Comp 540 Term Project

Bill Huang

Dec 6th 2024

Contents

1 Introduction

1.1 Background

1.1.1 Motivation

As humans, breathing is essential for our survival. We rely on our lungs to perform critical functions like breathing and gas or air exchange. Lungs are undoubtedly one of the most important organs for us. When someone has a collapsed lung, the person will have trouble breathing, and it could become a life-threatening event. A collapsed lung, also known as pneumothorax, can be caused by a blunt chest injury, damage from underlying lung disease, or most horrifying — it may occur for no obvious reason at all.

Pneumothorax is usually diagnosed by a radiologist on a chest X-ray, and can sometimes be very difficult to confirm as it requires manual segmentation of each image from experienced physicians. However, with the development of Artificial Intelligence, technology for image processing has become more and more precise for identifying these conditions within the X-ray images. Nevertheless, it is crucial to correctly identify these symptoms from the X-ray images. This paper aims to outline the process of developing a model and methods to correctly classify pneumothorax, as well as segment it from the X-ray images with precision.

Different researches have been conducted in order to increase the accuracy for this topic, especially with the use of deep learning techniques including different neural networks in order to process the images.

1.1.2 Technical Background

One of the most popular techniques used to address similar problems like pneumothorax classification is the use of Convolutional Neural Networks. Convolutional Neural Network is a popular machine learning model used for image processing where it usually consists of several layers. The first layer, convolutional layer, applies a filter (usually in the form of a matrix), called the convolution, on the input image, where it takes the dot product of the convolution with the image and outputs a new matrix which is a new representation of the input image. Strides and padding may be included for these convolutions. Strides represents the number of pixels by which the convolution moves across the image each time. Padding represents the process of adding zeros across the borders of the input image so that we can have a more desired dimension for the output matrix. The next layer is a pooling layer, where it can be a max pooling layer or an average pooling layer. An n x n max pooling layer takes the maximum value from each n x n grid from the output matrix, while an n x n average pooling layer takes the average value of each n x n grid from the output matrix. The last layer is a fully connected layer. In this layer, each neuron is connected to

every neuron in the output layer, and an activation function is given so that the final prediction is computed based on the output of the function.

Some popular activation functions are: Sigmoid function, Tanh function, ReLU function, and so on.

$$\text{Sigmoid: } S(x) = \frac{1}{1+e^{-x}}$$

$$\text{Tanh: } T(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$$

$$\text{ReLU: } R(x) = max(0, x)$$

These activation functions allow the neural networks to compute the final prediction as well as to optimize during the training process.

Figure 1 shows the architecture of a convolutional neural network that was broadly used for handwritten digit recognition. We can see that in the first layer, we apply a convolution (matrix) with different channels to the input image, so we get a new output matrix for each channel. After this layer, we apply a max pooling layer to the output matrices from the previous layer so that it reduces the dimensionalities of these matrices. Then we use the same process, applying the convolution and a max pooling layer again. At the end, we add two fully connected layers with ReLU activation function to get the predicted probabilities of each digit.

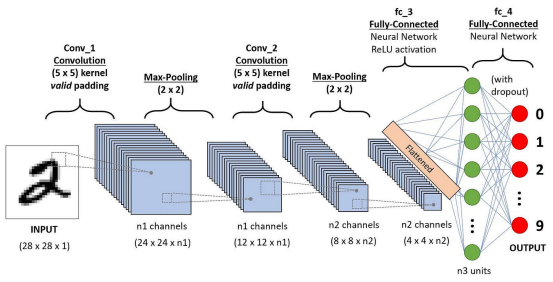

Figure 1: Convolutional Neural Network Architecture

There are several convolutional neural network architectures that are widely used in modern days, namely AlexNet, VGG, ResNet, and so on. All of these models use the general idea of convolutional neural networks, while having some variation within the structures.

AlexNet won the ImageNet Large Scale Visual Recognition Challenge in 2012 which was a image recognition and object detection competition. Figure 2 shows the architecture of an AlexNet model. It includes a 11 x 11 convolution, a 5 x 5 convolution, and three 3 x 3 convolutions in the first, third, and fifth layers respectively; it adds a 3 x 3 max pooling layer after all the convolutional layers. At the end, it adds two fully connected layers with 4096 neurons in each layer, and used the ReLU function as the activation function. The creator of this model, Alex Krizhevsky and Ilya Sutskever, realized that the computation bottlenecks of convolutional neural networks back then could all be parallelized in hardware. This network, that won the competition by a large margin, was able to show that the features obtained by learning can transcend manually-designed features, breaking the previous paradigm in computer vision.

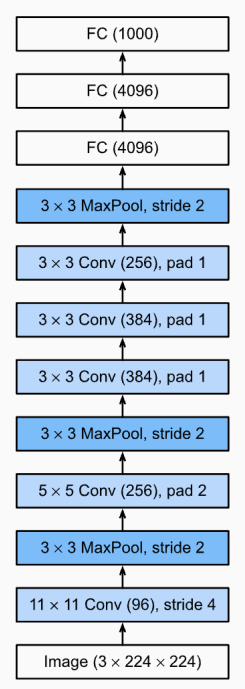

Figure 2: AlexNet Architecture

VGG, which stands for Visual Geometry Group, first introduced the idea of blocks for convolutional neural networks. The traditional building blocks for a convolutional neural network usually include a convolutional layer, a pooling layer, and a fully connected layer with an activation function. However, this kind of implementation reduces the spatial resolution of the image rapidly. A VGG block includes a sequence of 3 x 3 convolutional layers, followed by a 2 x 2 max pooling layer with a stride of 2. A VGG network consists of several

VGG blocks, followed by two fully connected layers. Figure 3 shows a general structure of of a VGG network. While AlexNet was able to produce promising results, it did not provide a general template in designing new neural networks. On the other hand, VGG was able to do so by introducing the concepts of blocks.

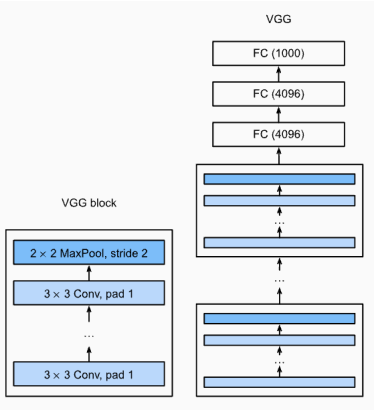

Figure 3: VGG Network Architecture

ResNet, also known as Residual Networks, integrates the idea of function classes. When we try to find a "truth" function, a larger nested function would always guarantee us to be closer to the "truth" function as it strictly increases the expressive power of our network. In this case, we can try to learn the residual function g(x) = f(x) - x instead of the true function f(x). This is easier to compute since g(x) = 0 would be the desire outcome, and we just need to train the weights and biases from the convolutional layers and fully connected layers so that the output from these layers get closer to 0. This is essentially the core mechanism for a ResNet block. Figure 4 shows the structure of a ResNet block, where it consists of two 3 x 3 convolutional layers, each followed by a batch normalization layer and a layer with ReLU activation function. An additional 1 x 1 convolutional layer can be introduced in order to transform the input into the desired shape. A batch normalization layer applies normalization to each minibatch so that each sample has the same mean and variance. A ResNet network would consist of ResNet blocks as well as other layers. Figure 5 shows a ResNet Network, where it consists of a convolutional neural network, three ResNet blocks, a pooling layer and a fully connected layer.

