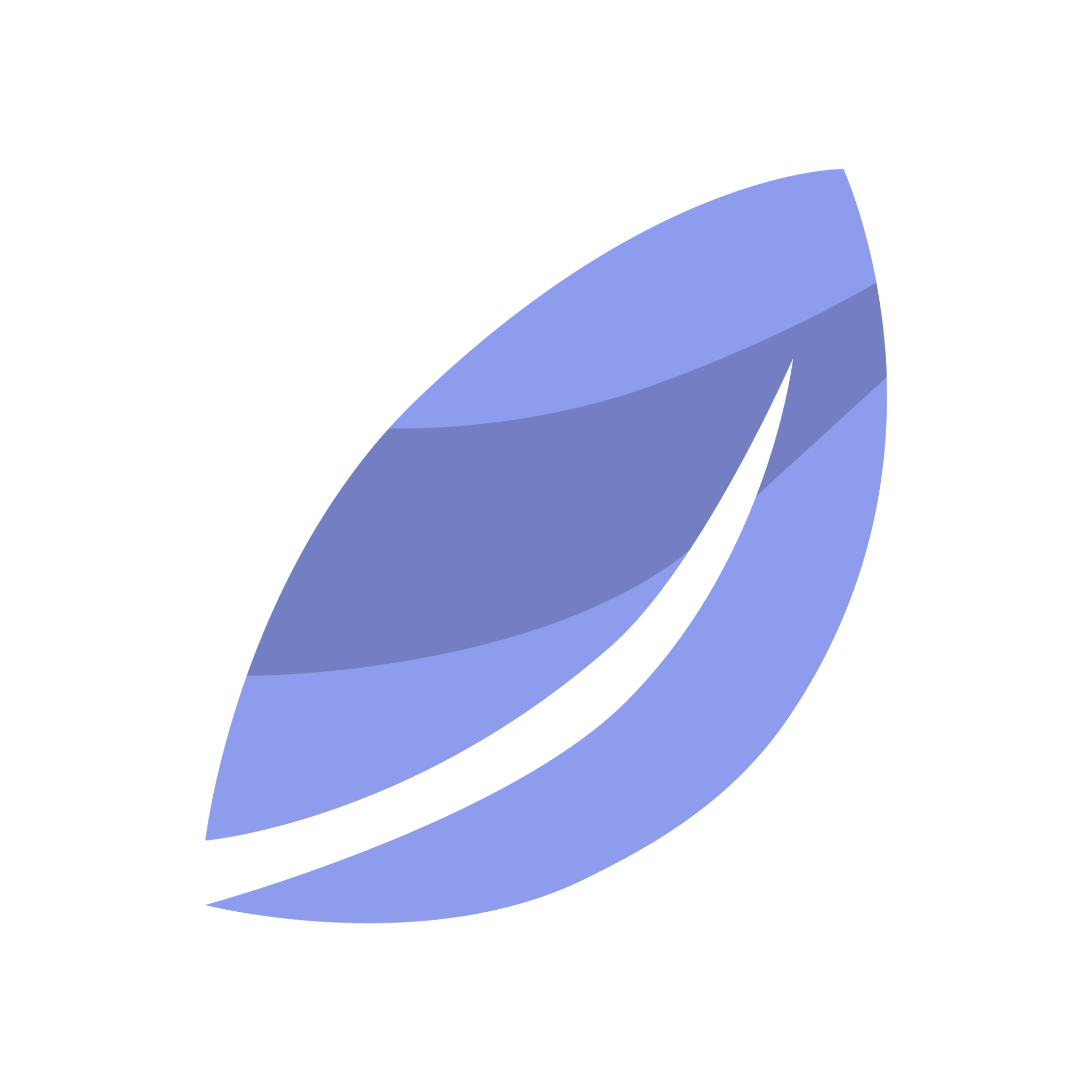
Pneumothorax Detection and Classification in Chest X-rays

Rachis Systems

# Mr.Yaser Zaidan

25 / 9 / 2024



### Technical Report.

[Introduction](#_6poptih5bzu9)

[Problem statement & Objective](#_3ez4mfia6dsq)

[● Problem statement](#_eiz6qiz9f7gk)

[● Objective](#_srbfz0ifj0ot)

[Methods](#_sb2xn6233s45)

[● Data Preprocessing and Augmentation](#_ttt03dbbl7wu)

[● CNN Implementation](#_iddgdvjo0a0b)

[● Model Architecture](#_8qr441bpq1rp)

[● Loss Function](#_5assxpv5fytn)

[● Optimizer and Scheduler](#_tojexceintdi)

[● Training the Model:](#_prjyaiejh2h4)

[● Training Loop](#_tqr3qy7meuni)

[● Early Stopping](#_aih88bhwbwmx)

[● Metrics](#_m84hryfnwp6r)

[○ Accuracy](#_ptguv9l19oi4)

[○ Intersection over Union (IoU)](#_a03vts8srbmy)

[● Train Configuration](#_8l5ebpno6k3a)

[● CNN Enhancement](#_yoaqa9sd68es)

[● Classification](#_jo158bkhanwu)

[Experimental Results for Segmentation](#_6uqsvv38nl6o)

[● Unet with Resnet34](#_w372tazh75i)

[Learning curves](#_z3rlr5euhujt)

[Example predictions](#_ql6hlc548i33)

[● Unet with VGG16](#_lujjll3x7c43)

