholds, at each point calculating an average precision value. The threshold values range from 0.4 to 0.75 with a step size of 0.05: *(0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75)*. In other words, at a threshold of 0.5, a predicted object is considered a "*hit*" if its IoU with a ground truth object is greater than 0.5.

At each threshold value *t*, a precision value is calculated based on the number of true positives (TP), false negatives (FN), and false positives (FP) resulting from comparing the predicted object to all ground truth objects:

$$\frac{TP(t)}{TP(t) + FN(t) + FP(t)}$$

A true positive is counted when a single predicted object matches a ground truth object with an IoU above the threshold. A false positive indicates a predicted object had no associated ground truth object. A false negative indicates a ground truth object had no associated predicted object. If there are no ground truth objects at all for a given image, any number of predictions (FPs) will result in the image receiving a score of zero and being included in the mAP.

The average precision of a single image is calculated as the mean of the above precision values at each IoU threshold:

$$\frac{1}{|thresholds|}\sum_{t}\frac{TP(t)}{TP(t) + FN(t) + FP(t)}$$

Finally, the score returned by the evaluation metric is the mean taken over the individual average precisions of each image in the test dataset.

$$\frac{1}{n}\sum\left(\frac{1}{|thresholds|}\sum_{t}\frac{TP(t)}{TP(t) + FN(t) + FP(t)}\right)$$

*n: total images predicted (test images)*

# 2. Analysis

## 2.1 Data Exploration & Exploratory Visualization

The dataset is made up of training and testing images which are DICOM files of chest radiographs (CXRs) and two CSV files. One CSV file contains labeled training data and the other is made up of details of the training data of the labeled CSV file. The breakdown of the dataset files are as follows:

1. **`train_labels.csv`**: This CSV file contains the annotations for patients' chest radiographs. These are the bounding boxes. It also contains patient IDs and their corresponding labels.
2. **`detailed_class_labels.csv`**: This CSV file contains detailed label description for each patient ID. It can be used for more nuanced classifications of classes.
3. **`train_images`***:* A directory containing chest radiograph images of patients. Each filename corresponds to a patient.
4. **`test_images`***:* A directory containing images that will be used to test the final model developed to detect cases in pneumonia.

The labeled training data CSV contains data provided as a set of patient IDs (which correspond to the filenames of the training images) and bounding boxes. The bounding boxes indicate the areas of the image that are infected with pneumonia and are defined as follows: `x`, `y`, `width`, `height`. The `x`

and `y` are coordinates of the upper left corners of the bounding boxes while the `width` and `height` are the lengths of the edges of the bounding boxes. The file also contains a `Target` column that indicates if a patient ID has pneumonia. There may be multiple rows per patient ID. This means that there may be more than one CXR per patient.

| | patientId | x | y | width | height | Target |
|---|---|---|---|---|---|---|
| 0 | 0004cfab-14fd-4e49-80ba-63a80b6bddd6 | NaN | NaN | NaN | NaN | 0 |
| 1 | 00313ee0-9eaa-42f4-b0ab-c148ed3241cd | NaN | NaN | NaN | NaN | 0 |
| 2 | 00322d4d-1c29-4943-afc9-b6754be640eb | NaN | NaN | NaN | NaN | 0 |
| 3 | 003d8fa0-6bf1-40ed-b54c-ac657f8495c5 | NaN | NaN | NaN | NaN | 0 |
| 4 | 00436515-870c-4b36-a041-de91049b9ab4 | 264.0 | 152.0 | 213.0 | 379.0 | 1 |
| 5 | 00436515-870c-4b36-a041-de91049b9ab4 | 562.0 | 152.0 | 256.0 | 453.0 | 1 |
| 6 | 00569f44-917d-4c86-a842-81832af98c30 | NaN | NaN | NaN | NaN | 0 |
| 7 | 006cec2e-6ce2-4549-bffa-eadfcd1e9970 | NaN | NaN | NaN | NaN | 0 |
| 8 | 00704310-78a8-4b38-8475-49f4573b2dbb | 323.0 | 577.0 | 160.0 | 104.0 | 1 |
| 9 | 00704310-78a8-4b38-8475-49f4573b2dbb | 695.0 | 575.0 | 162.0 | 137.0 | 1 |

*Figure 1: Training Data Label*

`NaN` values mean that there are no corresponding bounding boxes for a patient. This means that the chest radiograph of the patient does not indicate any case of pneumonia. This is evidenced with the corresponding `Target` value equating to 0.

### 2.1.1 Data Distribution

Analyzing the dataset distribution, we discover that there are more patients (roughly 3x) without lung opacity than those with lung opacity.

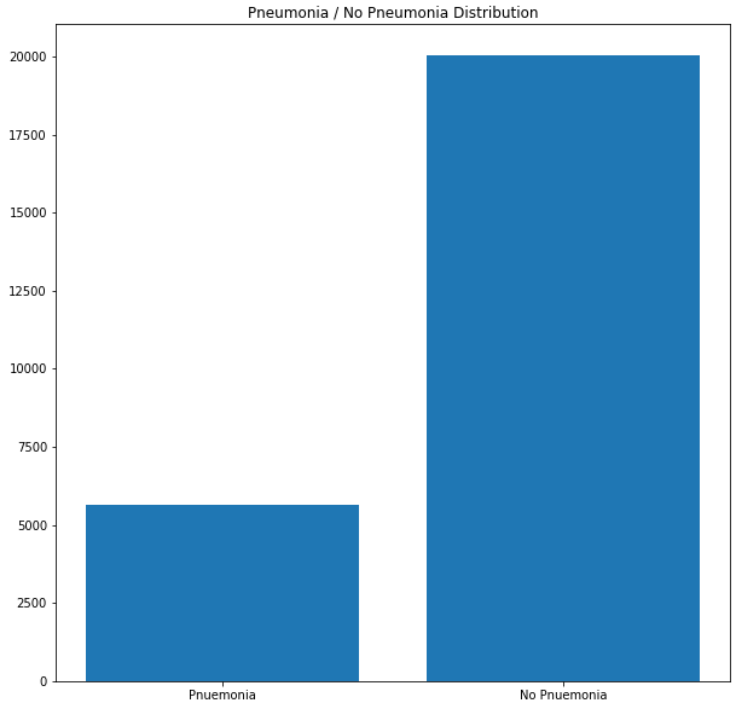


*Figure 2: Label Distribution*

This in no way indicates that all the patients without pneumonia or lung opacity have a clean bill of health. Data in **`detailed_class_labels.csv`** further indicates that there might be other issues that are wrong with a patient's lung other than lung opacity. The chart below gives a more nuanced distribution of the patients' CXR dataset.

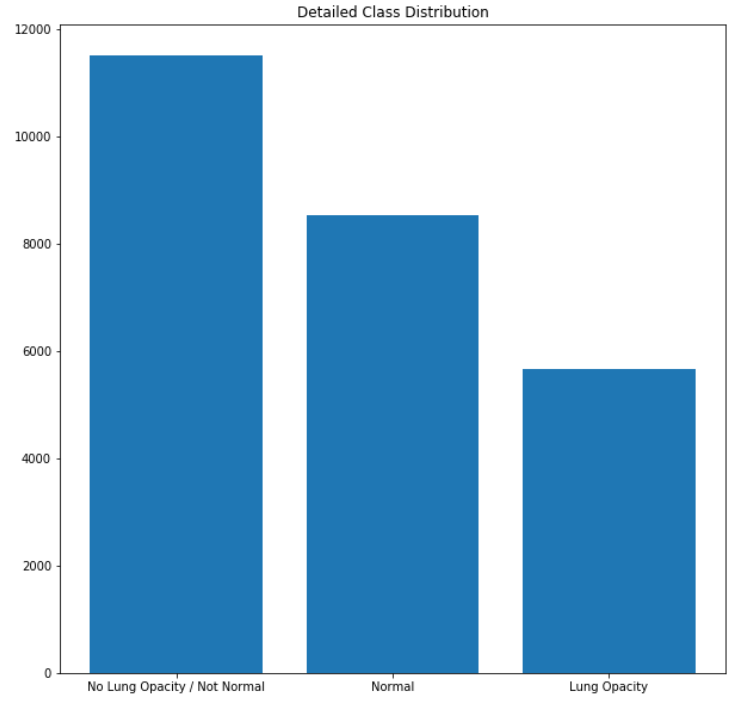


*Figure 3: Detailed Label Distribution*

This tells us that most of the patients, although not having lung opacity, have some defect or abnormality indicated in their CXR. The nature of the defect or abnormality could be one of any lung related defects.