DenseNet, a variation of ResNet, is another popular variation model of CNN. Instead of decomposing a function f(x) into x + g(x) from ResNet, DenseNet simply oncatenates the output. A DenseNet model consists of Dense blocks that consist of multiple convolution blocks. After going through each block, the input and output are concatenate instead of added.

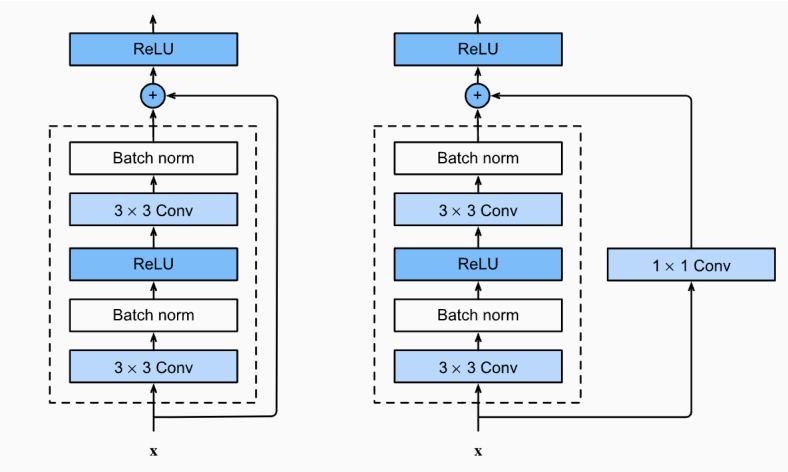

Figure 4: ResNet Block

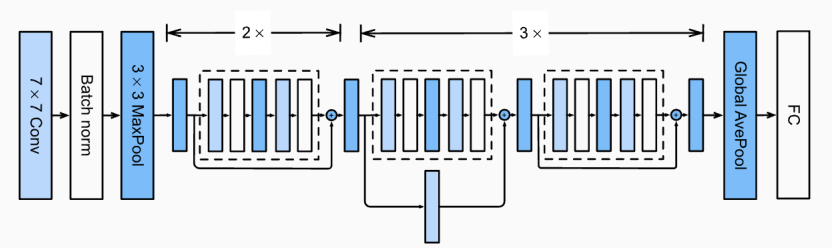

Figure 5: ResNet Network

For the task of biomedical image segmentation, UNet is one of the most popular models. UNet is a convolutional neural network designed for image segmentation but also widely applied for image analysis tasks. Figure 6 shows a general structure of the UNet. A UNet model usually consists of a contracting path and an expansive path, as they combine for a U-shaped architecture, which can be seen in the picture. The contracting path is called the encoder and the expansive path is called the decoder. The contracting path is responsible for feature extraction while the expansive path aims to recover the original resolution of the image. In addition, the encoders and decoders are connected at multiple levels through a mechanism called skip connections. These skip connections allow the decoder to access feature maps from the contracting path, preserving spatial information and fine details (Ehab et al.).

After predictions are made, a model can be assessed based on different metrics. For classification problems, one can usually look at a confusion matrix. Figure 7 shows a typical confusion matrix. From the table, several important metrics

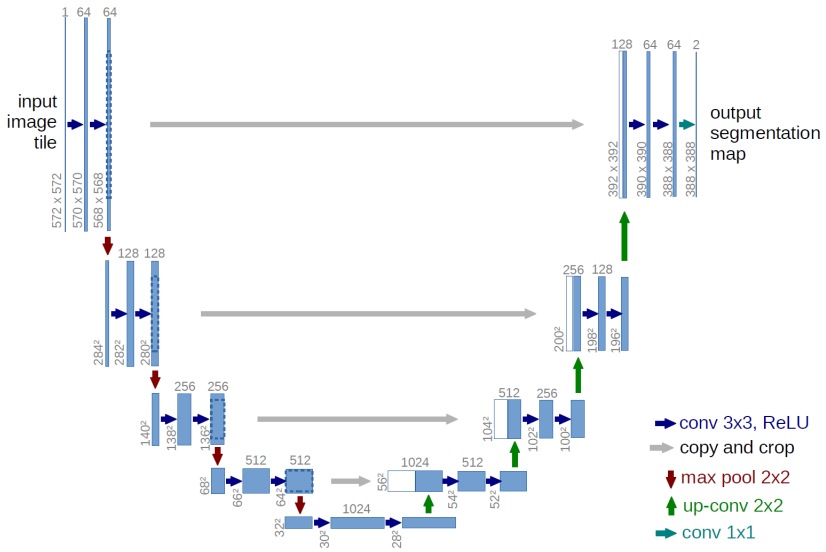

Figure 6: UNet Architecture

can be derived:

- Accuracy = $\frac{TP+TN}{TP+FP+FN+TN}$
- Precision = $\frac{TP}{TP+FP}$
- Recall = $\frac{TP}{TP+FN}$
- Sensitivity = $\frac{TN}{TN+FP}$
- F1 score = $\frac{2\times Precision\times Recall}{Precision+Recall}$
- Receiver Operating Characteristic (ROC) Analysis: plots the true positive rate against the false positive rate and the area under the curve (AUC) indicates the probability that a prediction will be correct

A popular metric that is derived from the metrics above is the DICE coefficient, where:

$$DICE = \frac{2N_{TP}}{2N_{TP}+N_{FP}+N_{FN}}$$

where N represents the number of samples in each category.

DICE coefficient is a metric commonly used in image segmentation. It measures the similarity of two sets, where a higher DICE coefficient represents a higher similarity between two sets, and vice versa. In the case of image segmentation,

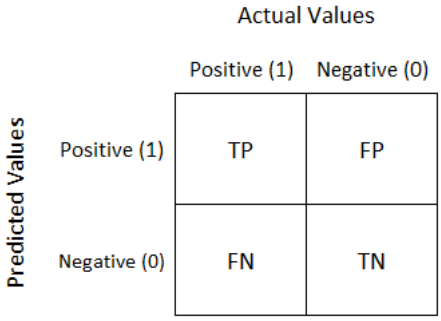

Figure 7: Confusion Matrix

we want a higher DICE coefficient as we want the prediction to be as close to the original image as possible.