[Learning curves](#_5k3mhs20p6lb)

[Example predictions](#_d1rapum8fdeu)

[● Unet with InceptionResnetV2](#_tfvjov8j7lq2)

[Learning curve](#_qj0j2yii0ery)

[Example predictions](#_j5ky5mxjsre0)

[● Unet with Xception](#_qiu6zg1c7459)

[Learning curve](#_2hyu3alciklp)

[Example predictions](#_5a1ykxmpzdjh)

[Comparison](#_gq7ov0wben2x)

[Final Thoughts on Segmentation Phase.](#_jmstp72rp7re)

[Experimental Results for Classification](#_7x4vvaojzfyz)

[Before hyperparameter tuning](#_ad9ucwpa48fs)

[After hyperparameter tuning](#_upou8m9y3idv)

[Final Thoughts on Classification Phase.](#_esdpdi3wxw1t)

# Introduction

Pneumothorax is a critical condition that requires timely detection and immediate action. It occurs when air collects in the pleural space between the lung and the chest wall, potentially leading to lung collapse. Early detection is crucial to prevent significant morbidity or patient death,

Chest X-ray (CXR) imaging is the primary diagnostic imaging technique for the diagnosis of pneumothorax.

A computerized diagnosis system can detect pneumothorax in chest radiographic images, which provide substantial benefits in disease diagnosis. In the present work, a deep learning neural network model is proposed to detect the regions of pneumothoraces in the chest X-ray images.

# Problem statement & Objective

## Problem statement

As we’ve mentioned, early detection is crucial for preventing severe complications or death. Traditional methods of detecting pneumothorax through chest X-rays require expert radiologists, which can be resource-intensive and slow, particularly in emergency scenarios. With the advent of deep learning technologies, automating the detection of pneumothorax in chest X-rays presents a promising solution to alleviate this challenge. However, developing an accurate and efficient model to detect this condition in medical images remains a complex task due to the variability in X-ray images and the imbalanced nature of medical data.

## Objective

The objective of this project is to design and implement a deep learning-based convolutional neural network (CNN) to automatically detect pneumothorax in chest X-ray images. This model will take X-ray images as input and classify whether the condition is present or absent. The primary goal is to achieve high accuracy while maintaining clinical relevance by optimizing model performance in terms of sensitivity, precision, and other key metrics. This project will involve data preprocessing, augmentation, CNN implementation, and enhancements to improve the model’s generalizability and robustness in detecting pneumothorax from diverse image data.

# Methods

## Data Preprocessing and Augmentation

A custom PyTorch Dataset class, PneumothoraxDataset, is implemented to read images and corresponding masks. The \_\_getitem\_\_ method loads an image and its corresponding mask by file name, converting them to PyTorch tensors.

1. Splitting

The dataset is split into training, validation, and test sets using a 70/15/15 split. The split ensures that images from the same patient remain in the same subset to avoid data leakage.

1. Loading

A Pytorch DataLoader is implemented for each dataset to load the data batch by batch when training to reduce memory usage

1. Augmentation

* Images are resized to a consistent size of 512x512 pixels.
* Augmentations such as random rotation (±15°), random horizontal flipping, and brightness/contrast adjustments are applied to enhance model generalization.
* The data is also normalized using a mean of 0.5 and a standard deviation of 0.5.
* Input images are converted to 3 channels as required by the ResNet CNN pretrained model

## CNN Implementation

### Model Architecture

A U-Net architecture is created using the segmentation\_models\_pytorch library with a pre-trained encoder. In this case, ResNet-34 is used as the backbone (with pre-trained ImageNet weights), making it suitable for medical image segmentation.

### Loss Function

A custom combined loss function is defined that integrates:

* + Binary Cross Entropy (BCE) loss.
  + Dice loss, which is useful for dealing with segmentation tasks.
  + Focal loss, which helps in addressing the class imbalance problem.

### Optimizer and Scheduler

The Adam optimizer is used with a learning rate of 0.001, and a learning rate scheduler (ReduceLROnPlateau) reduces the learning rate if validation performance plateaus.

### Training the Model:

#### Training Loop

The model is trained for a defined number of epochs, iterating over the training data. In each epoch:

1. Images and masks are passed through the model.
2. The combined loss is calculated and backpropagated to update the model’s weights.
3. After each epoch, the model is validated on the validation set.

#### Early Stopping

If validation performance stops improving, early stopping is triggered to avoid overfitting.

#### Metrics

During training, performance is evaluated based on:

##### Accuracy

Proportion of correct predictions.

##### Intersection over Union (IoU)

Measures the overlap between predicted and actual masks.

### Train Configuration

| Image Dimensions | | 512 x 512 |
| --- | --- | --- |
| Image Channels | | 3 |
| Mask Channels | | 1 |
| Batch Size | | 32 |
| Learning Rate | | 1e-3 |
| Device | | cuda |
| Epochs | | 25 |
| Normalization | STD | 0.5 |
| Mean | 0.5 |
| Focal Loss | alpha | 0.8 |
| gamma | 2 |
| Dice Loss | smooth | 1e-5 |
| Combined Loss | alpha | 0.5 |
| beta | 0.3 |
| gamma | 0.2 |
| IOU metric | threshold | 0.5 |
| smooth | 1e-5 |
| Early Stopping | patience | 5 |
| Reduce On Plateau | factor | 0.5 |
| patience | 3 |

## CNN Enhancement

The model is enhanced by experimenting with different encoders such as InceptionResNetV2, Xception, and Vgg16. also implementing a combined loss function, These experiments aim to improve performance by leveraging different architectures pre-trained on ImageNet.

