COMPLEX OR IRREGULAR LUNG NODULE DETECTION FROM COMPUTED TOMOGRAPHY SCANS

A PROJECT REPORT

Submitted by

KANISHK AADHAW P [CB.EN.U4CSE20227]
LAXMAN K R [CB.EN.U4CSE20234]
ARVIND B [CB.EN.U4CSE20209]
DHANUPRASAD M R [CB.EN.U4CSE20118]

in partial fulfillment for the award of the degree

of

BACHELOR OF TECHNOLOGY

IN

COMPUTER SCIENCE AND ENGINEERING

AMRITA SCHOOL OF COMPUTING

AMRITA VISHWA VIDYAPEETHAM

COIMBATORE - 641 112

MAY 2024

AMRITA VISHWA VIDYAPEETHAM

AMRITA SCHOOL OF COMPUTING, COIMBATORE – 641 112

BONAFIDE CERTIFICATE

This is to certify that the project report entitled **COMPLEX OR IRREGULAR LUNG NODULE DETECTION FROM COMPUTED TOMOGRAPHY SCANS** submitted by Kanishk Aadhaw P (CB.EN.U4CSE20227), Laxman K R (CB.EN.U4CSE20234), Arvind B (CB.EN.U4CSE20209), Dhanuprasad M R (CB.EN.U4CSE20118) in partial fulfillment of the requirements for the award of Degree **Bachelor of Technology** in Computer Science and Engineering is a bonafide record of the work carried out under our guidance and supervision at the Department of Computer Science and Engineering, Amrita School of Computing, Coimbatore.

Ms. Sathiya R R
(Assistant professor)
Department of CSE

Dr. Vidhya Balasubramanian
Chairperson
Department of CSE

Evaluated on:

INTERNAL EXAMINER

EXTERNAL EXAMINER

DECLARATION

We, the undersigned solemnly declare that the project report **COMPLEX OR IRREGULAR LUNG NODULE DETECTION FROM COMPUTED TOMOGRAPHY SCANS** is based on our own work carried out during the course of our study under the supervision of Ms. Sathiya R R, (Assistant professor), Computer Science and Engineering, and has not formed the basis for the award of any other degree or diploma, in this or any other Institution or University. In keeping with the ethical practice in reporting scientific information, due acknowledgement has been made wherever the findings of others has been cited.

Kanishk Aadhaw P[CB.EN.U4CSE20227]-
Laxman K R [CB.EN.U4CSE20234]-
Arvind B[CB.EN.U4CSE20209]-
Dhanuprasad M R[CB.EN.U4CSE20118]-

ABSTRACT

Lung cancer is the leading cause for cancer-related deaths accounting for the maximum mortality rates(1 in 5), also with a minimal survival rate of 15%. With the growing number of smokers increasing the vulnerability rate, early and accurate detection with the help of medical image classification and deep learning models is the need of hour. Computed Tomography (CT) scans often fail to identify the nodules which are irregular and complex,thereby failing to test malignancy. This paper uses deep learning-based convolutional network and shape modelling algorithm to assist in the detection of cancerous nodules and reduce misinterpretation.

ACKNOWLEDGEMENT

We would like to express our deep gratitude to our beloved Satguru **Sri Mata Amritanandamayi Devi**. This acknowledgment is intended to thank all those people involved directly or indirectly with our project. We express our thanks to **Dr. Vidhya Balasubramanian**, Chairperson, Department of Computer Science and Engineering and Principal, School of Computing, Amrita Vishwa Vidyapeetham, **Dr. C.Shunmuga Velayutham and Dr. N.Lalithamani,** Vice Chairpersons of the Department of Computer Science and Engineering. We express our gratitude to our guide, **Ms. Sathiya R R** for her guidance, support and supervision. We would like to extend our sincere thanks to our family and friends for helping and motivating us during the course of the project. Finally, we would like to thank all those who have helped, guided, and encouraged us directly or indirectly during the project work.

Kanishk Aadhaw P [CB.EN.U4CSE20227]

Laxman K R [CB.EN.U4CSE20234]

Arvind B [CB.EN.U4CSE20209]

Dhanuprasad M R [CB.EN.U4CSE20118]

TABLE OF CONTENTS

LIST OF FIGURES

ABBREVIATIONS

ASPP	Atrous Spatial Pyramid Pooling
BCE	Binary Cross Entropy
CBAM	Convolutional Block Attention Model
CNN	Convolutional Neural Network
COPD	Chronic Obstructive Pulmonary Disease
CT	Computed Tomography
DETR	Detection Transformer
FN	False Negative
FP	False Positive
IDRI	Image Database Resource Initiative
IoU	Intersection over Union
LIDC	Lung Image Database Consortium
MLP	Multi Layer Perceptron
NMS	Non Maximum Suppression
ReLU	Rectified Linear Unit
ROC	Receiver Operating Characteristic
SPPF	Spatial Pyramid Pooling Fast
TN	True Negative
TP	True Positive
V-Net	Volumetric Convolutional Neural Network
YOLO	You Only Look Once

Chapter 1

INTRODUCTION

Lung cancer is the main cause of cancer-related deaths around the world. In CT lung cancer screening, millions of CT filters will need to be analyzed, which is an gigantic burden for radiologists. So there is a need to create computer calculations to improve screening. It is crucial to begin with the step for investigation of lung cancer screening CT checks is the discovery of lung knobs, which may or may not speak to early organize lung cancer. A lung nodule is the small round range that's more than thick than ordinary standard tissue. Individuals with early-stage lung cancer that's treated are less likely to move to further stages than individuals who are analyzed afterwards. organize. Conventional symptomatic strategies frequently drop brief in giving convenient and precise comes about, driving to deferred medications and less positive outcomes. To address this challenge, a good learning models are utilized, which employments the therapeutic pictures such as CT filters and X-Ray filters. Typically broadly known as therapeutic picture classification. This approach encourages to examine information with progressed learning models which recognize designs related to lung cancer indications.