1.2 Literature Review

There has been an extensive use of modern machine learning models in the medical field for X ray images. Models with different architectures have been deployed for both classification and segmentation tasks. The following section presents some previous work for both tasks.

1.2.1 Classification

Blumenfeld et al. developed a convolutional neural network that consists of 4 convolutional blocks, where each block consists of a convolutional layer followed by a leaky ReLU activation function and a max pooling layer. Two fully connected layers were added to the network and softmax function was used for the classification. The convolutions of size 5 x 5 were used for the first two blocks while the ones of size 3 x 3 were used for the last two blocks. The model was then evaluated and achieved 0.95 for the ROC analysis, which means that the prediction would be correct 95% of the time.

Sarwinda et al. applied ResNet-18 and ResNet-50 in order to do an image classification for detection of colorectal cancer. The numbers represent the number of layers in the model, so ResNet-18 has 18 layers and ResNet-50 has 50 layers. ResNet-18 is a ResNet model where each ResNet block consists of two layers, while each block consists of three layers for ResNet-50. The authors were able to achieve an accuracy of 85% and 88% with ResNet-18 and ResNet-50 respectively.

Zhong et al. applied a DenseNet model for the classification of cancer images. More specifically, the authors applied a DenseNet-201 model and a variation of the model and they were able to achieve an accuracy of above 98% with both

models.

Musaev et al. was able to create an ensemble model which combines multiple CNN models (including CNN, ResNet and other variations of CNN models), and achieve an accuracy of 99.67% for the classification of Malaria cell images.

1.2.2 Segmentation

On the other hand, UNet is one of the most popular models used for the task of image segmentation. Wang et al. designed a U-Net model for both pneumothorax classification and segmentation. The model consists of a U-shaped architecture, where an encoder and a decoder part are both included. ResNet and some other variations of the model were used as the backbone of the networks. Figure 8 shows the architecture of the networks that were used for this experiment.

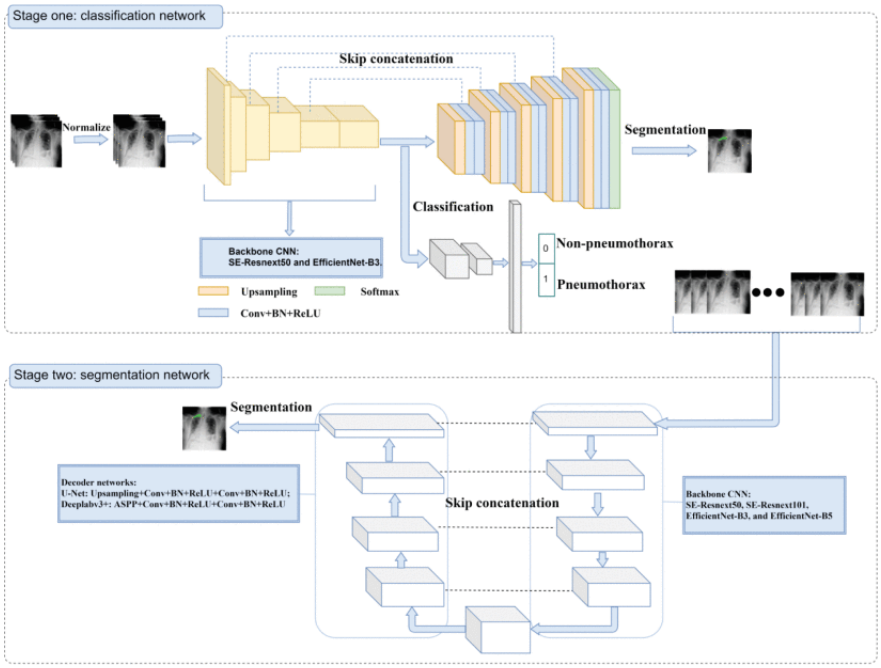

Figure 8: Stage One and Stage Two Architectures

The loss function for the training process in the first stage (classification) includes a binary cross entropy loss (L_{BCE}) and a focal loss (L_{FL}), where both are added to find the total loss. Binary cross entropy loss and focal loss are in the form of:

$$L_{BCE} = -\sum_{i=1}^{n} y_i \cdot log\hat{y}_i + (1 - y_i) \cdot log(1 - \hat{y}_i)$$

$$L_{FL} = -\sum_{i=1}^{n} [(1 - \hat{y}_i)^{\gamma} \cdot y_i \cdot log(y_i) + (\hat{y}_i)^{\gamma} \cdot (1 - y_i) \cdot log(1 - y_i)]$$

The dice function was used as the loss function for the training process in the second stage (segmentation), where:

$$L_{dice} = -\frac{2 \cdot \sum_{i=1}^{n} y_i \cdot \hat{y}_i}{\sum_{i=1}^{n} y_i^2 + \sum_{i=1}^{n} \hat{y_i}^2}$$

For model evaluation, AUC, accuracy, precision, recall and F1-score were used for stage one models, and the Dice coefficient was used for models from both stages, where they were able to achieve a DICE coefficient of 0.8883 and 0.8665 for stage one and stage two respectively.

Rahman et al. implemented the original UNet model and a variant of the UNet model in order to segment X ray images with tuberculosis. The original UNet model consists of a contracting path and an expanding path. The contracting path consists of the repeated application of two 3 x 3 convolutions, each followed by a ReLU and a 2 x 2 max pooling operation with stride 2 for down sampling. The expanding path consists of an upsampling of the feature map followed by a 2×2 convolution that halves the number of feature channels, a concatenation with the correspondingly cropped feature map from the contracting path, and two 3 x 3 convolutions, each followed by a ReLU. While the modified U-Net also consists of a contracting path and an expanding path as U-Net. The contracting path includes four steps. Each step consists of two convolutional 3 x 3 filters followed by a 2 x 2 max pooling function and ReLU. The U-Net might learn redundant features in successive convolution layers. However, modified UNet uses densely connective convolutions to mitigate this problem. Each step in the expanding path starts by performing an up-sampling function over the output of the previous layer. In the modified UNet, the corresponding feature maps in the contracting path are cropped and copied to the expanding path. These feature maps are then concatenated with the output of the up-sampling function. Instead of a simple concatenation in the skip connection of U-Net, Modified UNet employs bidirectional Convolutional Long Short Term Memory (BConvLSTM) to combine the feature maps extracted from the corresponding contracting path and the previous expanding up-convolutional layer in a non-linear way.

1.3 Objective

The objective of this project aims to develop a deep learning model that is able to classify pneumothorax from a set of chest radiographic images. Furthermore, if pneumothorax is present in an image, the model would aim to segment the image so that the location of the pneumothorax can be clearly identified.