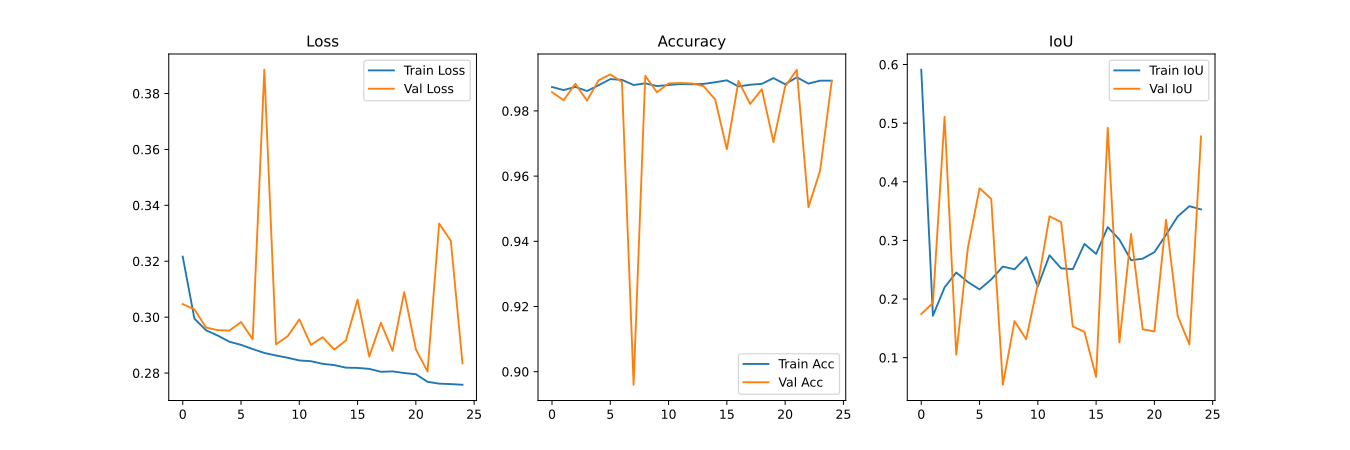
## Classification

After segmentation the images and their predicted masks are combined together and passed to a pretrained densenet model to extract features and then passed to XGBoost classifier along with the labels to do the final classification then the classifier is enhanced by implementing hyperparameter tuning using random search method.