### 2.1.2 Chest Radiograph Images

The images provided are of a special format known as DICOM files (`*.dcm`). They contain a combination of header metadata as well as underlying raw image arrays for pixel data. All DICOM files in the dataset are of a standard shape (1024 x 1024) that have already been preprocessed (scaling, annotations etc.). This shape is adequate for our convolutional neural network to handle.

```
(0008, 0005) Specific Character Set             CS: 'ISO_IR 100'
(0008, 0016) SOP Class UID                      UI: Secondary Capture Image Storage
(0008, 0018) SOP Instance UID                   UI: 1.2.276.0.7230010.3.1.4.8323329.14545.1517874380.120103
(0008, 0020) Study Date                         DA: '19010101'
(0008, 0030) Study Time                         TM: '000000.00'
(0008, 0050) Accession Number                   SH: ''
(0008, 0060) Modality                           CS: 'CR'
(0008, 0064) Conversion Type                    CS: 'WSD'
(0008, 0090) Referring Physician's Name         PN: ''
(0008, 103e) Series Description                 LO: 'view: PA'
(0010, 0010) Patient's Name                     PN: '38b67cdf-650c-4c90-bcfb-745f0db2ec97'
(0010, 0020) Patient ID                         LO: '38b67cdf-650c-4c90-bcfb-745f0db2ec97'
(0010, 0030) Patient's Birth Date               DA: ''
(0010, 0040) Patient's Sex                      CS: 'F'
(0010, 1010) Patient's Age                      AS: '44'
(0018, 0015) Body Part Examined                 CS: 'CHEST'
(0018, 5101) View Position                      CS: 'PA'
(0020, 000d) Study Instance UID                 UI: 1.2.276.0.7230010.3.1.2.8323329.14545.1517874380.120102
(0020, 000e) Series Instance UID                UI: 1.2.276.0.7230010.3.1.3.8323329.14545.1517874380.120101
(0020, 0010) Study ID                           SH: ''
(0020, 0011) Series Number                      IS: '1'
(0020, 0013) Instance Number                    IS: '1'
(0020, 0020) Patient Orientation                CS: ''
(0028, 0002) Samples per Pixel                  US: 1
(0028, 0004) Photometric Interpretation         CS: 'MONOCHROME2'
(0028, 0010) Rows                               US: 1024
(0028, 0011) Columns                            US: 1024
(0028, 0030) Pixel Spacing                      DS: ['0.14300000000000002', '0.14300000000000002']
(0028, 0100) Bits Allocated                     US: 8
(0028, 0101) Bits Stored                        US: 8
(0028, 0102) High Bit                           US: 7
(0028, 0103) Pixel Representation               US: 0
(0028, 2110) Lossy Image Compression            CS: '01'
(0028, 2114) Lossy Image Compression Method     CS: 'ISO_10918_1'
(7fe0, 0010) Pixel Data                         OB: Array of 154632 bytes
```

*Figure 4: Chest Radiograph Meta-data*

The `Pixel Data` attribute of the DICOM file contains the necessary data needed to visualize, train, validate and test the model. Below are examples of CXRs showing a normal lung, no lung opacity/not normal and lungs that have pneumonia.

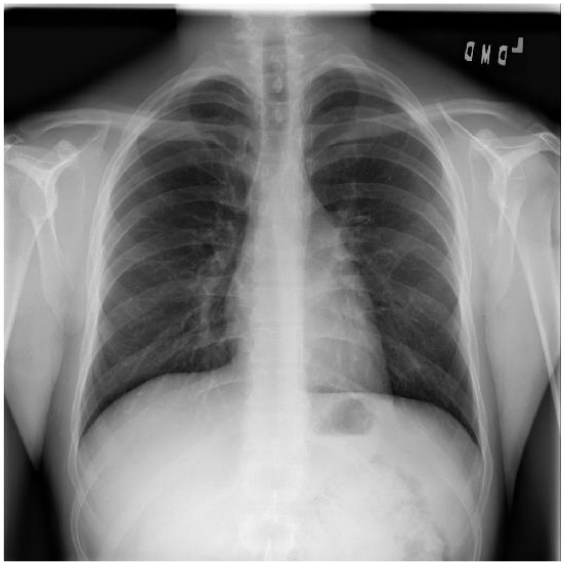

*Figure 5: Normal Lungs*

*Figure 6: No Lung Opacity/Not Normal*

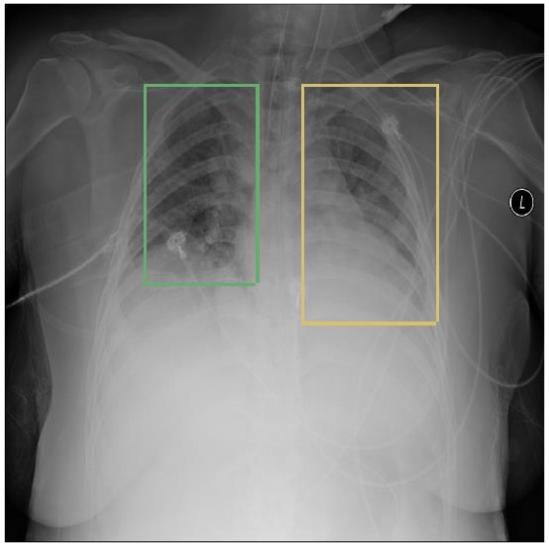

*Figure 7: Lung Opacity/Pneumonia*

## 2.2 Algorithms and Techniques

This pneumonia detection problem will be tackled using a deep learning (computer vision) algorithm known as Mask Region-based Convolutional Network (Mask R-CNN)[1]. Mask R-CNN is an instance segmentation algorithm that allows for the identification of pixel-wise locations of a desired class (pneumonia opacity). This means segmenting individual cases on pneumonia in a CXR.

Instance segmentation is difficult because it requires the correct detection of all objects in an image while also accurately segmenting all instances of the object. It combines elements from classical computer vision tasks of object detection (classifying individual objects and localizing each with bounding boxes) and semantic segmentation (classifying each pixel into a fixed set of categories without differentiating object instances).

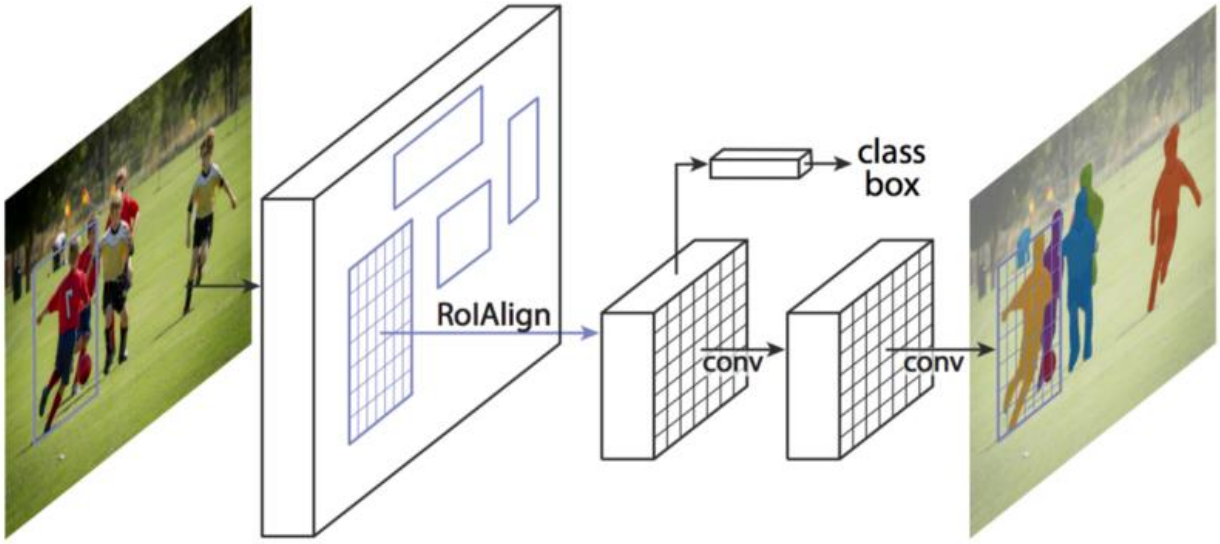


*Figure 8: Mask R-CNN framework for instance segmentation*

Mask R-CNN combines Faster R-CNN[2]'s detection of regions proposal and recognition of objects in each region by predicting bounding boxes with Fully Convolutional Network[3]'s (FCN semantic segmentation) to detect instances of an object class in an image.

At a high level, Mask R-CNN consists of the following:

[1] https://arxiv.org/pdf/1703.06870.pdf
[2] https://arxiv.org/pdf/1506.01497.pdf
[3] https://arxiv.org/pdf/1605.06211.pdf]

### 2.2.1 Backbone

The backbone is a standard convolutional neural network (ResNet-50, ResNet-101 or ResNeXt-101) that serves as a feature extractor – early layers detect low level features (edges and corners) while later layers detect high level features (lung opacity).

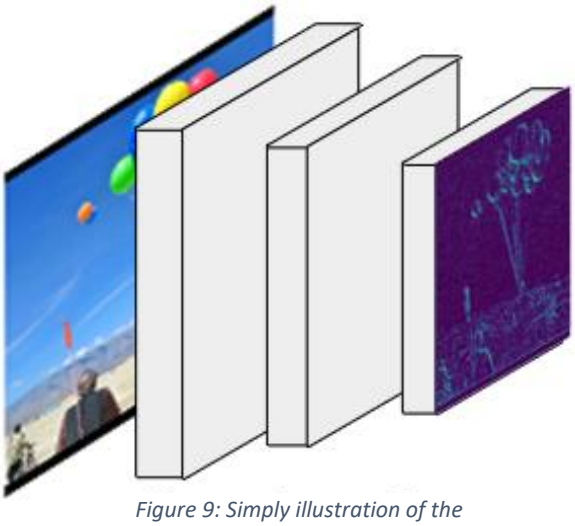

*Figure 9: Simply illustration of the backbone network*

Images are converted from $1024 \times 1024 \times 3$ (RGB), through the backbone network, to a feature map shape of $32 \times 32 \times 2048$. The feature map is passed to subsequent stages of the algorithm. Feature Pyramid Network (FPN)[4] is used to improve the standard feature extraction pyramid by adding a second pyramid that takes higher level features from the first pyramid and passes them down to the lower layers. This allows features at every level to have both higher and lower level features.