However, such method comes short when the lung nodules are complex in shape. In such cases, the outcomes tend to produce a higher number of false positives, also known as Type I Errors, or misclassifications. This issue poses a substantial risk in the realm of medical diagnosis, and it is imperative that we promptly address and rectify these errors.

1.1 Problem Definition

Lung Nodule Detection is an important task in medical imaging and plays a vital role in early treatment of lung diseases particularly lung cancer. Nodules are small masses of tissues in the lung either round or oval in shape. They are smaller than 3 centimeters in

diameter and are often non-cancerous. But some nodules can be an early sign on lung cancer especially if they grow in size over time. CT Scans provide detailed images of lungs and are commonly used to detect the nodules. Due to the high volume of scans and the small size of nodules, manual detection by radiologists can be time consuming. Automated detection of lung nodules can significantly improve the efficiency and accuracy of diagnosis. This is very important given that lung cancer is one of the leading causes of cancer related deaths worldwide and detecting it early can improve survival rates. They also can assist in monitoring the growth of nodules over time to

1.1.1 Applications

1. Research & Clinical Traits - It can be used in studies to evaluate the effectiveness of new treatments.
2. Treatment Plans - By monitoring the lung nodules doctors can develop personalized treatment plans based on size, location and growth rate of nodules.
3. Radiologist Efficiency - Assisting the radiologists in detection can help them focus on other complex tasks related to diagnosis.
4. Telemedicine - In remote areas where access to radiologists is limited, automated lung nodule detection can provide timely and effective consulations.
5. Intergation - Can be used with other AI systems to create comprehensive diagnostic tools. For example after detecting a nodule another algorithm could be used to classify it as a cancerous or non- cancerous.

1.1.2 Challenges

1. Variability in Nodule Characteristics - They can vary greatly in size, shape, density and location making it difficult to detect accurately.
2. False Positives - One of the significant challenge faced is high rate of false positives, where benign nodules are incorrectly identified as cancerous. This leads to unnecessary further testing and anxiety for patients.
3. Adopting to New Data - A model trained on data from one hospital or one type of CT scanner may not perform as well when applied to data from a different source. This is due to variation in imaging protocols, patients populations and other factors.

Our approach leverages a Volumetric Convolutional Neural Network (V-Net) for volumetric segmentation, and a YOLOv5s model augmented with shape modeling to accurately detect and classify nodules based on their shape characteristics. We further

improve the system by including a Detection Transformer (DETR) model for refined detection.

Chapter 2

LITERATURE SURVEY

2.1 Ensemble Learning for Medical Image Classification

Ensemble learning stands as a machine learning technique that holds significant promise in the realm of medical image classification. This approach involves amalgamating predictions from multiple models to generate more precise and robust outcomes. Its relevance becomes particularly pronounced when dealing with the intricacies of medical image analysis, where data often exhibits considerable variability, and models can be susceptible to noise-induced errors.

The utility of ensemble learning in elevating the accuracy of medical image classification is exemplified by several noteworthy studies. For instance, Müller et al. (2022) observed that stacking, bagging, and augmenting techniques led to substantial performance enhancements. Dhar (2021) introduced a novel ensemble learning model designed for pulmonary nodule detection, surpassing the capabilities of existing state-of-the-art models. Moreover, Ji et al. (2023) advanced the field by proposing a multistage ensemble learning model for the detection of Chronic Obstructive Pulmonary Disease (COPD), demonstrating superior performance compared to individual classifiers when evaluated on real-world datasets.

Furthermore, ensemble learning's potential has been explored and discussed in a broader context by Cao et al. (2020), who conducted a review on the application and evolution of computer-aided diagnosis models that rely on ensemble learning techniques for lung nodule classification and detection. These findings collectively underscore the pivotal role of ensemble learning in enhancing the accuracy and effectiveness of medical image analysis, offering promising prospects for more reliable diagnostic and classification systems in the healthcare domain.

2.2 Object Detection Model

Object detection models, particularly You Only Look Once (YOLO) architecture, offer a promising solution. Ji et al. (2023) introduces several modifications to the YOLOv5s object detection model specifically for pulmonary nodule detection. The incorporation of Convolutional Block Attention Model (CBAM) emphasise important features across both channel and spatial dimensions. Replacing the Rectified Linear Unit (ReLU) activation with Leaky ReLU aims to improve gradient flow said Ji et al. (2023).

The Atrous Spatial Pyramid Pooling (ASPP) enhanced the multi-scale feature extraction than standard Spatial Pyramid Pooling Fast (SPPF), through the use of dilated convolutions. This helps in preserving image details crucial for recognizing nodules of varying sizes. Finally the contextual transformer module combines self-attention from transformer architecture with traditional convolutional layers, aiming to improve feature representation and contextual understanding, potentially benefiting small nodule detection.

2.3 Summary

Ensemble learning techniques, which combine the outputs of multiple models, have proven highly effective in improving the accuracy and robustness of medical image classification. Studies by Müller et al. (2022), Dhar (2021) and Ji et al. (2023) demonstrate the performance gains achievable with ensemble methods for tasks like pulmonary nodule detection. Object detection models, specifically the YOLO architecture, offer a powerful approach for medical image analysis. Ji et al. (2023) demonstrates how modifications to the YOLOv5s model, such as the use of CBAM, Leaky ReLU, ASPP, and a Contextual Transformer module, can be specifically tailored to enhance lung nodule detection.