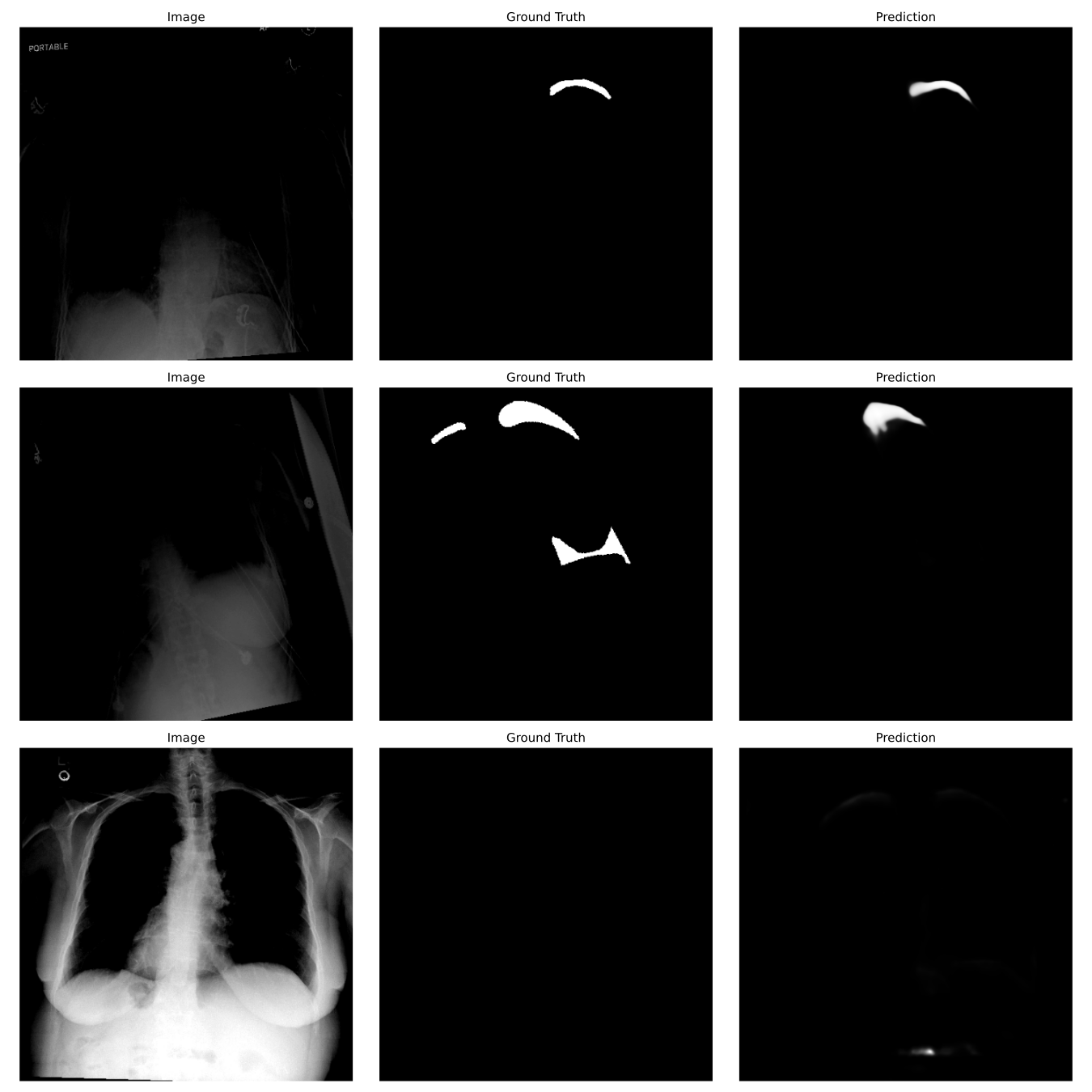
# Experimental Results for Segmentation

## Unet with Resnet34

### Learning curves

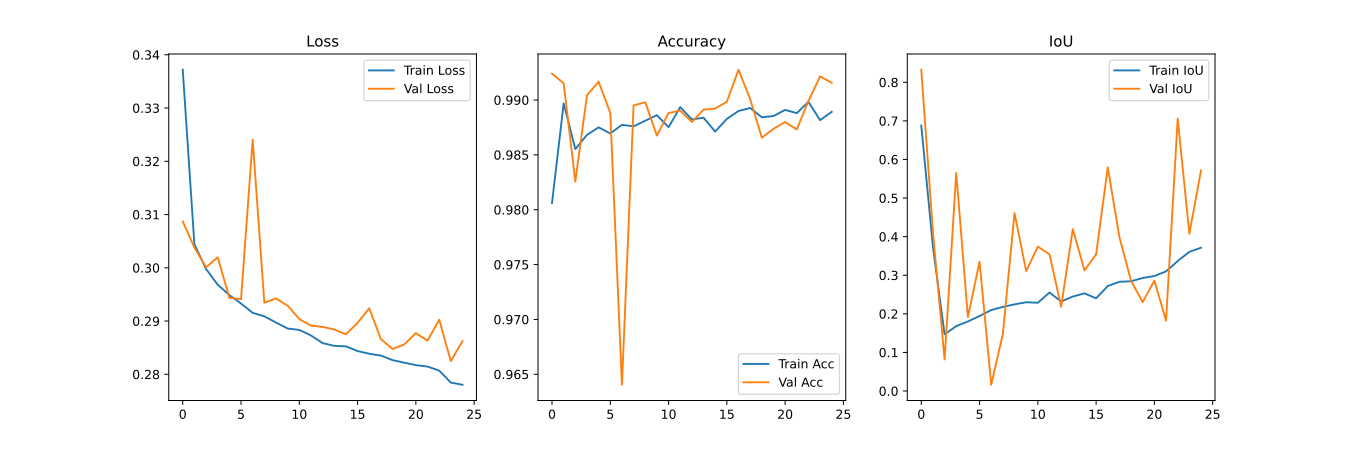


### Example predictions

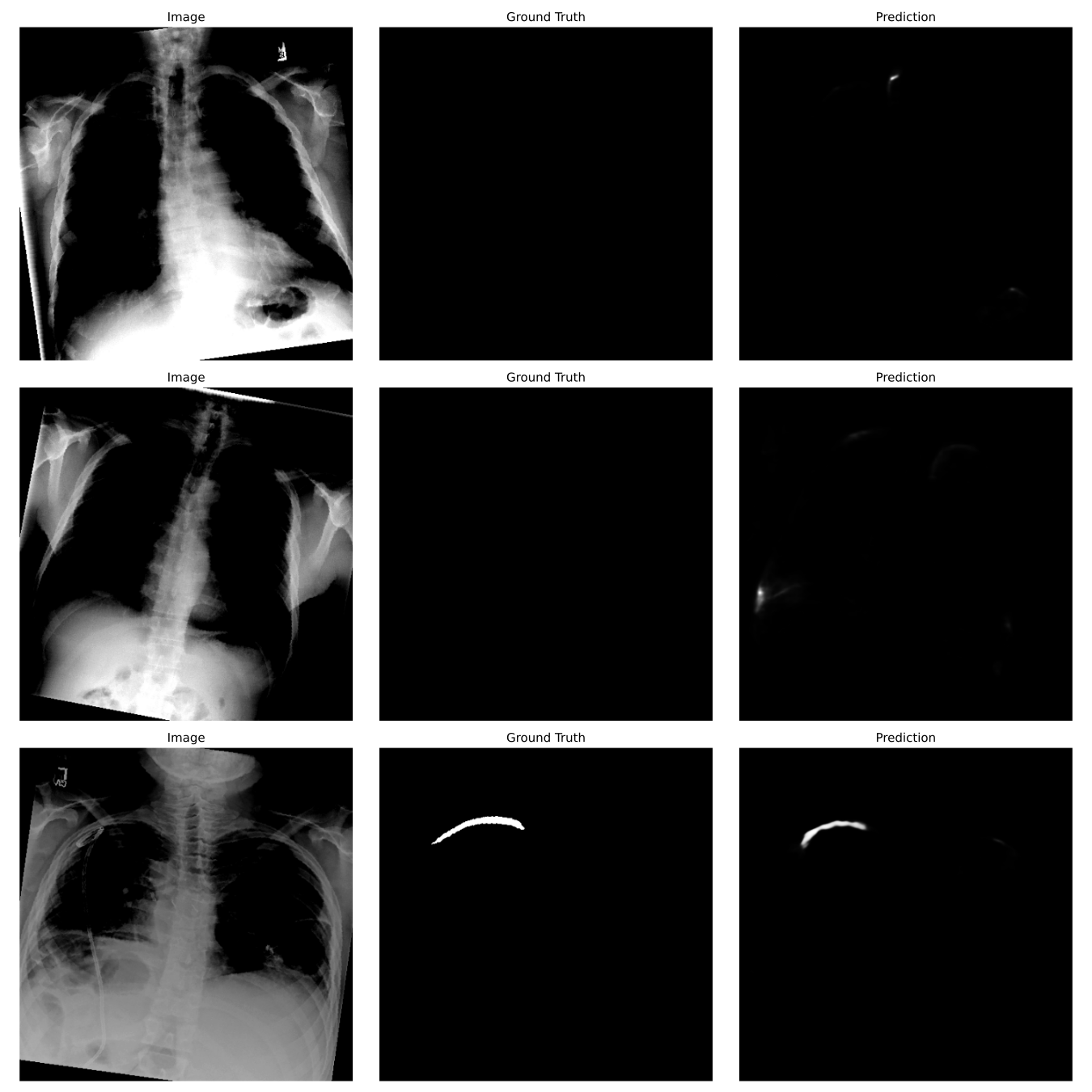


## Unet with VGG16

### Learning curves

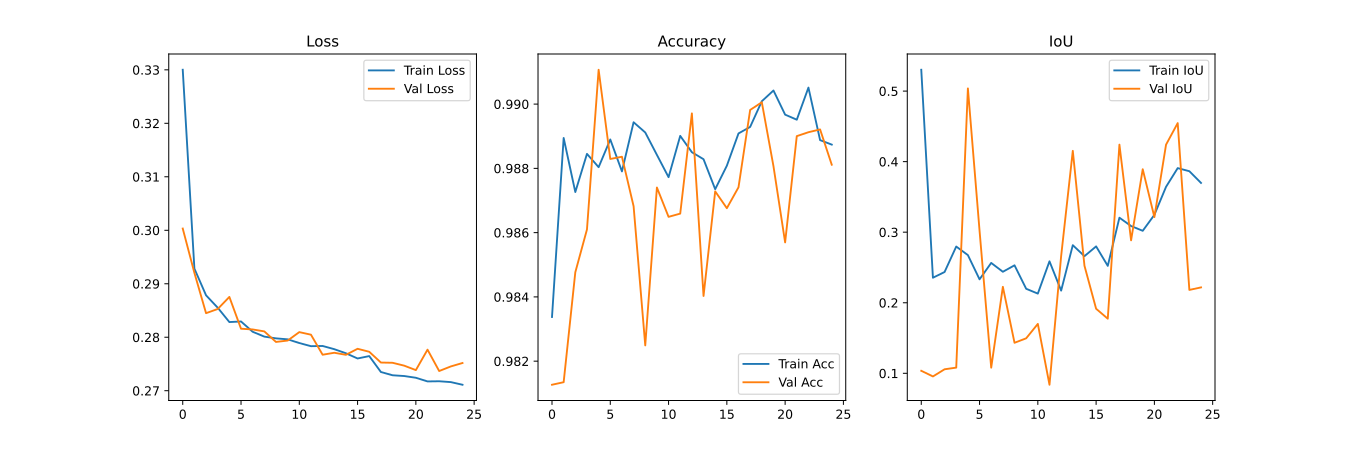