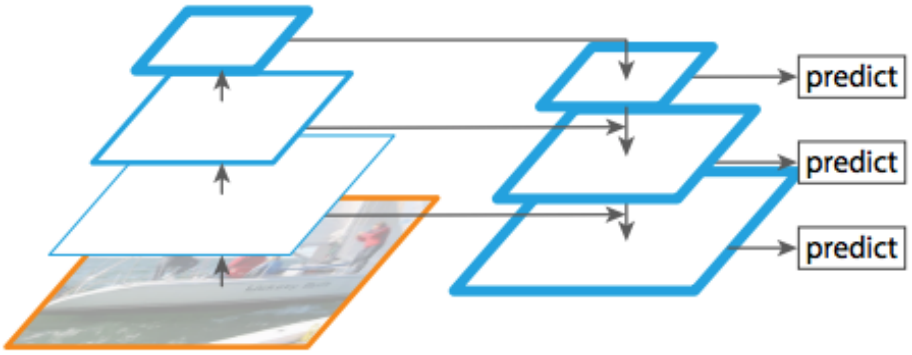


*Figure 10: Feature Pyramid Network*

### 2.2.2 Region Proposal Network (RPN)

Region Proposal Network is a lightweight neural network that scans the image and finds areas that contain the object class. These regions are called anchors and can range is size and sometimes overlap to cover as much of the image's surface as possible. RPNs run very fast because they don't scan over the image directly but over the backbone feature map in parallel (on GPU). This allows the RPN to reuse the extracted features efficiently to avoid duplicate calculations.

The RPN generates two outputs for each anchor:

[4] https://arxiv.org/pdf/1612.03144.pdf

1. **Anchor Class**: Foreground (FG) or background (BG). The FG class indicates there is likely an object in the anchor.

2. **Bounding Box Refinement (BBox)**: An FG anchor might not be centered perfectly over the object, so the RPN estimates the percentage change in the `x`, `y`, `width` and `height`, to refine the anchor box to fit the object properly.

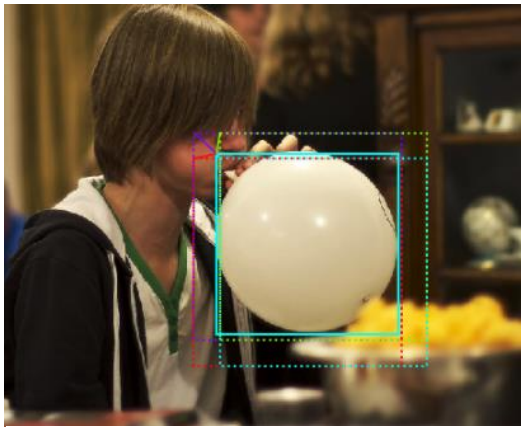

*Figure 11: 3 anchor boxes (dotted) and the shift/scale applied to them to fit the object precisely (solid). Several anchors can map to the same object.*

The top anchors that likely contain the object class are selected and their location and size refined. In the situation that several anchors overlap, the anchor with the highest FG score is selected and the rest, discarded. The final regions of interest are then passed to the RoI classifier and bounding box regressor.

### 2.2.3 RoI Classifier & Bounding Box Regressor

The RoI Classifier & Bounding Box Regressor runs on the RoIs proposed by the RPN and generates two outputs:

1. **Class**: This is the class of the object in the RoI. This network is deeper than the RPN class output and has the capacity to classify regions of specific classes (lung opacity). It also generates a background class which discards the RoI.
2. **Bounding Box Refinement (BBox)**: Similar to the RPN, it further refines the location and size of the bounding box and captures the object.

Classifiers typically require a fixed input size but due to the BBox in the RPN, the RoIs have different box sizes. RoI Pooling is used to crop and resize feature maps to a fixed size (e.g. $7 \times 7$) for each RoI. The Mask R-CNN paper proposed an RoIAlign layer that handles the polling process.

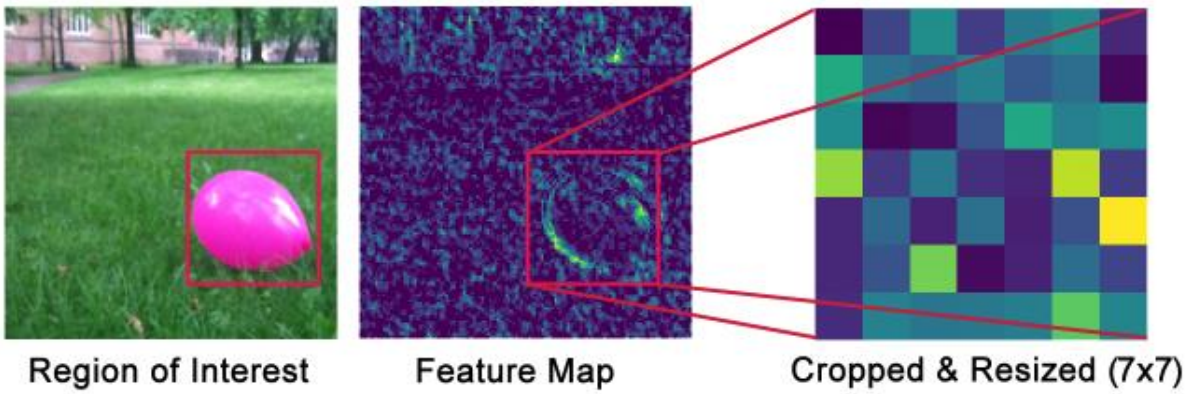


*Figure 12: RoI Pooling*

### 2.2.4 Mask Representation

The mask branch is a convolutional network that task the RoI classifier and generate masks for them. The generated masks are low resolution masks ($28 \times 28$ pixels) but they are soft masks represented by floating point numbers instead of integers. So, they hold more detail than binary masks. During training, the ground-truth mask is scaled down to compute the loss and during inference, the predicted mask is scaled up to the size of the RoI bounding box, giving the final masks for each object.

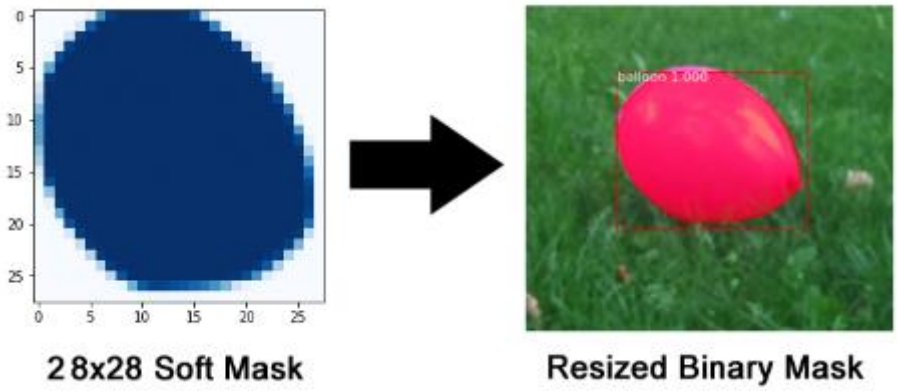


*Figure 13: Mask Representation*

*Figure 14: Final Representation Mask R-CNN*

## 2.3 Benchmark

Mask Region-based Convolutional Neural Network outperforms all state-of-the-art models in object detection and instance segmentation on the Common Object and COntext (COCO) dataset and using varying Intersection over Union (IoU) metrics (this is explained further in the *Metrics* section). State-of-the-art object detection model include R-FCN[5], SSD[6], YOLOv2[7] and NASNet[8]. The following table gives a breakdown of comparison of Mask R-CNN against the other models.

| Model | COCO 2015/2016 (IoU = 0.5) | COCO 2015/2016 (IoU = 0.75) | COCO 2015/2016 (Official metrics) |
|---|---|---|---|
| R-FCN | 53.2% | | 31.5% |
| SSD | 48.5% | 30.3% | 31.5% |
| YOLOv2 | 44.0% | 19.2% | 21.6% |
| NASNet | | 43.1% | |
| Mask R-CNN | 62.3% | 43.3% | 39.8% |

# 3. Methodology

## 3.1 Data Preprocessing

The dataset was provided by the Radiological Society of North America (RSNA) in collaboration with the US National Institutes of Health, The Society of Thoracic Radiology, MD.ai and Kaggle. Fortunately, the dataset has already been preprocessed with the following considerations:

1. The relatively high dynamic range, high bit-depth images have been rescaled to 8-bit encoding. This means that the images have already been windowed and leveled. In clinical practice, manipulating image bit-depth is typically done manually by a radiologist to highlight certain disease processes.
2. The typically large original image matrices of $> 2000 \times 2000$ have been resized to $1024 \times 1024$. The diagnosis of most pneumonia cases can be done at this resolution. With Mask R-CNN, this is also a very adequate resolution to train the model (although training can be done at lower resolutions with a certain degree of accuracy).