2.4 Data Set

The Lung Image Database Consortium (LIDC)-Image Database Resource Initiative (IDRI) consists of diagnostic and lung cancer screening thoracic computed tomography (CT) scans with marked-up annotated lesions.

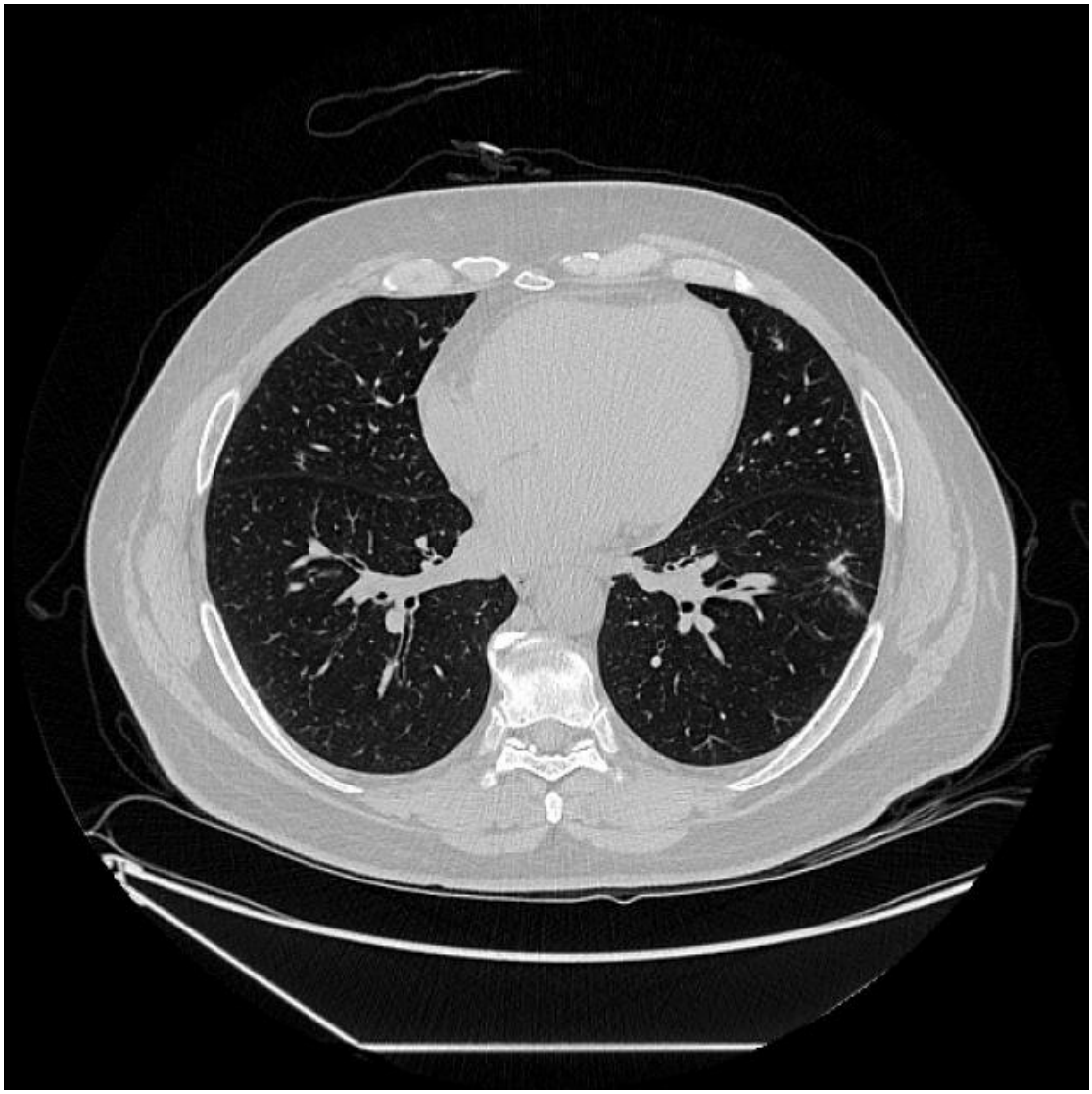

Figure 2.1: Sample CT scan image

2.5 Software/Tools Requirements

- Scikit-learn: is employed for various machine learning tasks, including feature selection, model selection, evaluation metrics, and preprocessing.
- Matplotlib: is used for creating static, animated interactive visualizations in Python
- Numpy: A python package used for processing arrays which are multi-dimensional
- Ultralytics Python package: that provides various utilites for YOLOv5s including functions for model training and inference.

Chapter 3

PROPOSED SYSTEM

This paper uses deep learning-based convolutional network and shape modelling algorithm to assist in the detection of cancerous nodules and reduce misinterpretation. The models based on Convolutional Neural Network (CNN) are V-Net, YOLOv5s & DETR.