### Example predictions

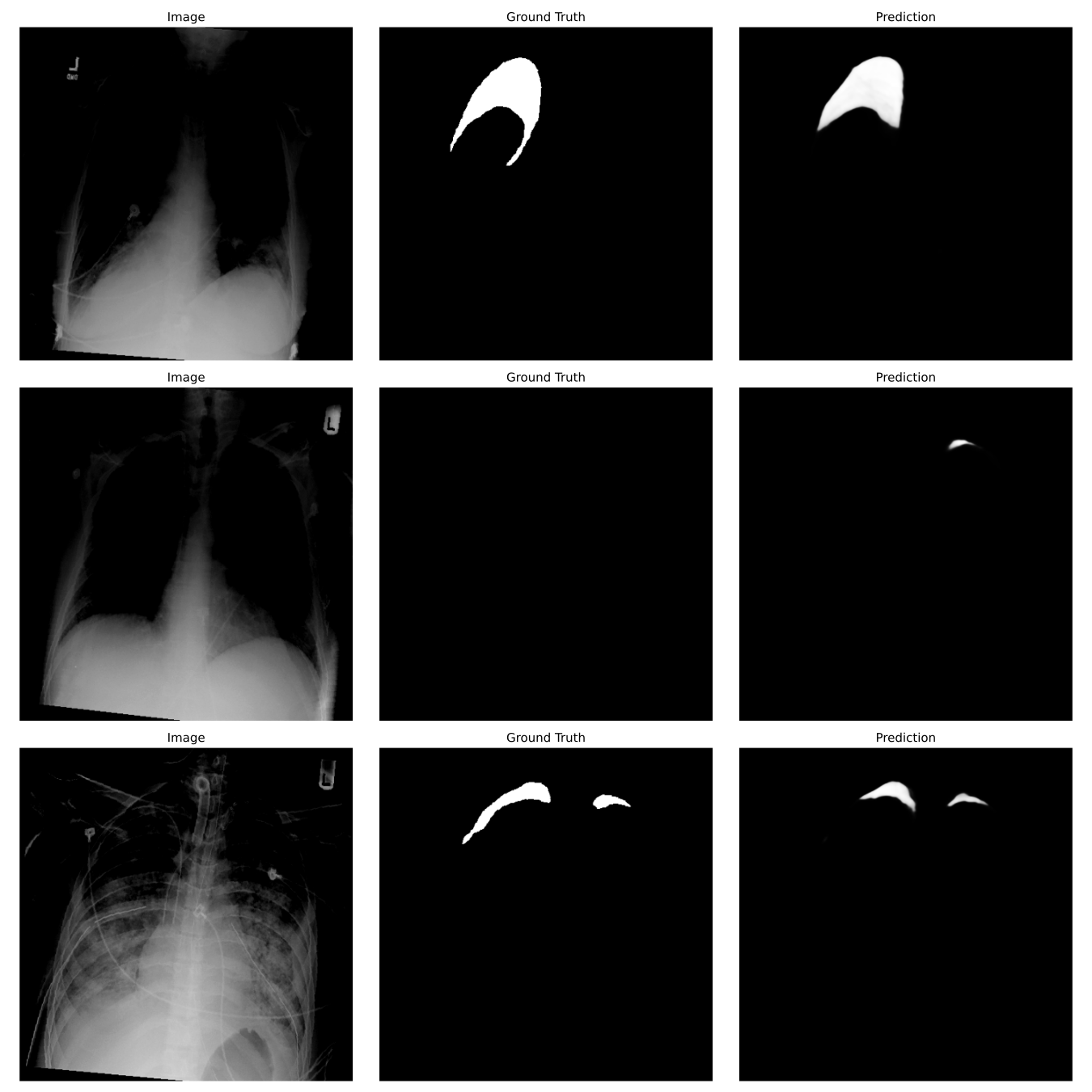


## Unet with InceptionResnetV2

### Learning curve

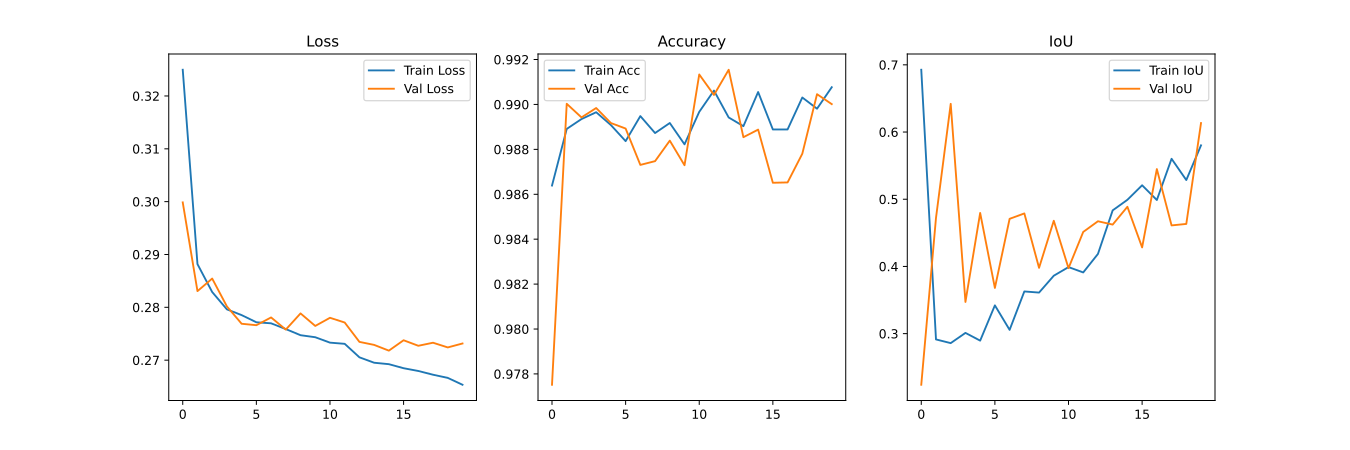


### Example predictions

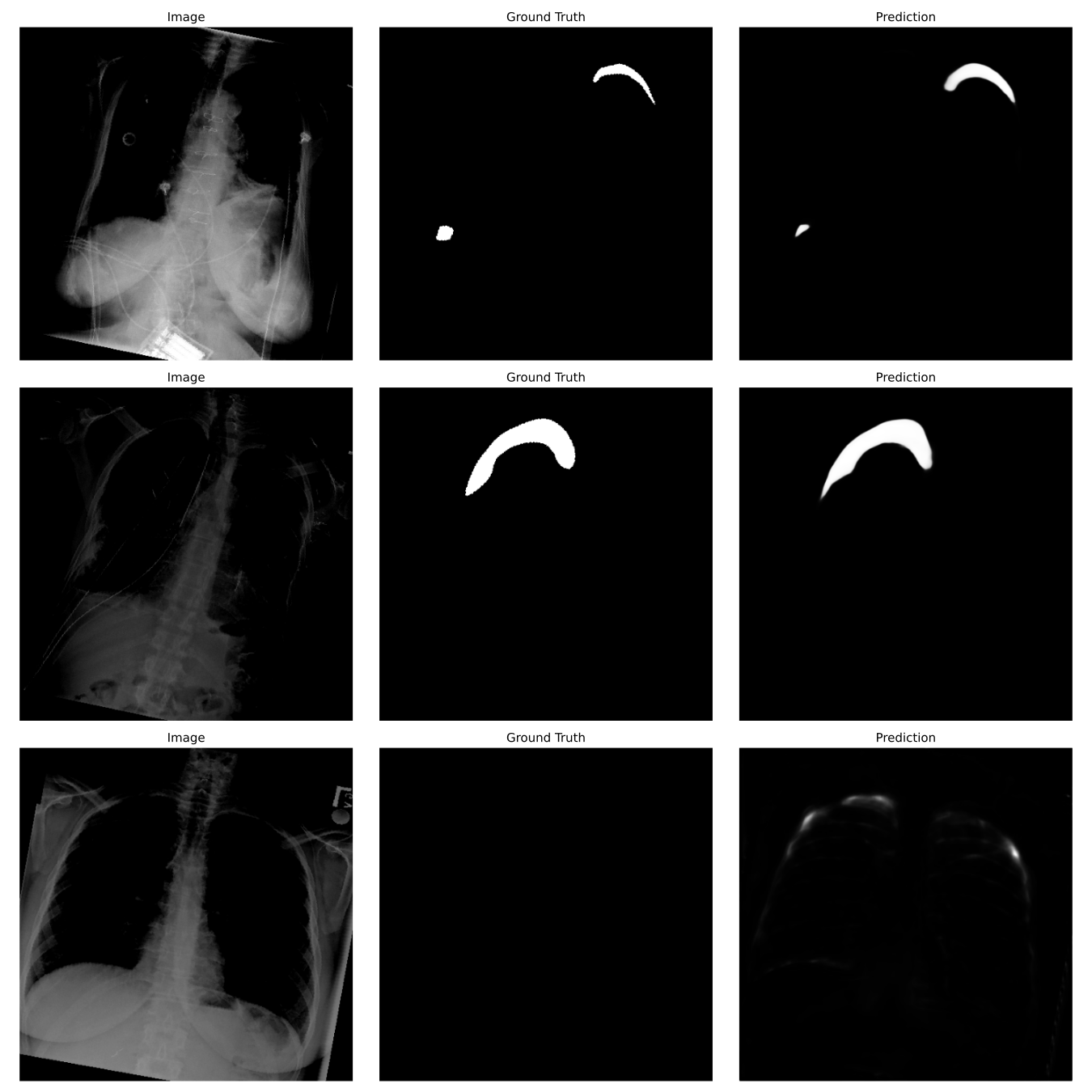


## Unet with Xception

### Learning curve



### Example predictions



# Comparison

|  | Loss | IOU | Accuracy |
| --- | --- | --- | --- |