Typically, the process required in data processing are as follows:

1. Windowing and leveling high bit-depth chest radiographs. This is usually handled by radiologists. These images will then be scaled down to 8-bit encoding.
2. Compressing image sizes from $> 2000 \times 2000$ to $1024 \times 1024$.
3. Annotating regions on the CXR infected with pneumonia. Annotating the regions generally involves creating ground-truth bounding boxes (`x`, `y`, `width` and `height`) around the infected areas of the CXR.

[5] Region-based Fully Connected Network (F-RCN) [https://arxiv.org/pdf/1605.06409.pdf]
[6] Single-Shot Detector (SSD) [https://arxiv.org/pdf/1512.02325.pdf]
[7] You Only Look Once (YOLO) [https://arxiv.org/pdf/1506.02640.pdf]
[8] Neural Architecture Search Net (NASNet) [https://arxiv.org/pdf/1611.01578.pdf]

4. An alternative to annotations is creating individual image masks of each affected region and feeding the masks into the training model.

The implementation of Mask R-CNN algorithm for this project can generate masks of the infected areas of the image via annotations provided or by using directly, the provided mask images of affected areas.

## 3.2 Implementation

### 3.2.1 Utilities

The `utils/utils.py` file contains functions to handle common tasks used throughout the project.

*Dataset Download*

`download_dataset` and `download_mask_rcnn` functions simply download the dataset and Mask R-CNN algorithm. In addition to downloading the dataset, `download_dataset` unzips, prepares and structures the dataset for use.

*Dataset Parser*

In [2.1], it is explained that the `train_labels.csv` contains rows of patients and the coordinates of lung opacity (for those that have) and that some patients have multiple lung opacities represented as multiple rows in the file. To use this data, this information needs to be parsed into an object of patients and their information. The `parse_data` function converts each patient into an object of the form:

```
patientId: {
        'id': 'patientId',
        'dicom': path/to/dicom/file,
        'label': either 0 or 1 for normal or pneumonia,
        'boxes': list of box(es)
}
```

The function takes in the directory of training images and the train labels as a dataframe and returns a parsed dataset.

```
def parse_data(dicom_dir, df):
    extract_box = lambda row: [row['x'], row['y'], row['width'], row['height']]
    parsed = {}
    for n, row in df.iterrows():
        pid = row['patientId']
        if pid not in parsed:
            parsed[pid] = {
                'id': pid,
                'dicom': dicom_dir + '/%s.dcm' % pid,
                'label': row['Target'],
                'boxes': []}
        if parsed[pid]['label'] == 1:
            parsed[pid]['boxes'].append(extract_box(row))
    return parsed
```

*Visualizing DICOM file*

The parsed data can be visualized, and this is achieved with the `draw_dicom` function. This function uses the `overlay_box` function to draw the bounding boxes of lung opacity for patients with pneumonia.

```
def draw_dicom(data, plt_pos=None, figsize=None):
    dicom = pydicom.read_file(data['dicom'])
    image = dicom.pixel_array
    image = np.stack([image] * 3, axis=2)
    for box in data['boxes']:
        rgb = np.floor(np.random.rand(3) * 256).astype('int')
        image = overlay_box(image=image, box=box, rgb=rgb, stroke=6)
    if figsize is not None:
        pylab.figure(figsize=figsize)
    if plt_pos is not None:
        pylab.subplot(plt_pos)
    pylab.imshow(image, cmap=pylab.cm.gist_gray)
    pylab.axis('off')


def overlay_box(image, box, rgb, stroke=1):
    box = [int(b) for b in box]
    x1, y1, width, height = box
    y2 = y1 + height
    x2 = x1 + width
    image[y1:y1 + stroke, x1:x2] = rgb
    image[y2:y2 + stroke, x1:x2] = rgb
    image[y1:y2, x1:x1 + stroke] = rgb
    image[y1:y2, x2:x2 + stroke] = rgb
    return image
```

*Loading Last Trained Model*

Training can be stopped and continued at any time. The `load_last_model` function ensures that training can be continued from its last training epoch instead of starting afresh.

```
def load_last_model(model_dir, config):
    dir_names = next(os.walk(model_dir))[1]
    key = config.NAME.lower()
    dir_names = filter(lambda f: f.startswith(key), dir_names)
    dir_names = sorted(dir_names)
    if not dir_names:
        import errno
        raise FileNoteFoundError(errno.ENOENT, 'Could not find model directory under {}'.format(model_dir))
    fps = []
    for d in dir_names:
        dir_name = os.path.join(model_dir, d)
        checkpoints = next(os.walk(dir_name))[2]
        checkpoints = filter(lambda f: f.startswith("mask_rcnn"), checkpoints)
        checkpoints = sorted(checkpoints)
        if not checkpoints:
            print('No weight files in {}'.format(dir_name))
        else:
            checkpoint = os.path.join(dir_name, checkpoints[-1])
            fps.append(checkpoint)
    weights_path = sorted(fps)[-1]
    return weights_path
```

*Detecting test data*

Given the number of DICOM images for test, the `detect` function runs inferences on the test data with the trained model and saves the result in a CSV file called `detections.csv`.

```
def detect(model, dicom_dir, config, csv_path='detections.csv', orig_size=1024):
    resize_factor = orig_size / config.IMAGE_SHAPE[0]

    with open(csv_path, 'w') as file:
        for image_id in tqdm(dicom_dir):
            dicom = pydicom.read_file(image_id)
            image = dicom.pixel_array
            if len(image.shape) != 3 or image.shape[2] != 3:
                image = np.stack((image,) * 3, -1)
            patient_id = os.path.splitext(os.path.basename(image_id))[0]
            results = model.detect([image])
            result = results[0]
            out_str = ''
            out_str += patient_id
            assert(len(result['rois']) == len(result['class_ids']) == len(result['scores']))
            if len(result['rois']) == 0:
                pass
            else:
                num_instances = len(result['rois'])
                out_str += ','
                for i in range(num_instances):
                    out_str += ' '
                    out_str += str(round(result['scores'][i], 2))
                    out_str += ' '
                    x = result['rois'][i][1]
                    y = result['rois'][i][0]
                    width = result['rois'][i][3] - x
                    height = result['rois'][i][2] - y
                    bbox_str = "{} {} {} {}".format(x*resize_factor, y*resize_factor,
                        width*resize_factor, height* resize_factor)
                    out_str += bbox_str
            file.write(out_str+'\n')
```

Other functions like `splash_class_ids` work like `overlay_box` but is used within the Jupyter notebook to display visualizations of DICOM images.

## 3.2.2 Configuration

Configuration is divided into training configuration (`PneumoniaDetectionConfig`) and inference configuration (`PneumoniaDetectionInferenceConfig`). Intuitively, the `PneumoniaDetectionConfig` class configures the parameters required for training while the `PneumoniaDetectionInferenceConfig` class configures the parameters for inference/detection. `PneumoniaDetectionConfig` extends the `Config` class of the Mask R-CNN algorithm while `PneumoniaDetectionInferenceConfig` extends the training configuration.

```
class PneumoniaDetectionConfig(Config):
    NAME = 'pneumonia_detection'
    IMAGES_PER_GPU = 2
    VALIDATION_STEPS = 100
    STEPS_PER_EPOCH = 300
    NUM_CLASSES = 1 + 1  # background + 1 pneumonia class
    DETECTION_MIN_CONFIDENCE = 0.9  # skip detections less than 90%
    POST_NMS_ROIS_TRAINING = 2000
    POST_NMS_ROIS_INFERENCE = 2000
    IMAGE_MIN_DIM = 1024
    IMAGE_MAX_DIM = 1024
    RPN_ANCHOR_SCALES = (8, 16, 32, 64, 128)
    MAX_GT_INSTANCES = 200
    DETECTION_MAX_INSTANCES = 400
    RPN_NMS_THRESHOLD = 0.9
    USE_MINI_MASK = True
    RPN_TRAIN_ANCHORS_PER_IMAGE = 64
    TRAIN_ROIS_PER_IMAGE = 128

class PneumoniaDetectionInferenceConfig(PneumoniaDetectionConfig):
    GPU_COUNT = 1
    IMAGES_PER_GPU = 1
    DETECTION_MIN_CONFIDENCE = 0.95  # skip detections less than 95%
```

### 3.2.3 Loading Dataset

The dataset is split into training and validation sets and then loaded into a `PneumoniaDetectionDataset` defined in `pneumonia_detection/pneumonia_detection_dataset.py`. This class also loads the mask of an images passed to it. Which is used in training and inference. The dataset is split in a 9:1 ratio of training to validation datasets. Only dataset with bounding boxes are loaded into the `PneumoniaDetectionDataset` training and validation objects.

Mask R-CNN relies on annotations or masks of objects in images in other for it to perform instance segmentation. Images without annotations will not be useful in training.