1.4 Machine Learning Pipeline

The machine learning pipeline for this project will follow a traditional data science pipeline, where it would consist of three big parts: data processing, data modeling, and prediction results with conclusion. More specifically, data processing will include data cleaning and wrangling, data visualization, as well as feature engineering if necessary. The modeling section will consist of two sections, one for classification and one for segmentation. More specifically, three models will be used for classification, a CNN model, a pre-trained ResNet model, and a pre-trained DenseNet model; for the segmentation section, two models will be applied which are both UNet models but with different encoders for the contracting path. Lastly, the model performance will be presented in the results section.

2 Data Processing

2.1 Data Description

The dataset was provided by "Society for Imaging Informatics in Medicine" (SIIM) and is a part of Kaggle competition, namely the SIIM-ACR Pneumothorax Segmentation competition.

The dataset consists of both training and testing data. Both datasets are comprised of images in DICOM format, as well as annotations in the form of imges IDs and run-length-encoded (RLE) masks in a different CSV file. More specifically, for each image, which are uniquely identified by an "ImageID" (name of the column), are provided in a CSV file. Images with instances of pneumothorax are filled with run-length encoding in the "EncodedPixels" column in the CSV file whereas the ones not having pneumothorax are filled with "-1" in the EncodedPixels column.

A run-length-encoded mask is a form to compress the data without loss of any information. It stores a sequence of the same values in the data into a single value. For example, if we have a sequence like "aaabbc", then we can rewrite it as "a3b2c1" in the RLE form.

The training set consists of a variety of images of chest radiographs: from the ones masked with regions of occurrences pneumothoraces in different sections of the chest to the ones absent with pneumothorax. The number of pixels that are masked in each image is different, implying that the affected region may be small or large and hence, the mask percentage in the image can vary. There are in total 12089 images in the training set.

2.2 Data Wrangling

The initial stage of data wrangling including reading data and cleaning data. We first read in the CSV file called "train-rle.csv" which contains each unique image ID as well as the run-length encoding of the masks of each picture.

After loading the training data, the entire dataset consists of 12089 images of which, 42 do not have any label and hence will be removed. The number of images under consideration is thus reduced to 12047. Among the 12047 images, 9378 images have no mask and the rest, 2669 are masked. These 12047 images are the final dataset we will be using.

Since the data in "EncodedPixels" column is in run-length-encoded format, we need to convert these to a more readable format. Functions are created in order to decode the masks from the RLE format. The function is created in a way such that when we encounter a "-1" in the column, then we have no masks from

the picture; On the other hand, if the column contains something that is not "-1", that means we have masks contained for the corresponding image. We first initialized a list with a length of 1024 x 1024 that is filled with 0s. We then reshaped the list of numbers in the "EncodedPixels" column into a matrix with two columns, where the numbers in the first column would represent the starting index, while the ones in the second column would represent the ending index. We would then fill the list we created with 1 from the starting index to the ending index. At the end, we would reshape the list to a matrix of shape 1024 x 1024. The masks will now become a 1024 x 1024 matrix filled with 0s and 1s.

2.3 Exploratory Data Analysis and Visualization

After reading the CSV file, we decided to explore the image data. The image data is in DICOM format, which stands for Digital Imaging and Communication in Medicine. DICOM is a standard format for storing, transmitting, and displaying medical images and related data. A DICOM file usually contains a header and an image set. In our case, the DICOM files contain metadata including different features of the image like the date, the image ID and so on, as well as the image data in 1024 x 1024 pixels. Figure 9 shows an example of the image data in DICOM format in a gray scale.

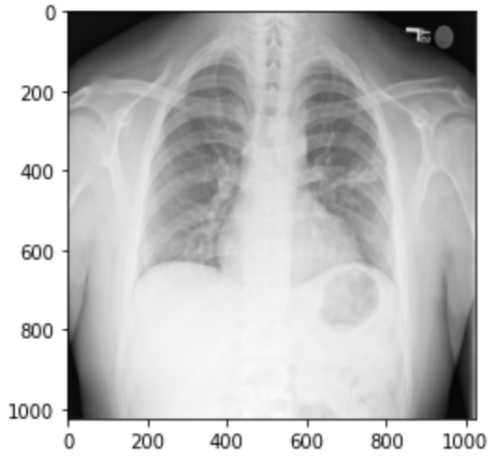

Figure 9: Example image data

The original images give us some basic information of the lungs, while the ones with the masks would give us almost all the information that we need, so we need to combine the original images and their corresponding masks based on their unique image ID. We applied the functions we created to the "Encoded-Pixels" column so that we can see the masks in a more readable way.

Figure 10 shows an example of the image with their masks included. We can see that the areas with a different colors show the masks, which represent the presence of pneumothorax in the lungs.

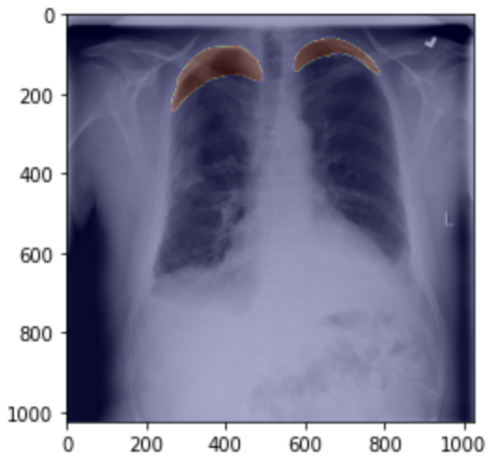

Figure 10: Example of image data with masks

3 Methods and Models

3.1 Data Loader

After exploring the data, the images are organized into different categories. The pictures with pneumothorax are organized into the positive group and the ones without pneumothorax are organized in to the negative group. The images are further divided into subgroups, namely images, names, and masks. The "images" subgroup contains the pixel array of each image; the "names" subgroup contains the image ID of each image; the "masks" subgroup contains the pixel array of the masks corresponding to each image. These files are converted to npy files for better readability in Python. This process is done for both the training set and testing set.

An initial attempt was made to save the data in a dimension of 1024 x 1024, so for the training data, we would have 9378 images with negative labels with the size of 1024 x 1024, and 2669 images with positive labels with the size of 1024 x 1024. It turned out that these images would take up too much RAM in the system, so the resolution of the images were reduced to 128 x 128 for the second attempt, in which we were able to load these images successfully.

Similar process was applied to the masks, where the images were also restructured to 128 by 128 pixels.