| ResNet34 | 0.2761 | 0.4733 | 0.9885 |
| VGG16 | 0.2811 | 0.5504 | 0.9928 |
| InceptionResNetv2 | 0.2718 | 0.2238 | 0.9943 |
| Xception | 0.2674 | 0.6085 | 0.9920 |

# Final Thoughts on Segmentation Phase.

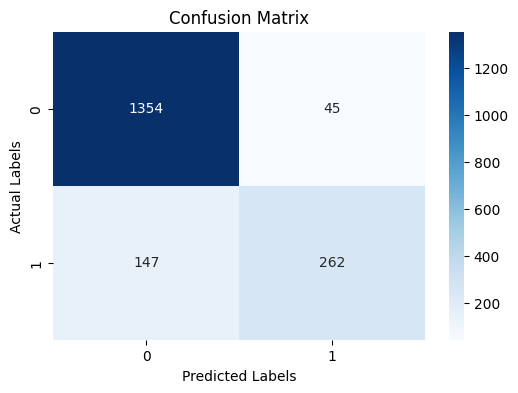
Based on the results, while the **InceptionResNetV2** model achieved the highest accuracy, the **Xception** model stands out as the preferred choice for this medical segmentation task. This is due to its lower loss compared to the other models and its superior performance in mask prediction, as demonstrated in the visualizations. Furthermore, Xception was the fastest in terms of prediction time, making it both efficient and reliable for this application.