To determine if images are loaded properly, a random image is selected and plotted together with its masks/annotations. The snippet below displays a random image and its masks/annotations.

```
image_id = random.choice(train_dataset.image_ids)
image_loc = train_dataset.image_reference(image_id)
image = train_dataset.load_image(image_id)
mask, class_ids = train_dataset.load_mask(image_id)

plt.figure(figsize=(15, 15))

plt.subplot(121)
plt.imshow(image[:, :, 0], cmap='gray')
plt.axis('off')

plt.subplot(122)
masked = np.zeros(image.shape[:2])
for i in range(mask.shape[2]):
    masked += image[:, :, 0] * mask[:, :, i]
plt.imshow(masked, cmap='gray')
plt.axis('off')
```

### 3.2.4 Model Preparation

Once the training and validation datasets are loaded, a Mask R-CNN object is created. Initial weights for training are then loaded by the model. The first time the model was trained, COCO training weights were used. Subsequently, if training is continuing from any epoch, the last trained weight is used. There is also the option of using already trained `pneumonia_detection` model weights if it is available.

To avoid overfitting, the training images are augmented with the `imgaug` library. The prepared model is then trained.

### 3.2.5 Training and Loss Examination

The model is trained for 100 epochs and each epoch runs through 300 steps. This takes about a day and a couple of hours (on Amazon AWS p2.xlarge instance running 1 GPU) based on the configuration parameters defined in the `PneumoniaDetectionConfig` class. The configuration directly affects training time and detection accuracy. The computed loss functions are saved in the `logs/training_log.csv` file.

At the end of training, the final model weights are saved in the `logs/mask_rcnn_pneumonia_detection.h5` file. The plots of log functions are below:

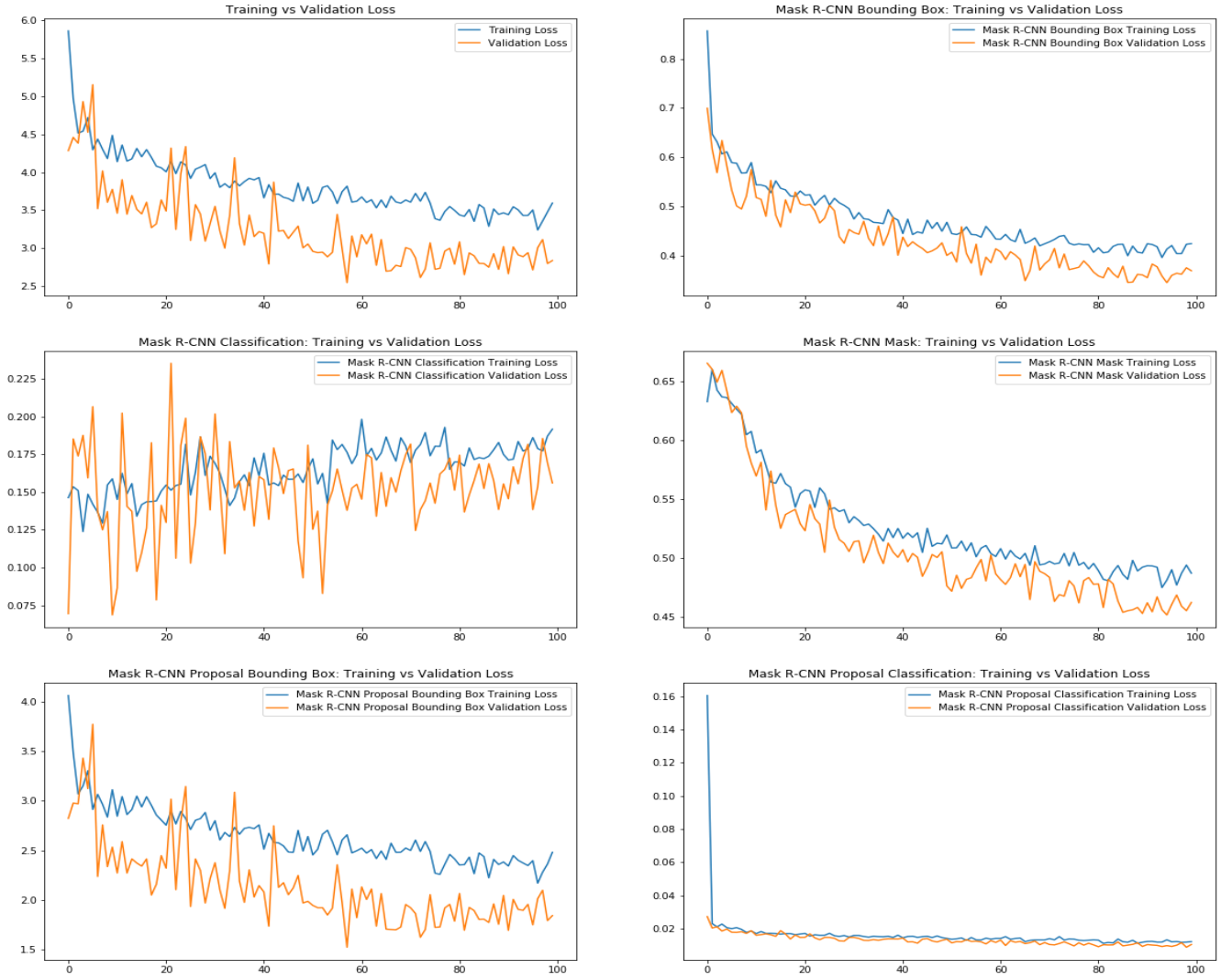


*Figure 15: Loss Functions (Training vs Validation)*

### 3.2.6 Inference & Detection

Inference is tested using the trained model weights on the validation dataset. The result of the inference is visualized, comparing ground-truth images with the inferred counterpart.

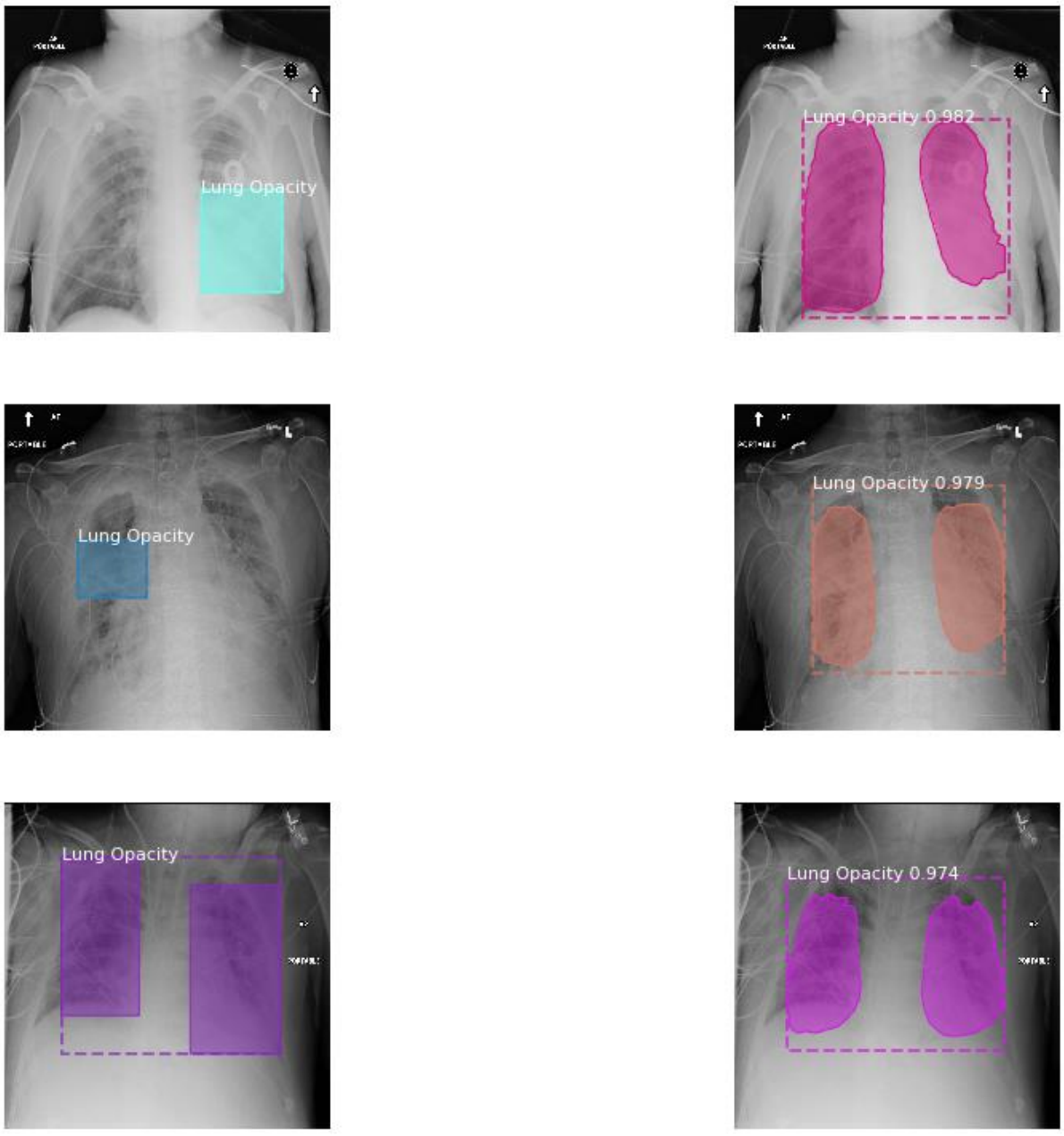


*Figure 16: Sample of pneumonia detection. The images on the left are the ground-truth images while the ones on the right are the detected results of the model*

### 3.2.7 Model Evaluation

Using the metrics described in [1.3], the model is evaluated using the validation dataset. The validation dataset is used because it contains the ground-truth values. The test dataset does not contain any ground-truth values; therefore, it cannot be used for evaluation.