3.1 System Analysis

3.1.1 Module details of the system

V-Net

V-Net is one of the most commonly used convolutional neural network used for medical image segmentation and analysis of 3D volumes.In the left side of compression path the image is compressed at different resolutions where the details of each voxel from a particular layer goes through a residual function which helps to pick up complex pattern with the given training data and helps in the automatic segmentation of lung nodules.A weighted learning module in the residual function add weights to the feature data obtain from each layer which helps in refining shalow, complex or irregular nodules present in CT scan data.In the de convolutional process,the size of the inputs is increased or information is gathered by combining the convolutional layer and the residual function does the same process here. Utilizing convolutional layers in place of pooling layers leads to a reduced memory footprint for the network during training. There is also a feature mapping done in order to get the locations of the initial voxels. The V-Net precisely marks the nodule position and can get great division comes about for lung nodules of diverse shapes and sizes, giving a great premise for assist determination. Our CNN is prepared with CT filter pictures and its veil of lung nodules, and learns to anticipate division for the complete volume at once. We present a novel objective work, that we advance amid preparing, based on Dice coefficient. In this way we are able bargain with circumstances where there's a solid awkwardness between the number of

closer view and foundation voxels.

YOLOv5s

The key advancement of YOLO is its capacity to perform real-time protest discovery in a single pass through the neural arrange, making it inconceivably quick and effective. Not at all like conventional CNNs, which employments complex multi-stage pipelines, YOLO employments a single bound together show for both locale proposition and classification. YOLO is a one stage framework model which is used for object detection of one or multiple nodules by assigning probabilities to the each detected feature using a single convolutional network. Every CT scan slice is used as a single image for inference and training. After the segmentation of the images, N*N grids cells are made from which we label classes are predict them along with the confidence value/probability. Grids with probability greater than o are taken and labelled as significant. When predicting bounding boxes, YOLO uses a combination of the anchor boxes and the predicted offsets to determine the regions of interest. The best practise is to put the image slice for segementation, scale the image to a bigger size so that YOLO detects it well.The fusion of these two together will involve overlaying the segmentation results from V-Net onto the original image localization of objects from YOLO.

Shape modeling algorithm Lung nodules can vary greatly in shape, size, and texture. It is really important to understand the local shape information of the detected nodule as this might be a used for candidate generation and label creation.This step says about the type of the nodule which can be solid or ground-glass opacity nodules. The parameters "compactness," "eccentricity," and "solidity" are used to describe different shape characteristics of the segmented tumor region.These shape descriptors provide quantitative measures of the tumor's shape characteristics, which can be useful for analysis and detection of tumors by the model. In conventional lung knob location calculations, analysts plan distinctive sorts of highlights based on the gray esteem, area, shape and surface of lung knobs in CT pictures where it is utilized to perform perform candidate knob location based on neighborhood shape data and nearby escalated scattering data. This permits for more exact discovery and characterization of knobs, lessening the probability of untrue positives and untrue negatives.

DETR

The DETR demonstrate is an encoder-decoder transformer with a convolutional spine. Two heads are included on beat of the decoder yields in arrange to perform question discovery. A straight layer for the course names and a Multi Layer Perceptron (MLP) for the bounding boxes. The show employments protest questions to distinguish objects in an picture. Each protest inquiry looks for a specific protest within the picture.