The data is then split into training set, validation set, and testing set. Since the original data has 9378 (∼78%) negatively labeled data and 2669 (∼22%) positively labeled data, there is a clear class imbalance between the two labels. Thus, the process of splitting the dataset is stratified, which preserves the proportion of classes in both training and testing set. This means that both training set and testing set will still contain around 78% of negatively labeled data and 22% of positively labeled data. Figure 11 shows the distribution of the data in training, validation, and testing set. We can see that all 3 datasets have the same distribution between the images with and without pneumothorax.

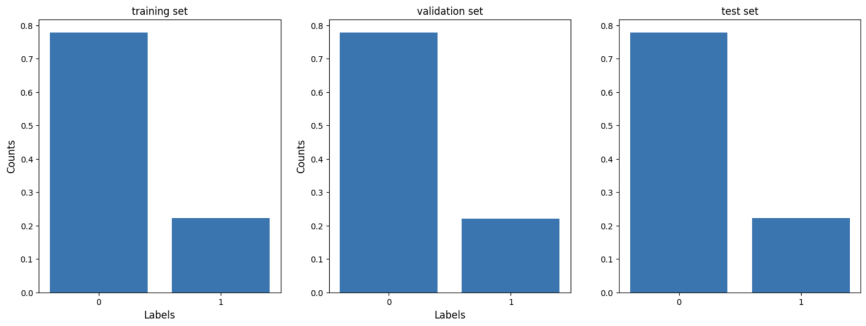

Figure 11: Distributon of data

3.2 Classification

As previously mentioned, three models are created for the task of classification: a CNN model, a ResNet model, and a DenseNet model. The CNN model will be trained, while the ResNet model and the DenseNet model will be pre-trained.

The CNN model serves as a baseline model. It consists of a basic convolutional neural network architecture. More specifically, the network consists of CBR blocks, where each block contains a convolutional layer, a batch normalization layer, and a Leaky ReLU activation function. A dropout layer is also added to each block in order to prevent overfitting. Pooling layers are incorporated between each blocks in order to reduce dimensionality. A fully connected layer is added at the end for the task of classification.

The CNN model created consists of 4 CBR blocks, the size of the input image is originally set to 224 by 224. Then at each block, the number of filters are 32, 64, 128, 256 at each block respectively. Models were experimented with the size of the filter set to 3, 5, and 7, and it turns out a filer with size 7 achieves the best model performance. Thus, a filter of size 7 x 7 is applied to the images. Stride is set to 2 and there is no padding for the model.

Different values of learning rates, weight decay, and batch sizes are experimented in the training stage, which were all ran on 10 epochs. The ones that resulted in the best AUC value were chosen as the final model. In this case, a learning rate of 5e-4, weight decay of 1e-3, and a batch size of 64 were chosen. The model was then trained on 40 epochs and the AUC for each epoch was recorded. The parameters that correspond to the highest AUC among the 40 epochs was set as the parameters for the final model.

For the ResNet model, a pre-trained model was used. Since the CNN model created was able to achieve a relatively good performance with around 5 layers and around 2 million total parameters, ResNet-18, a relatively simpler ResNet model, was chosen as the second model. ResNet-18 has 18 layers and around 11 million parameters.

The third model chosen is the DenseNet model. DenseNet-121 was the pre-trained model that was chosen since it has 6 million parameters, which seems to be an appropriate amount looking at the first two models.

3.3 Segmentation

For the sake of this task, only images with pneumothorax were included in the dataset for this task. We would like to evaluate on a model solely based on its performance on the task of segmentation, which is why the ones without pneumothorax are excluded. This means that we have a total of 2669 images in the final dataset for this task. The data will be split into training, validation,

and testing set in the same fashion as before.

UNet model [1] is considered one of the most popular and state-of-the-art models for segmentation of the biomedical images because of its unique architecture. Two UNet model will be created and evaluated for our task of segmentation. Both UNet models will use the pre-trained weights.

Since ResNet-18 was used for the classification task, it became a natural choice for the encoder of the first UNet model. More specifically, the model used ResNet18 as the encoder and imagenet data for its pre-trained weights and parameters. The model has around 24 million parameters.

DenseNet-121 was the original choice for the encoder of the second UNet model. However, due to licensing issues, the model was not being able to be used, so VGG-13 was chosen as the encoder for the second UNet model. The model also used imagenet to pre-train its weights and parameters, and it has around 9 million parameters.

Both models were trained on 10 epochs, the best parameters are then chosen. Both models are then evaluated on the test set. The model performance is evaluated on an average dice loss on the testing set, where we compute the total dice loss for all samples in the test set, and divide it by the number of total samples in the test set.

[1] The UNet models used for this section come from a Github repo created by Iakubovskii

4 Results

4.1 Classification

The baseline CNN model is first applied to the training and validation set. Figure 12 and 13 show the loss and accuracy for both training set and validation set respectively. The loss for the training set continued to decrease, and the accuracy for the training set continued to increase through 40 epochs. On the other hand, it appears that both the loss and the accuracy for the validation set are converging. This shows that the model was mostly not overfitting. The final model was run on the testing set and was able to achieve an accuracy of 82.3%. Figure 14 shows the confusion matrix after model made the predictions.

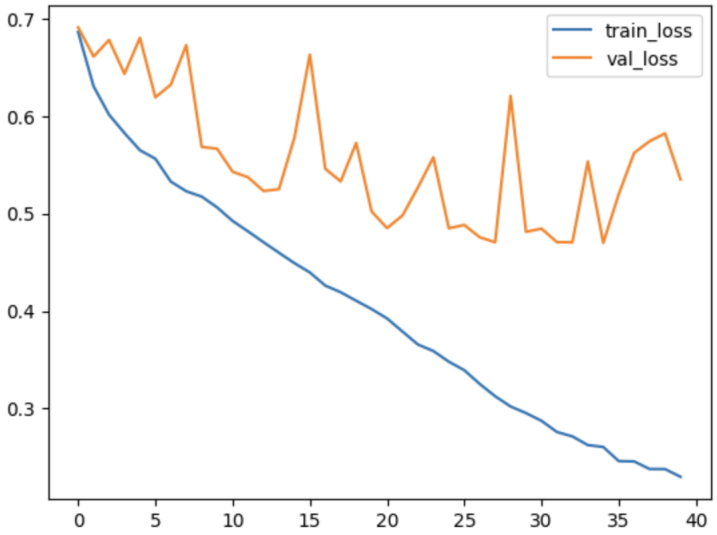

Figure 12: Training and Validation Loss for CNN

The second model was the ResNet-18 model. We ran 10 epochs on the pretrained model and used the model with the most optimal weights for the testing set. Figure 15 and 16 show the loss and accuracy for training and validation set. We see that both loss and accuracy seem to stay roughly the same throughout the training and validation set. Figure 17 shows the confusion matrix with predictions made on the test set. It appears that the model predicted most of the images to have no pneumothorax, possibly due to the imbalance in training and validation set (which should be addressed in future work).