### 3.2.8 Test dataset inference

Finally, the model is used to detect lung opacity in the test set. The results are saved in a CSV file called `detections.csv`. 5 randomly selected test images are visualized (some with bounding boxes and some without).

## 3.3 Refinement

Initially, the focus of training was to get a final model to experiment with. This required optimizing the configuration for speedy training. To achieve that, the number of training epochs was set at 50 and the number of steps per epoch was set at 100. Image sizes were cropped to $64 \times 64$ which required the RPN anchor scales to be reduced to (2, 4, 8, 16, 32). As expected, this configuration yielded very poor lung opacity detection accuracy.

Increasing the size of the images to $128 \times 128$ and $512 \times 512$ increased overall accuracy but not by any significant margin. Increasing the number of training epochs with image sizes at $512 \times 512$ did cause an increase in detection accuracy. This gave the idea that using the original size of the images, increasing the number of epochs, steps per epoch and RPN anchors will cause a significant increase in detection accuracy.

A decision had to be made between using the original size of $1024 \times 1024$ or $512 \times 512$ while also increasing the other parameters mentioned in the paragraph above. My desire was to train the model for 30,000 iterations using 1000 steps per iteration. This will take several weeks of training. Considering a constraint on GPU resources (the model was trained on Amazon AWS), the final decision was to train the model at $1024 \times 1024$ for 100 epochs and 300 steps per epoch using RPN anchor sizes of (8, 16, 32, 64, 128). To ensure some level of detection confidence, the model was configured to train with a detection confidence of >=90%. For inference, the model only determines if there is lung opacity in an image if it has >=95% confidence in its detection. The rest of the final configuration used are available in [3.2.2] and in Figure 17 and 18 below.

The final loss functions and inference, give the outcome of the training using these parameters.

```
BACKBONE                        resnet101
BACKBONE_STRIDES                [4, 8, 16, 32, 64]
BATCH_SIZE                      2
BBOX_STD_DEV                    [ 0.1  0.1  0.2  0.2]
COMPUTE_BACKBONE_SHAPE          None
DETECTION_MAX_INSTANCES         400
DETECTION_MIN_CONFIDENCE        0.9
DETECTION_NMS_THRESHOLD         0.3
FPN_CLASSIF_FC_LAYERS_SIZE      1024
GPU_COUNT                       1
GRADIENT_CLIP_NORM              5.0
IMAGES_PER_GPU                  2
IMAGE_MAX_DIM                   1024
IMAGE_META_SIZE                 14
IMAGE_MIN_DIM                   1024
IMAGE_MIN_SCALE                 0
IMAGE_RESIZE_MODE               square
IMAGE_SHAPE                     [1024 1024    3]
LEARNING_MOMENTUM               0.9
LEARNING_RATE                   0.001
LOSS_WEIGHTS                    {'mrcnn_class_loss': 1.0, 'mrcnn_mask_loss': 1.0, 'mrcnn_bbox_loss': 1.0, 'rpn_bbox_loss': 1.0,
'rpn_class_loss': 1.0}
MASK_POOL_SIZE                  14
MASK_SHAPE                      [28, 28]
MAX_GT_INSTANCES                200
MEAN_PIXEL                      [ 123.7  116.8  103.9]
MINI_MASK_SHAPE                 (56, 56)
NAME                            pneumonia_detection
NUM_CLASSES                     2
POOL_SIZE                       7
POST_NMS_ROIS_INFERENCE         2000
POST_NMS_ROIS_TRAINING          2000
ROI_POSITIVE_RATIO              0.33
RPN_ANCHOR_RATIOS               [0.5, 1, 2]
RPN_ANCHOR_SCALES               (8, 16, 32, 64, 128)
RPN_ANCHOR_STRIDE               1
RPN_BBOX_STD_DEV                [ 0.1  0.1  0.2  0.2]
RPN_NMS_THRESHOLD               0.9
RPN_TRAIN_ANCHORS_PER_IMAGE     64
STEPS_PER_EPOCH                 300
TOP_DOWN_PYRAMID_SIZE           256
TRAIN_BN                        False
TRAIN_ROIS_PER_IMAGE            128
USE_MINI_MASK                   True
USE_RPN_ROIS                    True
VALIDATION_STEPS                100
WEIGHT_DECAY                    0.0001
```

*Figure 17: Training Configuration*

```
Configurations:
BACKBONE                        resnet101
BACKBONE_STRIDES                [4, 8, 16, 32, 64]
BATCH_SIZE                      1
BBOX_STD_DEV                    [0.1 0.1 0.2 0.2]
COMPUTE_BACKBONE_SHAPE          None
DETECTION_MAX_INSTANCES         400
DETECTION_MIN_CONFIDENCE        0.95
DETECTION_NMS_THRESHOLD         0.3
FPN_CLASSIF_FC_LAYERS_SIZE      1024
GPU_COUNT                       1
GRADIENT_CLIP_NORM              5.0
IMAGES_PER_GPU                  1
IMAGE_MAX_DIM                   1024
IMAGE_META_SIZE                 14
IMAGE_MIN_DIM                   1024
IMAGE_MIN_SCALE                 0
IMAGE_RESIZE_MODE               square
IMAGE_SHAPE                     [1024 1024    3]
LEARNING_MOMENTUM               0.9
LEARNING_RATE                   0.001
LOSS_WEIGHTS                    {'rpn_class_loss': 1.0, 'rpn_bbox_loss': 1.0, 'mrcnn_class_loss': 1.0, 'mrcnn_bbox_loss': 1.0,
'mrcnn_mask_loss': 1.0}
MASK_POOL_SIZE                  14
MASK_SHAPE                      [28, 28]
MAX_GT_INSTANCES                200
MEAN_PIXEL                      [123.7 116.8 103.9]
MINI_MASK_SHAPE                 (56, 56)
NAME                            pneumonia_detection
NUM_CLASSES                     2
POOL_SIZE                       7
POST_NMS_ROIS_INFERENCE         2000
POST_NMS_ROIS_TRAINING          2000
ROI_POSITIVE_RATIO              0.33
RPN_ANCHOR_RATIOS               [0.5, 1, 2]
RPN_ANCHOR_SCALES               (8, 16, 32, 64, 128)
RPN_ANCHOR_STRIDE               1
RPN_BBOX_STD_DEV                [0.1 0.1 0.2 0.2]
RPN_NMS_THRESHOLD               0.9
RPN_TRAIN_ANCHORS_PER_IMAGE     64
STEPS_PER_EPOCH                 300
TOP_DOWN_PYRAMID_SIZE           256
TRAIN_BN                        False
TRAIN_ROIS_PER_IMAGE            128
USE_MINI_MASK                   True
USE_RPN_ROIS                    True
VALIDATION_STEPS                100
WEIGHT_DECAY                    0.0001
```

*Figure 18: Inference Configuration*

# 4. Results

## 4.1 Model Evaluation and Validation

During training, a validation set was used to evaluate the model. The evaluation metrics, as described in [1.3], was used with the validation set. Over an IoU threshold ranging from 0.4 to 0.75 and with a step size of 0.05 (i.e. 0.4, 0.45, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75), the mean Average Precision score was 27.76. With an IoU threshold of 0.5, the mean Average Precision score was 43.11.

The final configuration parameters where chosen because, given the resource constraints, they performed the best among other combinations of configuration parameters.

Figure 17 and 18 shows the full list of configuration parameters used in training the model and detecting lung opacity.

## 4.2 Justification

The benchmark results of Mask R-CNN on the COCO 2015/2016 test dataset are:

1. IoU = 0.5: 62.3
2. IoU = 0.75: 43.3
3. Official metrics: 39.8

Evidently, the model trained on lung opacity detection did not meet the evaluation scores of the benchmark. This is, in part, due to the choice of the number of epochs. In the Mask R-CNN paper, the model was trained on 8 GPUs for 160,000 epochs with a batch size of 16 and takes about 44 hours in a synchronized 8-GPU implementation.

The pneumonia detection project was trained on 1 GPU on an Amazon AWS p2 instance, hence the need to reduce the number of epochs. Notwithstanding, the final solution was able to detect cases of lung opacity in most cases. Although, in some cases, the bounding boxes covering detected lung opacity were bigger than their ground-truth equivalent. Considering ground-truth bounding boxes were annotated manually, it's also possible the model can detect regions of some pneumonia not detected manually.

# 5. Conclusion

## 5.1 Free-Form Visualization

We will inspect the model to see how it detects lung opacity in a patient with ID 22c1b139-cb5e-41cd-83fd-7e06d6f4038e. The patient's CXR is located in `dataset/train_images/22c1b139-cb5e-41cd-83fd-7e06d6f4038e.dcm`.