3.2 System Design

3.2.1 Flow diagram of the system

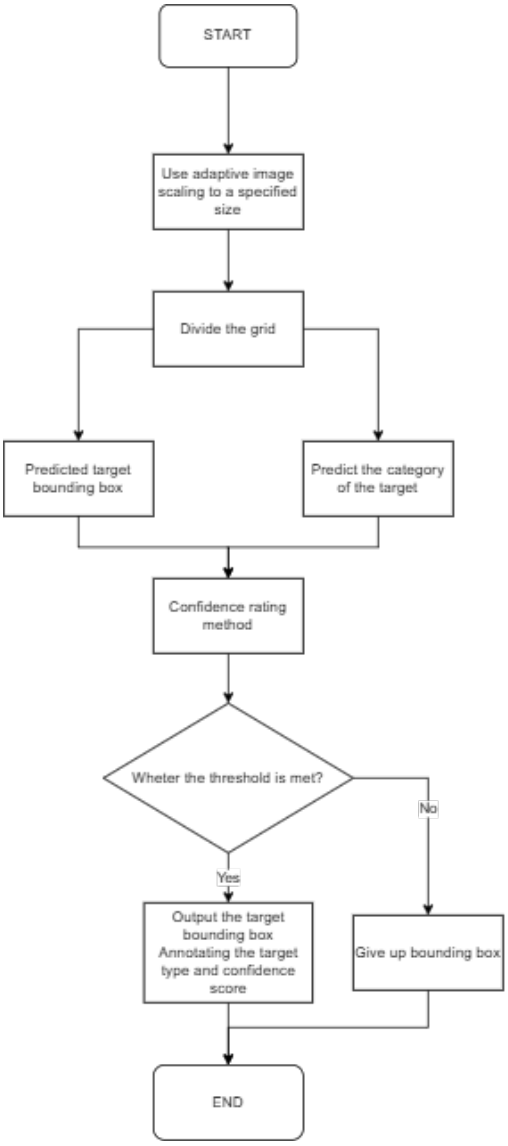

Figure 3.1: YOLOv5s data flow diagram

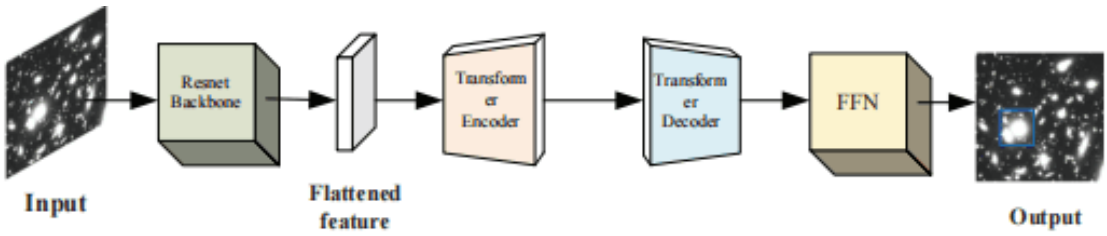

Figure 3.2: DETR data flow diagram

3.2.2 Architecture diagram

V-Net

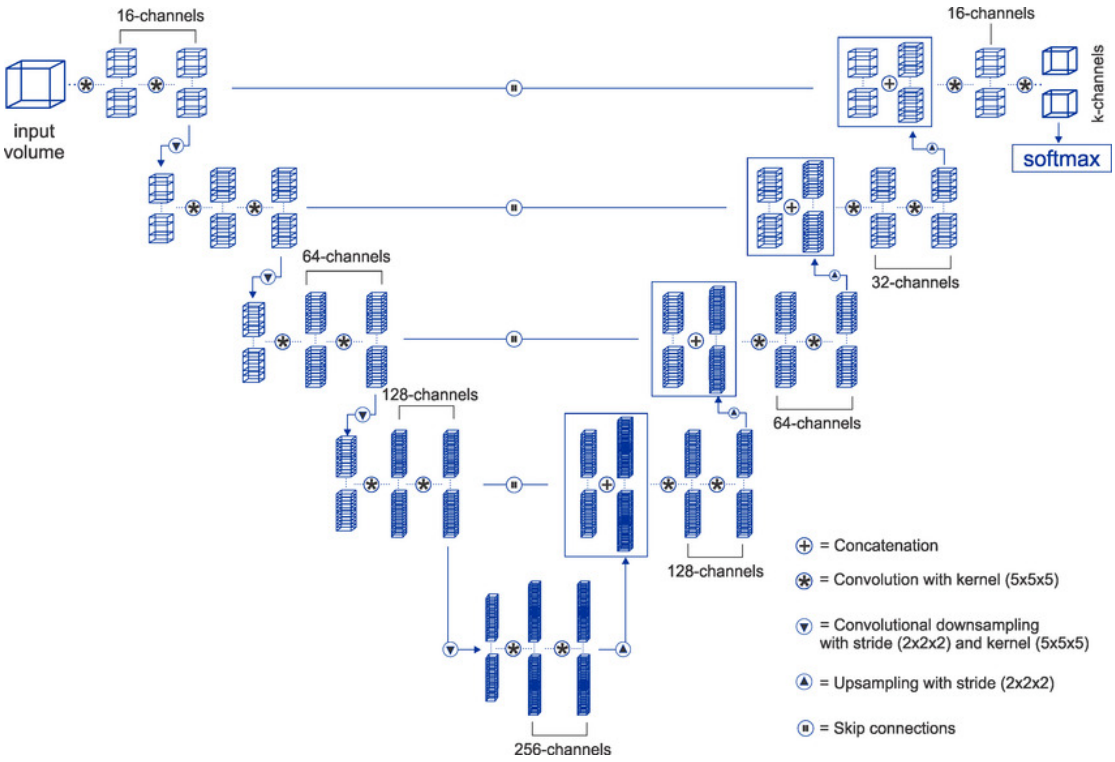

Figure 3.3: Architecture of V-Net

YOLOv5s

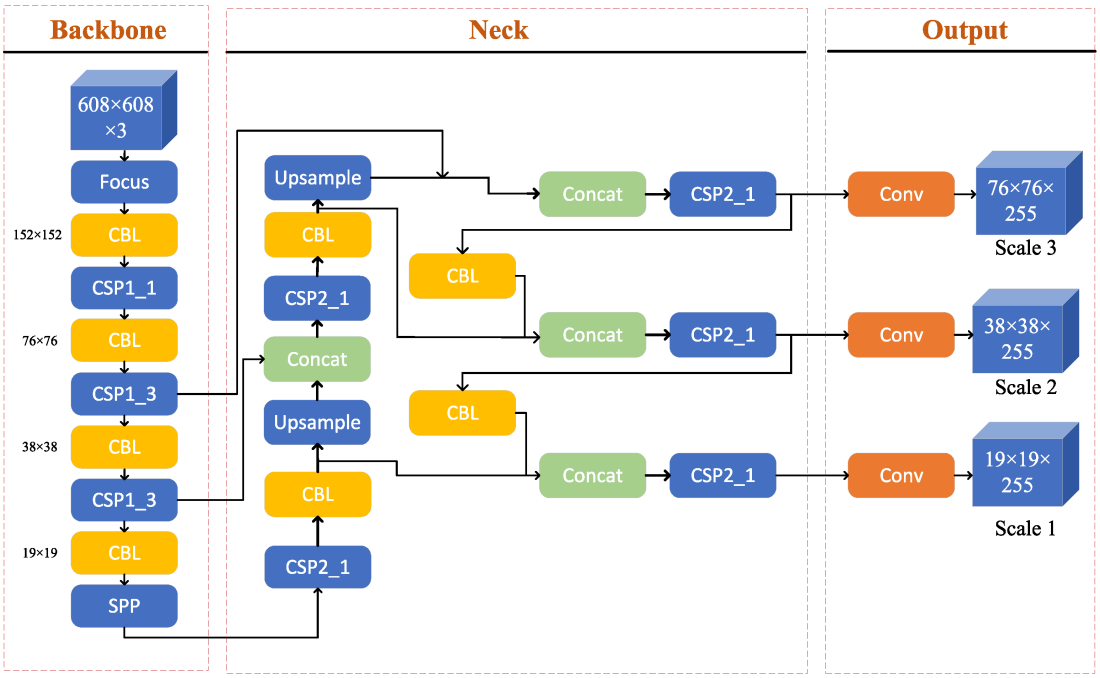

Figure 3.4: Architecture of YOLOv5s

DETR

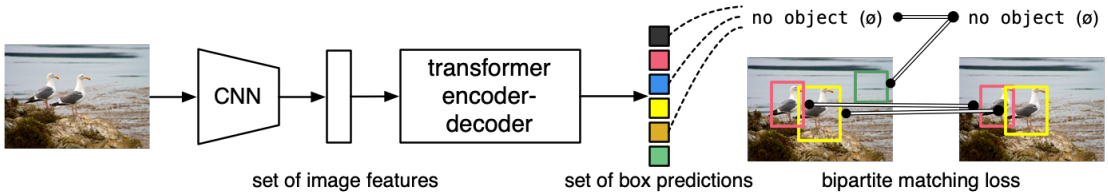

Figure 3.5: Architecture of DETR

Chapter 4

IMPLEMENTATION AND TESTING

4.1 V-Net Implementation

4.1.1 Resizing

```
def resize_images(src_folder, dest_folder, target_size=(256, 256)):
    for root, _, files in os.walk(src_folder):
        for filename in files:
            src_path = os.path.join(root, filename)
            try:
                with Image.open(src_path) as img:
                    img_resized = img.resize(target_size, Image.
    ANTIALIAS)
                    dest_path = os.path.join(dest_folder, filename)
                    img_resized.save(dest_path)
                    print(f"Resized {filename} to {target_size[0]}x{
    target_size[1]}")
            except Exception as e:
                print(f"Error processing {filename}: {e}")
```

This is a crucial step in preprocessing as it ensures all input images have consistent dimensions, which is essential for neural networks to process them efficiently. When training CNN's having uniform image sizes simplifies the architecture design and educes computational complexity. So why choose the 256*256 pixels. Larger images contain more details but increase memory usage and computational load during training. In the opposite smaller images reduce computational cost but may lose critical information. In the middle lies 256*256 where it is large enough to capture relevant features while remaining computationally manageable. Both the images and the masks are resized.

4.1.2 Diagnosis

```
df = pd.DataFrame({"image_path": train_files, "mask_path":mask_files
    })
def diagnosis(mask_path):
    value = np.max(cv2.imread(mask_path))
    if value:
        return 1
    else:
        return 0
df['mask'] = df["mask_path"].apply(lambda x: diagnosis(x))
df.head()
```

The function takes the mask path and takes the maximum pixel value meaning if there is a nodule present then it returns 1 otherwise it returns 0. At the end a data frame is created where the image is mapped to the mask and the presence of the nodule is indicated.