The last model is the DenseNet model. Again we show the loss and accuracy for training and validation set from figure 18 and 19. There doesn't seem to be much of a decreasing trend in validation loss, but the validation accuracy shows

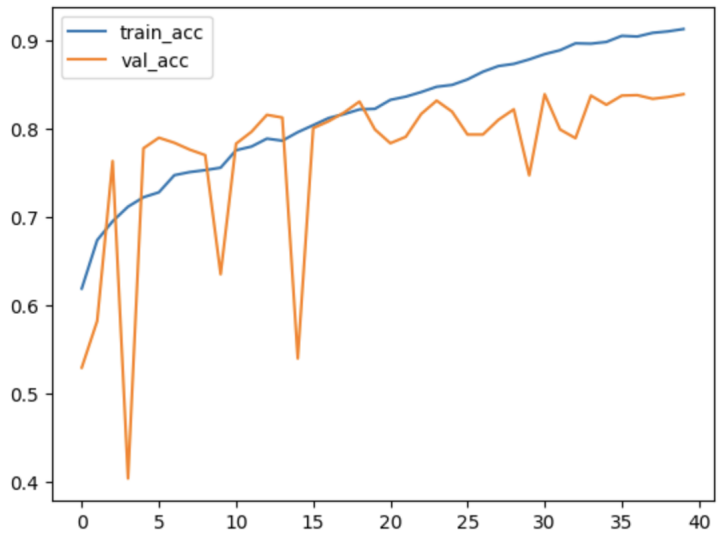

Figure 13: Training and Validation Accuracy for CNN

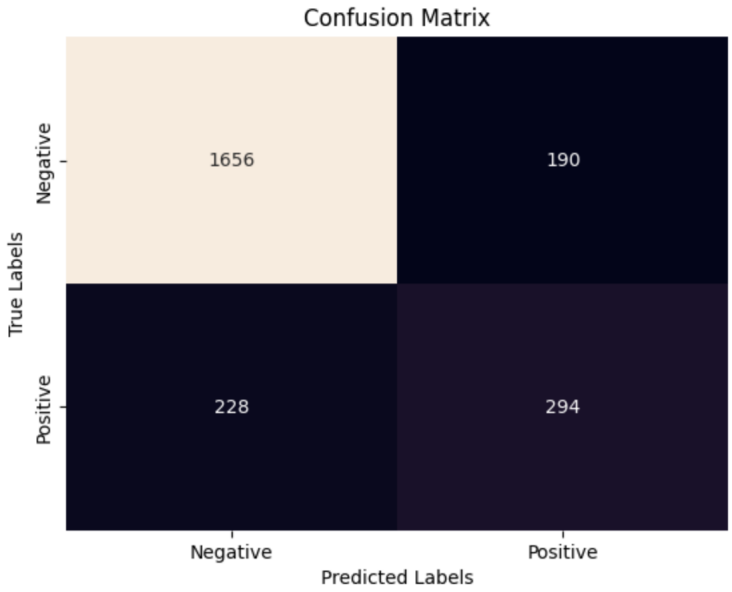

Figure 14: Confusion Matrix for CNN

a slightly increasing trend. Furthermore, figure 20 shows the confusion matrix for this model.

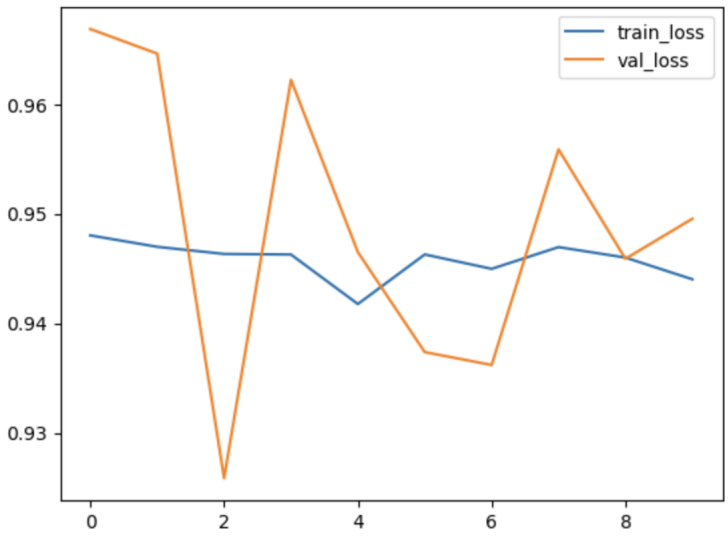

Figure 15: Training and Validation Loss for ResNet

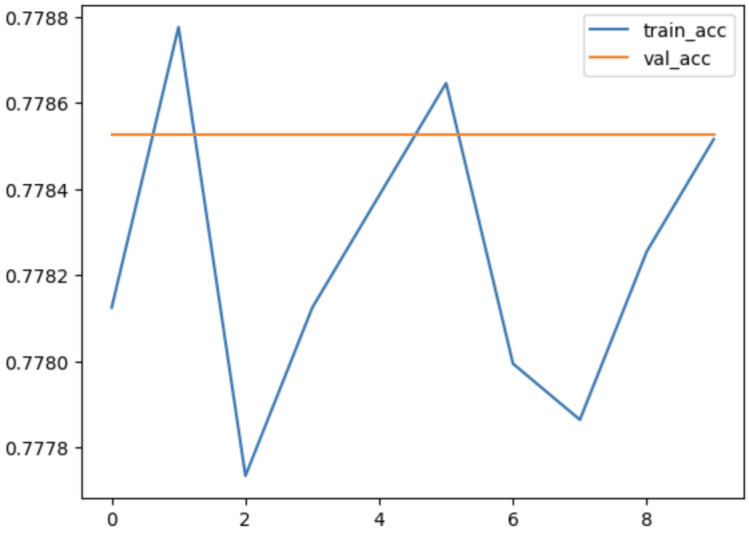

Figure 16: Training and Validation Accuracy for ResNet

Lastly, we present the accuracies that can be used to assess the model performance in table, since there is a large number of negatively labeled data, certain metrics may not be able to accurately reflect the true model performance. The CNN model has an accuracy of **82.3%**; ResNet-18 has an accuracy of **76.4%**; DenseNet-121 has an accuracy of **82.6%**. DenseNet was able to achieve the