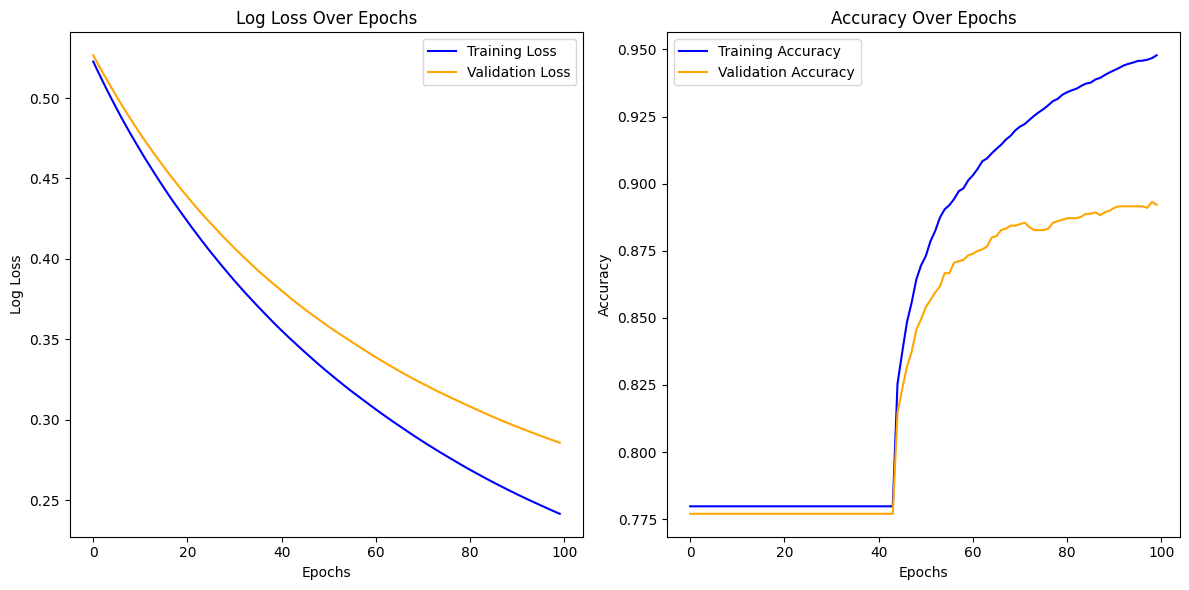
# 

# Experimental Results for Classification

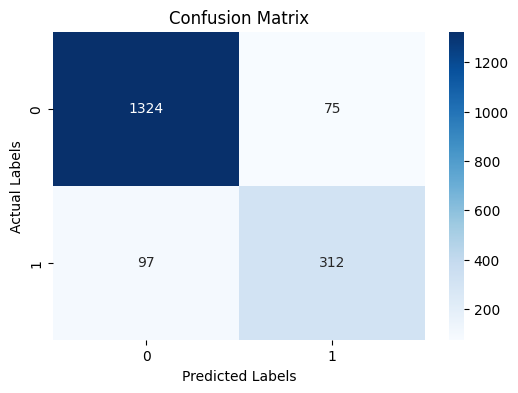
For this step the Xception model is used to predict the masks due to it’s superior performance compared to the other models

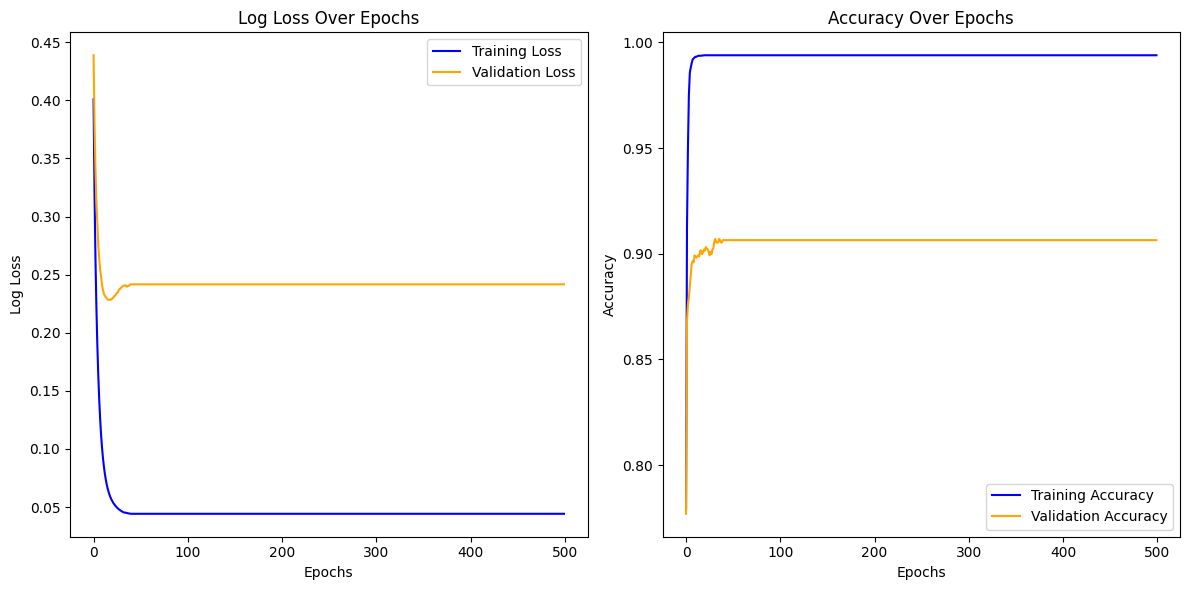
## Before hyperparameter tuning





## After hyperparameter tuning





# Comparison

|  | Class | Precision | Recall | F1 Score |
| --- | --- | --- | --- | --- |
| Before Random Search | 0 | 0.90 | 0.97 | 0.93 |
| 1 | 0.85 | 0.64 | 0.73 |
| After Random Search | 0 | 0.93 | 0.95 | 0.94 |
| 1 | 0.81 | 0.76 | 0.78 |

# Final Thoughts on Classification Phase.

After conducting a random search with the XGBoost classifier, the model demonstrated a strong recall for detecting pneumothorax, particularly in identifying positive cases. However, the learning curve indicates noticeable overfitting, which suggests that while the model performs well on the training data, its generalization to new, unseen data is limited. Given this, exploring a deep learning-based classifier may offer a more robust solution, potentially improving both recall and overall performance while mitigating overfitting issues.