	image_path	mask_path	mask
0	/content/seg/Images/54_15.bmp	/content/seg/mask/54_15_mask.bmp	1
1	/content/seg/Images/1_34.bmp	/content/seg/mask/1_34_mask.bmp	1
2	/content/seg/Images/21_183.bmp	/content/seg/mask/21_183_mask.bmp	1
3	/content/seg/Images/44_11.bmp	/content/seg/mask/44_11_mask.bmp	0
4	/content/seg/Images/25_33.bmp	/content/seg/mask/25_33_mask.bmp	0

Figure 4.1: Dataframe containing path of image and mask

4.1.3 Training & Testing

```
from sklearn.model_selection import train_test_split
df_train, df_test = train_test_split(df, test_size=0.15)
df_train, df_val = train_test_split(df_train, test_size=0.15)
print(df_train.values.shape)
print(df_val.values.shape)
print(df_test.values.shape)
```

This is a commonly used method to divide a dataset into training and testing subsets. The primary purpose is to assess how well a model generalizes to unseen data. The function splits the dataset into three subsets: Training set - used for training and fitting the model Validation set - to tune hyperparameters and evaluating model performance during training helps prevent overfitting. Train set - used to assess the models performance after training The test size specifies the proportion of data to allocate to the test set (15% used here). Splitting is done twice here - first into training and validation sets then again into training and validation subsets.

4.1.4 Train Generator

```
    train_generator_args = dict(rotation_range=0.1,
                                width_shift_range=0.05,
                                height_shift_range=0.05,
                                shear_range=0.05,
                                zoom_range=0.05,
                                horizontal_flip=True,
                                vertical_flip=True,
                                fill_mode='nearest')

train_gen = train_generator(df_train, BATCH_SIZE,
                                    train_generator_args,
                                    target_size=IMAGE_SIZE)

val_gen = train_generator(df_val, BATCH_SIZE,
                                    dict(),
                                    target_size=IMAGE_SIZE)

def train_generator(data_frame, batch_size, aug_dict,
        image_color_mode="rgb",
        mask_color_mode="grayscale",
        image_save_prefix="image",
        mask_save_prefix="mask",
        save_to_dir=None,
        target_size=(256,256),
        seed=1):