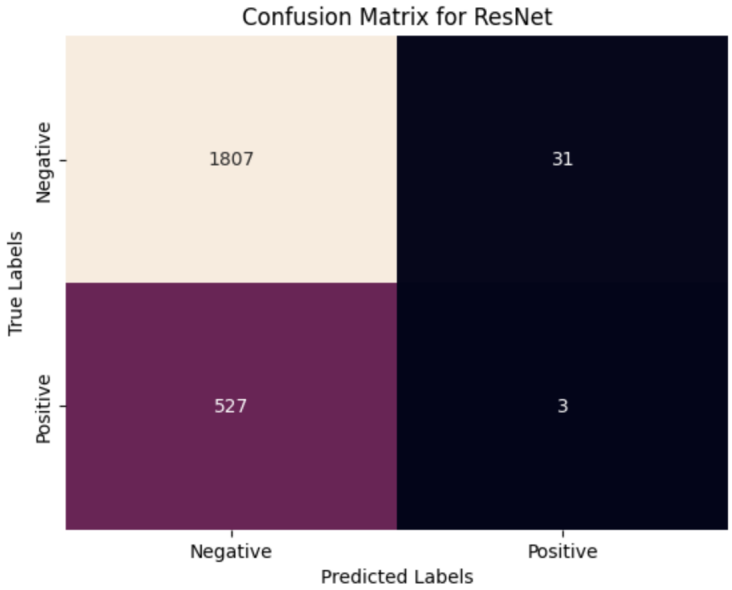

Figure 17: Confusion Matrix for ResNet

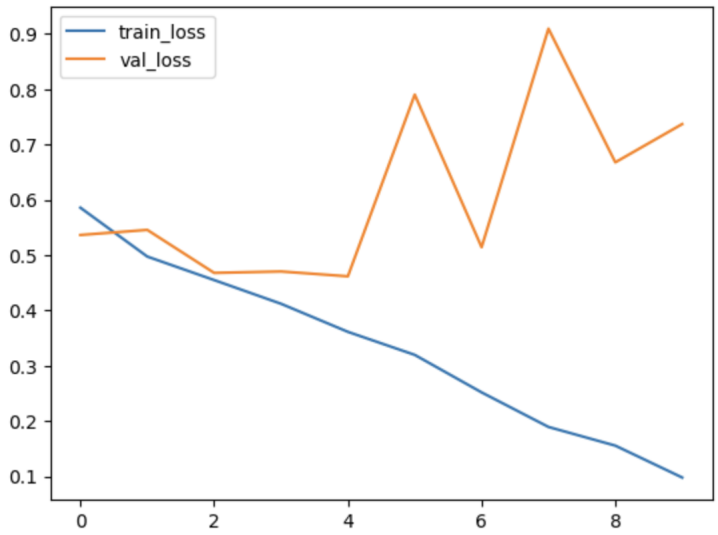

Figure 18: Training and Validation Loss for DenseNet

highest accuracy among the three models.

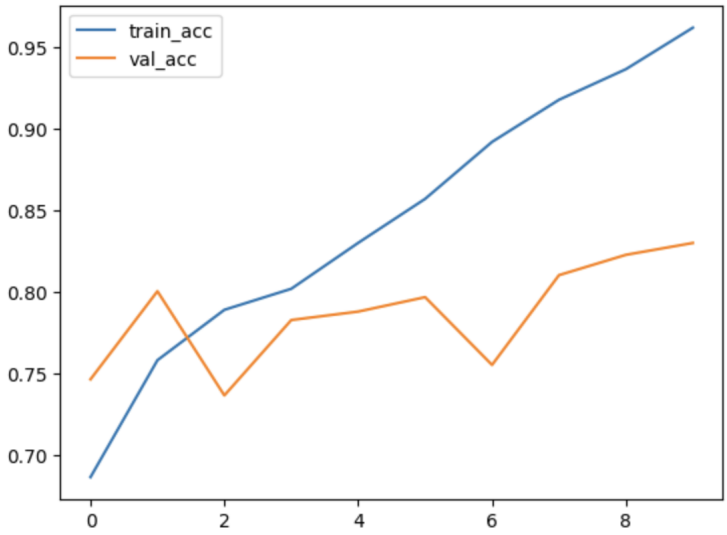

Figure 19: Training and Validation Accuracy for DenseNet

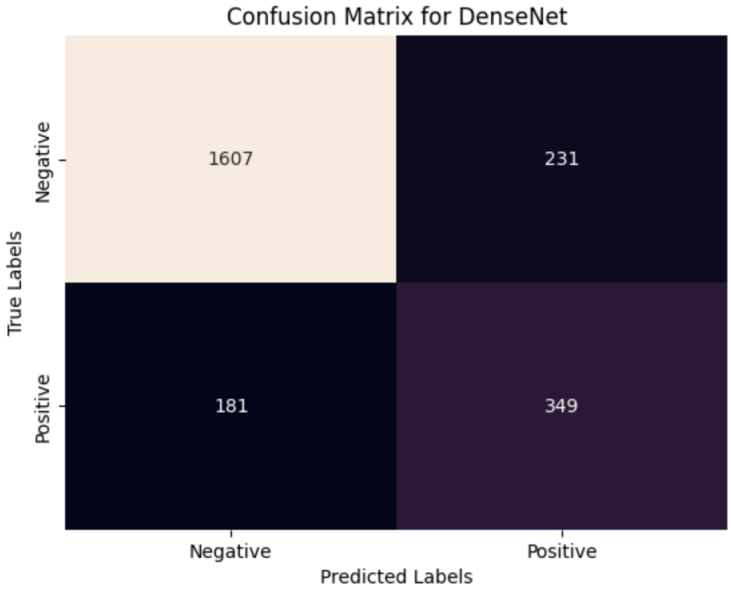

Figure 20: Confusion Matrix for DenseNet

4.2 Segmentation

As previously mentioned, two UNet models were implemented for the task of segmentation. The first one uses ResNet-18 as the encoder and the second one

uses VGG-13 as the encoder. Dice loss was used as the key metric for model performance on segmentation, where the loss is computed as each epoch.

Figure 21 shows the dice loss for each epoch during the training and validation set for the first model (ResNet-18). There is a steady decrease in the validation loss and it gradually converges after 10 epochs.

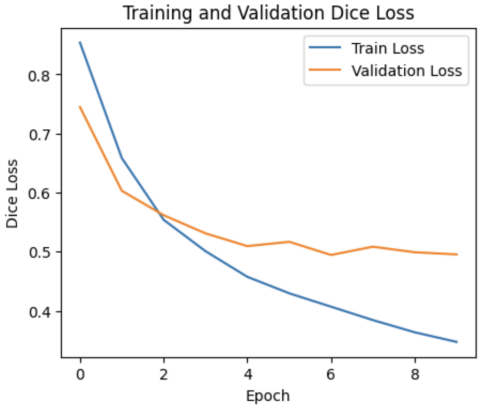

Figure 21: Training and Validation Dice Loss

The model is then run on the test set. Figure 22 shows the original masks and the predictions from the model for the first 3 samples in the test set. We see that the model was able to prediction the relative position of the masks, but can have some discrepancies in the sizes.