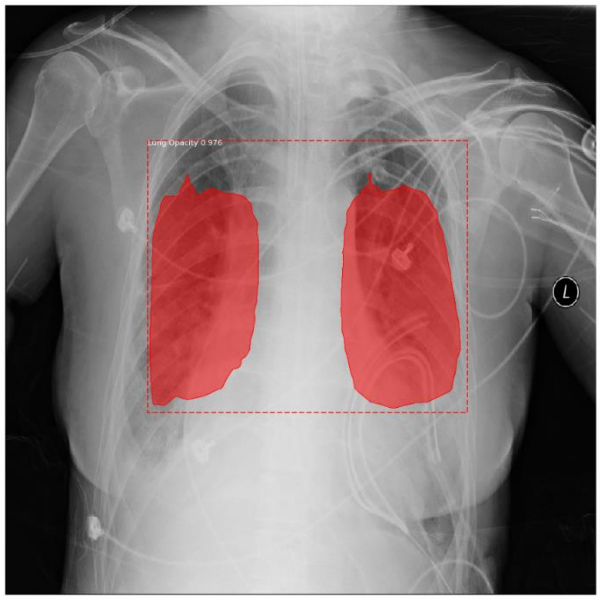


*Figure 19: Final model detection of patient 22c1b139-cb5e-41cd-83fd-7e06d6f4038e*

**Stage 1: Regional Proposal Network (RPN)**

The RPN runs a lightweight binary classifier on a lot of bounding boxes over the image and returns object/no-object scores. Bounding boxes with high *objectness* score are passed to stage two to be classified.

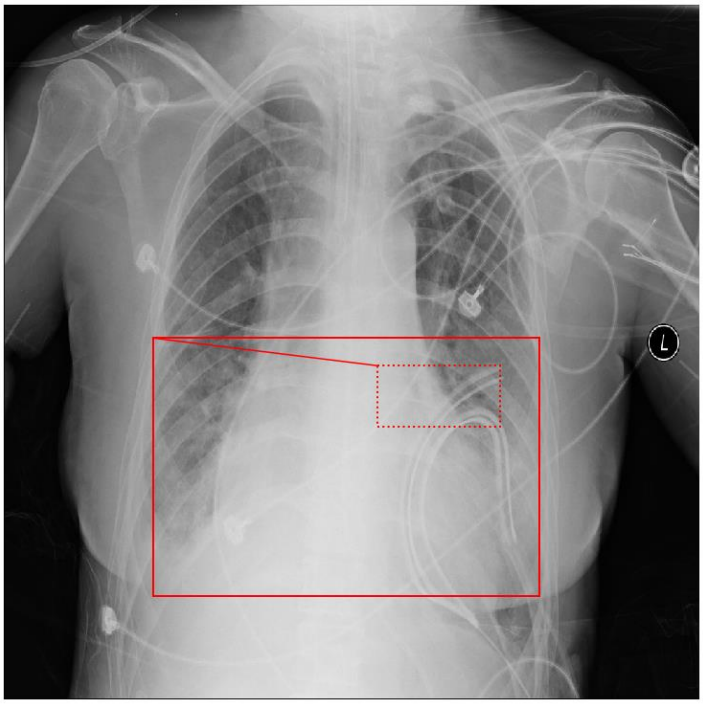

*Figure 20: Positive anchors before refinement (dotted) and after refinement (solid)*

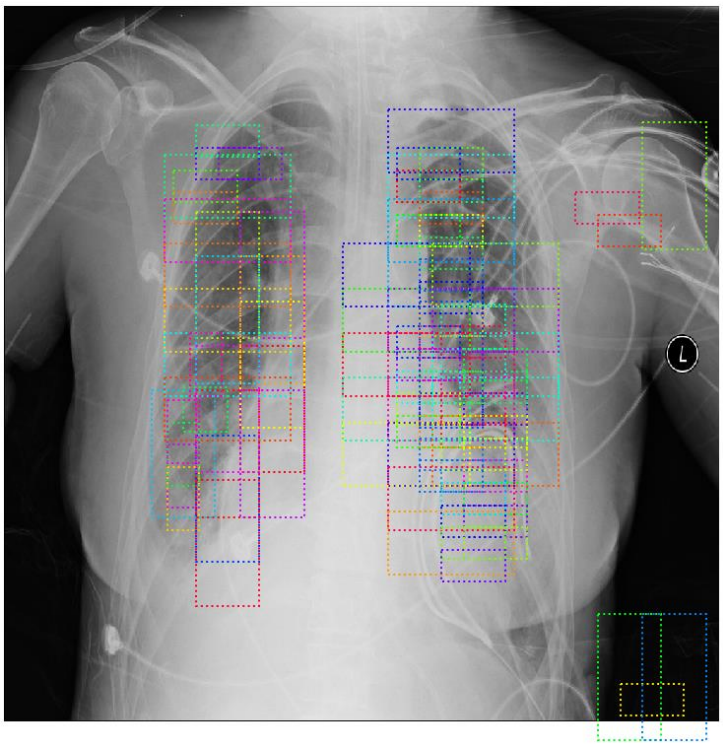

*Figure 21: Top anchors by score (before refinement)*

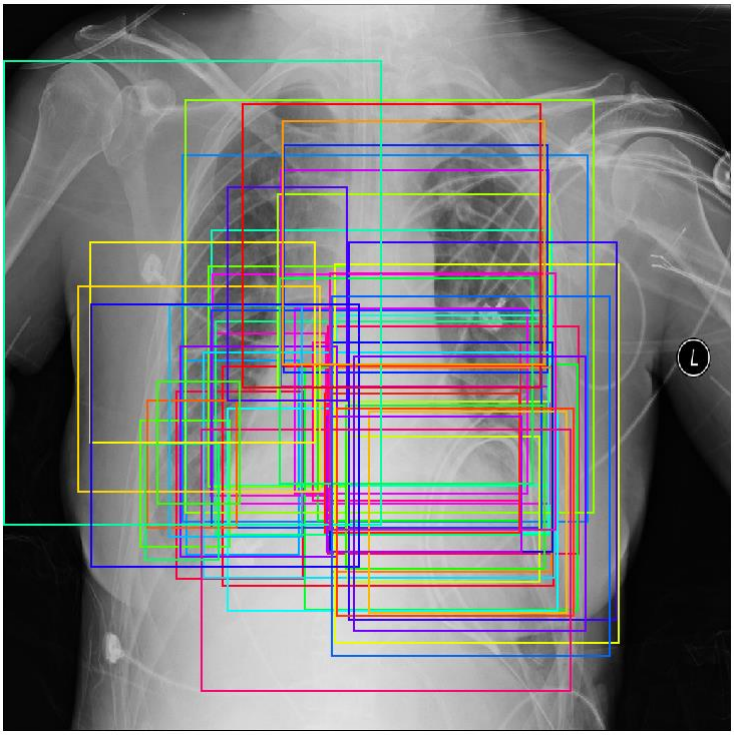

*Figure 22: Final proposals*

**Stage 2: Proposal Classification**

This stage takes the region proposals from the RPN and classifies them.

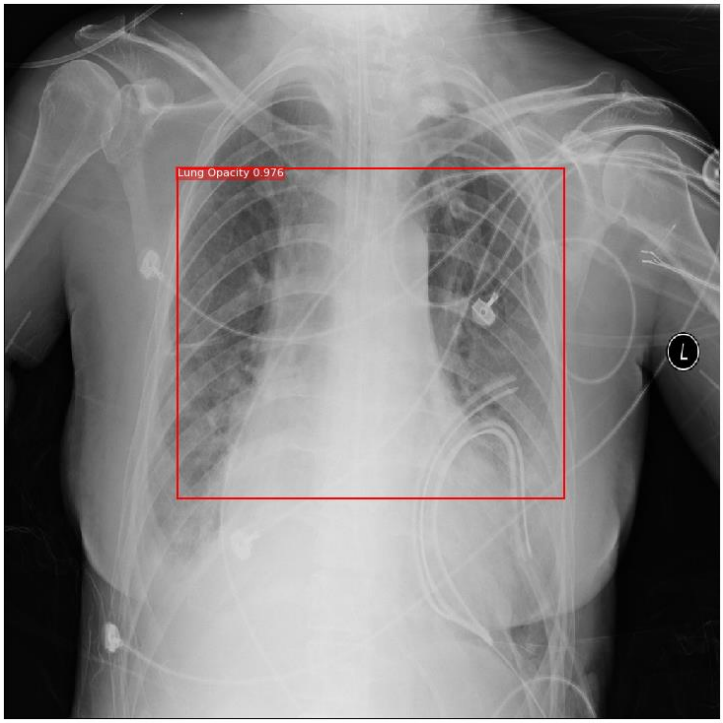


*Figure 23: A class probability of 1 is first detected with a confidence score of 97.6%. This is the final proposal classification*