```

```
27      image_datagen = ImageDataGenerator(**aug_dict)
28      mask_datagen = ImageDataGenerator(**aug_dict)
29
30      image_generator = image_datagen.flow_from_dataframe(
31          data_frame,
32          x_col = "image_path",
33          class_mode = None,
34          color_mode = image_color_mode,
35          target_size = target_size,
36          batch_size = batch_size,
37          save_to_dir = save_to_dir,
38          save_prefix  = image_save_prefix,
39          seed = seed)
40
41      mask_generator = mask_datagen.flow_from_dataframe(
42          data_frame,
43          x_col = "mask_path",
44          class_mode = None,
45          color_mode = mask_color_mode,
46          target_size = target_size,
47          batch_size = batch_size,
48          save_to_dir = save_to_dir,
49          save_prefix  = mask_save_prefix,
50          seed = seed)
51
52      train_gen = zip(image_generator, mask_generator)
53
54      for (img, mask) in train_gen:
55          img, mask = adjust_data(img, mask)
56          yield (img,mask)
57
58  def adjust_data(img,mask):
59      img = img / 255.
60      mask = mask / 255.
61      mask[mask > 0.5] = 1
62      mask[mask <= 0.5] = 0
63
64      return (img, mask)
```

It sets up a data generator for image segmentation tasks. Data augmentation is applied to both the input and their corresponding masks. The goal is to improve the models ability to generalize by creating variations of the training data. The ImageDataGenerator class from Tensorflow Keras is used to generate augmented versions of the images during training. Augmentation techniques introduce variations to the original images, making the model more robust. A dictionary is created which contains various augmentation parameters like rotation_range, shear_range, zoom_range, fill_mode, horizontal_flip_vertical_flip, width_shift_range,height_shift_range. These parameters control how training images are modified during training. The flow_from_dataframe method is used to create the generators. Then both the generators are zipped to combine into one single generator and for one single batch it has augmented images and the corresponding masks. The generator is then passed into adjust function where normalization happens. The images and masks are divided by 255 and any pixel value > 0.5 set to 1 and others to 0. This operation converts the mask into a binary representation, making it suitable for segmentation tasks.

Encoding Layer

```
img_input = Input(shape= (256, 256, 3))
x = Conv2D(64, (3, 3), padding='same', name='conv1',strides= (1,1))(
    img_input)
x = BatchNormalization(name='bn1')(x)
x = Activation('relu')(x)
x = Conv2D(64, (3, 3), padding='same', name='conv2')(x)
x = BatchNormalization(name='bn2')(x)
x = Activation('relu')(x)
x = MaxPooling2D()(x)

x = Conv2D(128, (3, 3), padding='same', name='conv3')(x)
x = BatchNormalization(name='bn3')(x)
x = Activation('relu')(x)
x = Conv2D(128, (3, 3), padding='same', name='conv4')(x)
x = BatchNormalization(name='bn4')(x)
x = Activation('relu')(x)
x = MaxPooling2D()(x)

```

```
x = Conv2D(256, (3, 3), padding='same', name='conv5')(x)
x = BatchNormalization(name='bn5')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same', name='conv6')(x)
x = BatchNormalization(name='bn6')(x)
x = Activation('relu')(x)
x = Conv2D(256, (3, 3), padding='same', name='conv7')(x)
x = BatchNormalization(name='bn7')(x)
x = Activation('relu')(x)
x = MaxPooling2D()(x)

x = Conv2D(512, (3, 3), padding='same', name='conv8')(x)
x = BatchNormalization(name='bn8')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same', name='conv9')(x)
x = BatchNormalization(name='bn9')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same', name='conv10')(x)
x = BatchNormalization(name='bn10')(x)
x = Activation('relu')(x)
x = MaxPooling2D()(x)

x = Conv2D(512, (3, 3), padding='same', name='conv11')(x)
x = BatchNormalization(name='bn11')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same', name='conv12')(x)
x = BatchNormalization(name='bn12')(x)
x = Activation('relu')(x)
x = Conv2D(512, (3, 3), padding='same', name='conv13')(x)
x = BatchNormalization(name='bn13')(x)
x = Activation('relu')(x)
x = MaxPooling2D()(x)

x = Dense(1024, activation = 'relu', name='fc1')(x)
x = Dense(1024, activation = 'relu', name='fc2')(x)
```

Input layer is created first with the shape of 256*256*3. Then comes the encoding layer (Convolutional Blocks) which are stacked together. Each black has two 3*3 conv layers followed by batch normalization and ReLU activation. After each pair of con-

volutional layers, a max-pooling operation is applied to downsample the feature maps. The number of filters increases form 64 to 512.Next there are two Fully conencted layers having 1024 units each. They follow the conv layers to capture high level features and prepare for classification.

Decoding Layer

```
x = UpSampling2D()(x)
x = Conv2DTranspose(512, (3, 3), padding='same', name='deconv1')(x)
x = BatchNormalization(name='bn14')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(512, (3, 3), padding='same', name='deconv2')(x)
x = BatchNormalization(name='bn15')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(512, (3, 3), padding='same', name='deconv3')(x)
x = BatchNormalization(name='bn16')(x)
x = Activation('relu')(x)

x = UpSampling2D()(x)
x = Conv2DTranspose(512, (3, 3), padding='same', name='deconv4')(x)
x = BatchNormalization(name='bn17')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(512, (3, 3), padding='same', name='deconv5')(x)
x = BatchNormalization(name='bn18')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(256, (3, 3), padding='same', name='deconv6')(x)
x = BatchNormalization(name='bn19')(x)
x = Activation('relu')(x)

x = UpSampling2D()(x)
x = Conv2DTranspose(256, (3, 3), padding='same', name='deconv7')(x)
x = BatchNormalization(name='bn20')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(256, (3, 3), padding='same', name='deconv8')(x)
x = BatchNormalization(name='bn21')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(128, (3, 3), padding='same', name='deconv9')(x)
x = BatchNormalization(name='bn22')(x)
```

```
x = Activation('relu')(x)

x = UpSampling2D()(x)
x = Conv2DTranspose(128, (3, 3), padding='same', name='deconv10')(x)
x = BatchNormalization(name='bn23')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(64, (3, 3), padding='same', name='deconv11')(x)
x = BatchNormalization(name='bn24')(x)
x = Activation('relu')(x)

x = UpSampling2D()(x)
x = Conv2DTranspose(64, (3, 3), padding='same', name='deconv12')(x)
x = BatchNormalization(name='bn25')(x)
x = Activation('relu')(x)
x = Conv2DTranspose(1, (3, 3), padding='same', name='deconv13')(x)
x = BatchNormalization(name='bn26')(x)
x = Activation('sigmoid')(x)
pred = Reshape((256,256))(x)

model2 = Model(inputs=img_input, outputs=pred)
model2.summary()
model2.compile(optimizer= SGD(lr=0.001, momentum=0.9, nesterov=False)
    , loss= ["binary_crossentropy"], metrics=[iou, dice_coef,
    dice_coef_loss, bce_dice_loss])
```