Similarly, we show the dice loss from training and validation set for the second model (VGG-13) in figure 23. The loss shows a similar trend as the first one, where it gradually decreases and converges after 10 epochs.

Figure 24 shows the original masks and predictions for the first 3 samples in the test set from the second model Similar to the first model, this model is able to predict the relative position of the masks, but with some differences in the sizes.

Lastly, we present the average dice loss on the test set from both models. The first model (ResNet-18) produced an average dice loss of **0.499** on the test set, while the second model (VGG-13) produced an average dice loss of **0.50** for the test set. While both models have very similar performance, the UNet with ResNet-18 as its encoder was able to produce a slightly lower dice loss for the test set.

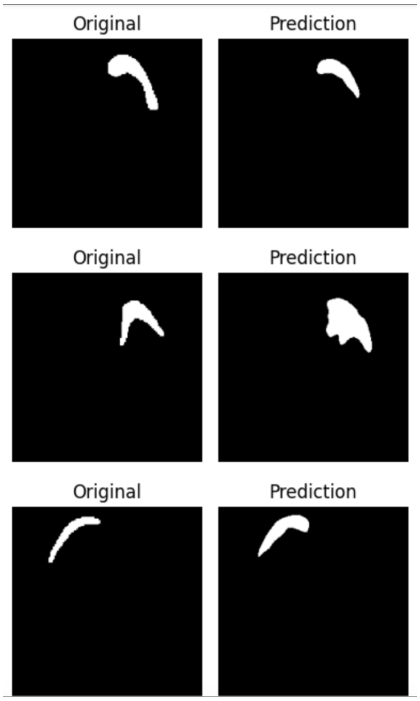

Figure 22: First three masks and predictions

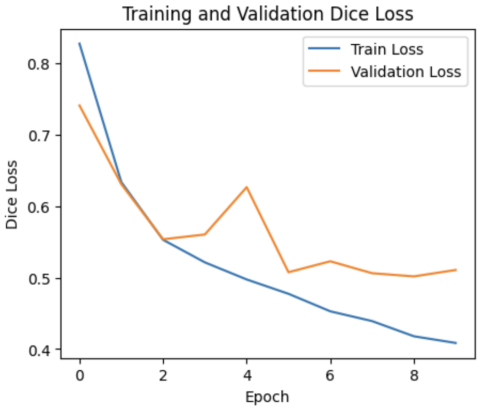

Figure 23: Training and Validation Dice Loss

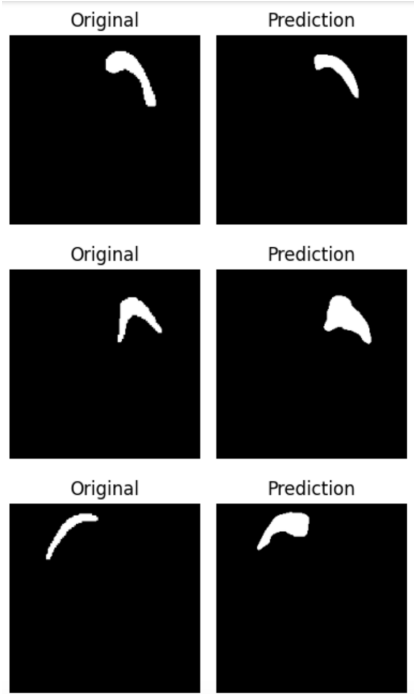

Figure 24: First three masks and predictions

5 Conclusion

This report outlines a systematic machine learning pipeline of classification and segmentation on X ray images of pneumothorax. The report first explains the motivation of the project, it then describes the process of data cleaning and data wrangling. Three models were evaluated for the task of classification, and two other models were created for the task of segmentation.

For the task of classification, we showed that the DenseNet model has the best performance, achieving an accuracy of 82.6%. For the task of segmentation, a UNet model with ResNet-18 as its encoder was able to achieve the best performance, producing a dice loss of 0.499.

Future work is needed to refine the models in order to improve the performance. It would also be helpful to have more images in order to have a bigger sample size. Nevertheless, future work should be conducted since the direction of this project has the potential to revolutionize the medical field by aiding radiologists and healthcare providers in the early detection of pneumothorax, reducing diagnostic time, and improving patient outcomes. With further development, such models could be integrated into hospital systems, offering real-time decision support and significantly enhancing the speed and accuracy of diagnoses. Ultimately, the continued advancement of AI in healthcare has the potential to not only streamline processes but also save lives, making medical interventions more efficient, accessible, and equitable.

References

Aviel Blumenfeld, Eli Konen, and Hayit Greenspan. Pneumothorax detection in chest radiographs using convolutional neural networks. In *Medical Imaging 2018: Computer-Aided Diagnosis*, volume 10575, pages 15–20. SPIE, 2018.

Walid Ehab, Lina Huang, and Yongmin Li. Unet and variants for medical image segmentation. 2024.

Pavel Iakubovskii. Segmentation models pytorch. `https://github.com/qubvel/segmentation_models.pytorch`, 2019.

Javokhir Musaev, Abdulaziz Anorboev, Yeong-Seok Seo, Ngoc Thanh Nguyen, and Dosam Hwang. Icnn-ensemble: An improved convolutional neural network ensemble model for medical image classification. *IEEE Access*, 2023.

Tawsifur Rahman, Amith Khandakar, Muhammad Abdul Kadir, Khandaker Rejaul Islam, Khandakar F Islam, Rashid Mazhar, Tahir Hamid, Mohammad Tariqul Islam, Saad Kashem, Zaid Bin Mahbub, et al. Reliable tuberculosis detection using chest x-ray with deep learning, segmentation and visualization. *Ieee Access*, 8:191586–191601, 2020.

Devvi Sarwinda, Radifa Hilya Paradisa, Alhadi Bustamam, and Pinkie Anggia. Deep learning in image classification using residual network (resnet) variants for detection of colorectal cancer. *Procedia Computer Science*, 179:423–431, 2021.

Xiyue Wang, Sen Yang, Jun Lan, Yuqi Fang, Jianhui He, Minghui Wang, Jing Zhang, and Xiao Han. Automatic segmentation of pneumothorax in chest radiographs based on a two-stage deep learning method. *IEEE Transactions on Cognitive and Developmental Systems*, 14(1):205–218, 2022. doi: 10.1109/TCDS.2020.3035572.

Ziliang Zhong, Muhang Zheng, Huafeng Mai, Jianan Zhao, and Xinyi Liu. Cancer image classification based on densenet model. In *Journal of physics: conference series*, volume 1651, page 012143. IOP Publishing, 2020.