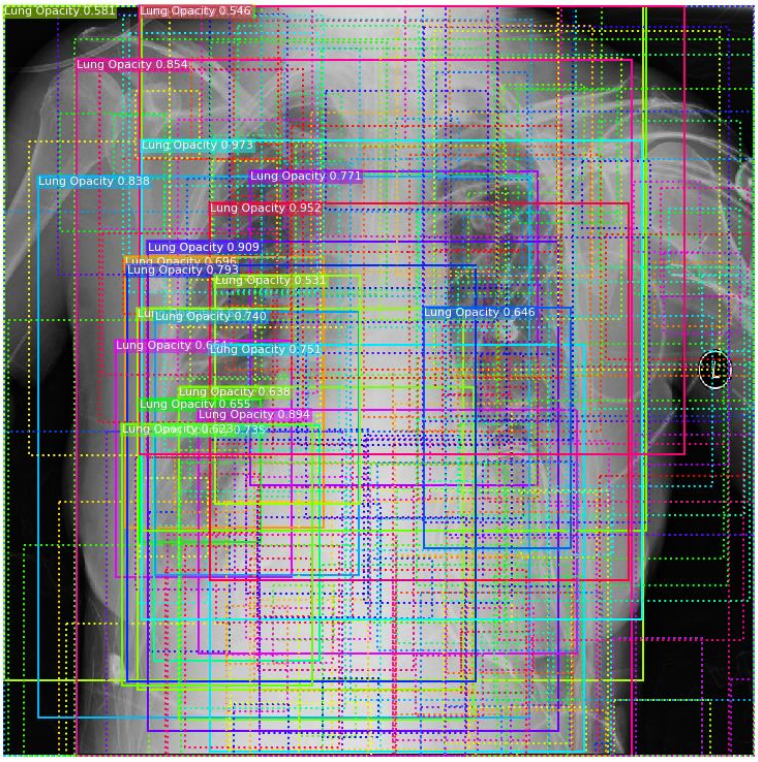


*Figure 24: The model proposes several ROIs with varying confidence scores before refinement is carried out on the proposals.*

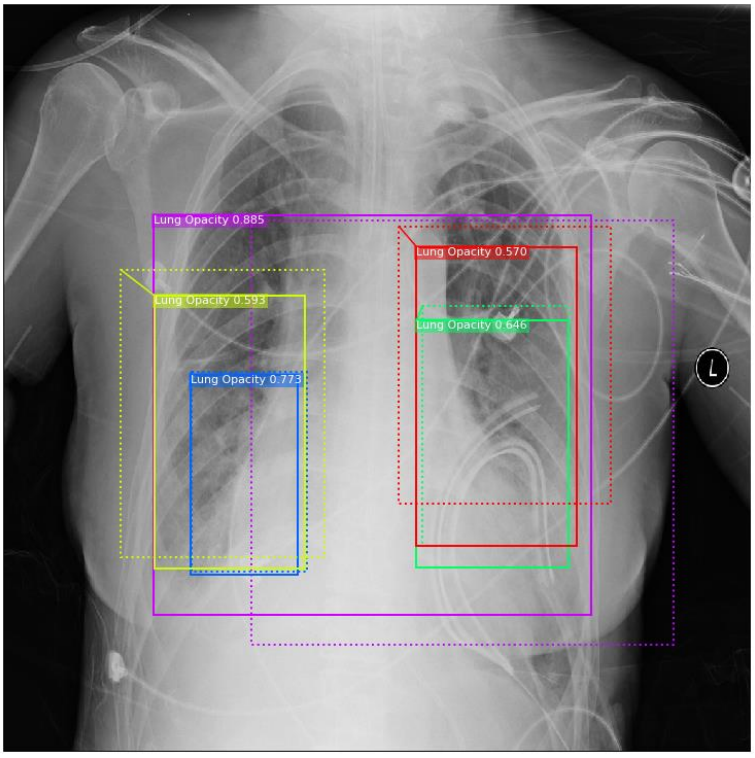


*Figure 25: Bounding box refinement is applied after the initial proposal to produce this result*

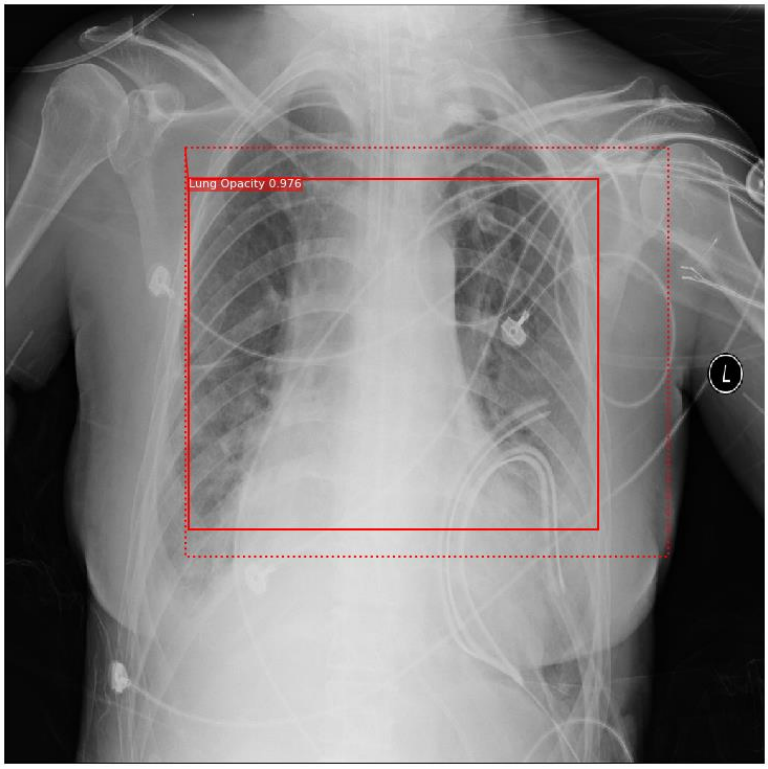


*Figure 26: After Non-Max Suppression is applied to the refined proposal, the final bounding box of Figure 23 is produced*

**Stage 3: Generating Masks**

This stage takes the detections (refined bounding boxes and class IDs) from the previous layer and runs the mask head to generate segmentation masks for every instance.

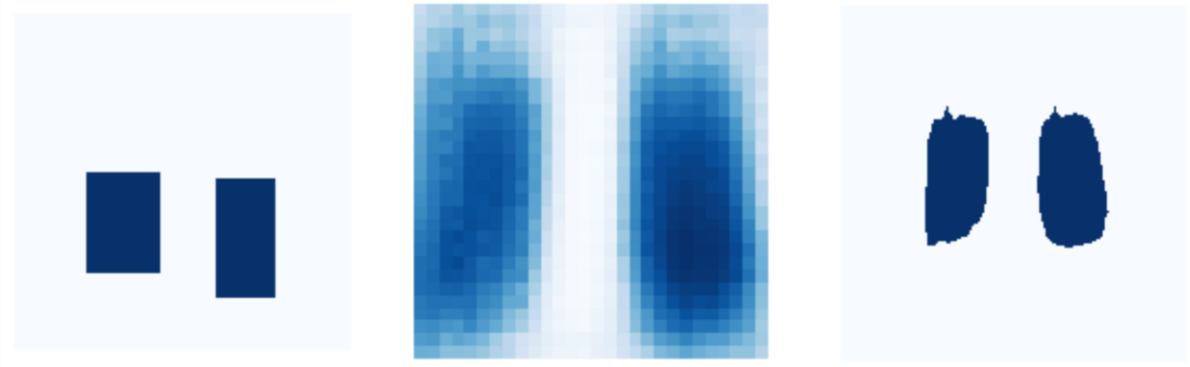

*Figure 27: Mask generation*

The image on the left of Figure 27 is the target mask generated from the annotations provided in `train_labels.csv`. The middle image is the mask generated by the model after detection and before refinement. The mask on the right is the final generated mask after the model has detected lung opacity.

## 5.2 Reflection

The end-to-end process I employed for tackling the pneumonia/lung opacity detection problem can be summarized as follows:

1. Downloading, studying and sampling the dataset.
2. Determining the strategy and algorithm to use in solving the problem. There are several options available for tackling computer vision problems. I had to first decide if I wanted to have a classification, semantic segmentation, object detection or instance segmentation solution. The outcome of this decision determines the sort of algorithm to be used.
3. Preparing the dataset for training.
4. Training, hyperparameter tuning, rinsing and repeating.
5. Evaluation and testing.
6. Model analysis.

Step 4 was particularly challenging for me because a lot of patience was involved. It took several hours to several days to train depending on different hyperparameters and sometimes the outcome of the training was poor. Hyperparameter tuning is an experimental process that cannot be rushed especially with computer vision problems.

The most interesting part of the process was comparing detected pneumonia regions with their ground-truth values. Since the ground-truth values are rectangles, it is unlikely the detected masks will also be rectangles. This is because the ground-truth values where done manually therefore, some small percentage of the bounding boxes might not contain lung opacity. The model, on the other hand, attempts to detect exact pixels that have pneumonia.

## 5.3 Improvement

The pneumonia detection model can be more accurate in detecting lung opacity. With the available resources, I would train the model for at least 30,000 iterations with 1000 steps per iteration. For a more nuanced result, annotating both the “Normal Lung” and “No Lung Opacity / Not Normal” CRXs and training all three classes (Lung Opacity, Normal, No Lung Opacity / Not Normal), can teach the model the distinctions between the three classes. That way, the model can generalize to detect normal and defective lungs.

For general purpose training and detection, increasing the number of classes by providing more annotations of different lung diseases will produce a model that can detect different types of lung ailments. This can result in a robust solution that medical practitioners can use for diagnosing varying lung defects in patients.