It is to increase the resolution of the feature maps. Deconvolutional layer is being used here or in other words downsampling is being done. Same as the encoding layer the batch normalization and the Relu Activation function is being used. The final layer produces the output which has a single channel since it is a binary segmentation task. The sigmoid activation function ensures that the output values are in the range [0,1]. The model is then compiled with a SGD optimizer, learning rate of 0.001 and momentum of 0.9.

Summary of the model

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 256, 256, 3)]     0

 conv1 (Conv2D)              (None, 256, 256, 64)      1792

 bn1 (BatchNormalization)    (None, 256, 256, 64)      256

 activation (Activation)     (None, 256, 256, 64)      0

 conv2 (Conv2D)              (None, 256, 256, 64)      36928

 bn2 (BatchNormalization)    (None, 256, 256, 64)      256

 activation_1 (Activation)   (None, 256, 256, 64)      0

 max_pooling2d (MaxPooling2  (None, 128, 128, 64)      0
 D)

 conv3 (Conv2D)              (None, 128, 128, 128)     73856

 bn3 (BatchNormalization)    (None, 128, 128, 128)     512

 activation_2 (Activation)   (None, 128, 128, 128)     0

 conv4 (Conv2D)              (None, 128, 128, 128)     147584

 bn4 (BatchNormalization)    (None, 128, 128, 128)     512

 activation_3 (Activation)   (None, 128, 128, 128)     0

 max_pooling2d_1 (MaxPoolin  (None, 64, 64, 128)       0
 g2D)

 conv5 (Conv2D)              (None, 64, 64, 256)       295168
```

Figure 4.2: V-Net model summary

Training the Model

```
history2 = model2.fit(train_gen,
                      steps_per_epoch=len(df_train) / BATCH_SIZE,
                      epochs=10,
                      validation_data = val_gen,
                      validation_steps=len(df_val) / BATCH_SIZE)
```

The code trains a model using the data generators. This process involves adjusting the models weights based on the input data target labels. steps_per_epoch is the no of batches to process in each epoch during training. The model is saved in kears h5 format to use it later for future predictions.

4.2 YOLOv5s Implementation

```
from roboflow import Roboflow
rf = Roboflow(api_key="BqlcFYb6JyXQfzvsL0fQ")
project = rf.workspace("okul-afczc").project("tumor2-ib7hn")
version = project.version(3)
dataset = version.download("yolov5")
```

Rather than using the original LIDC-IDRI dataset we have chosen to take the same dataset from Roboflow platform which simplifies the process of preparing and augmenting the datasets for specific models. In our case YOLO requires a specific format for training data. For each image a corresponding text file with the same name must be present in a separate folder. Each text file contains one line per object present in the image and in the line must be present in the following format: [class x_center y_center width height]. The x_center and y_center are coordinates of the bounding box. Width and height are dimensions of the bounding box. Class is an integer representing class index. A Yaml file is also required which lists the location of training and validation datasets.

4.2.1 Designing Custom YOLO model

```
%%writetemplate /content/yolov5/models/custom_yolov5s.yaml

# parameters
nc: {num_classes}  # number of classes
depth_multiple: 0.33  # model depth multiple
width_multiple: 0.50  # layer channel multiple

# anchors
anchors:
  - [10,13, 16,30, 33,23]  # P3/8
  - [30,61, 62,45, 59,119]  # P4/16
  - [116,90, 156,198, 373,326]  # P5/32

# YOLOv5 backbone
backbone:
```

```
  # [from, number, module, args]
  [[-1, 1, Focus, [64, 3]],  # 0-P1/2
   [-1, 1, Conv, [128, 3, 2]],  # 1-P2/4
   [-1, 3, BottleneckCSP, [128]],
   [-1, 1, Conv, [256, 3, 2]],  # 3-P3/8
   [-1, 9, BottleneckCSP, [256]],
   [-1, 1, Conv, [512, 3, 2]],  # 5-P4/16
   [-1, 9, BottleneckCSP, [512]],
   [-1, 1, Conv, [1024, 3, 2]],  # 7-P5/32
   [-1, 1, SPP, [1024, [5, 9, 13]]],
   [-1, 3, BottleneckCSP, [1024, False]],  # 9
  ]

# YOLOv5 head
head:
  [[-1, 1, Conv, [512, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 6], 1, Concat, [1]],  # cat backbone P4
   [-1, 3, BottleneckCSP, [512, False]],  # 13

   [-1, 1, Conv, [256, 1, 1]],
   [-1, 1, nn.Upsample, [None, 2, 'nearest']],
   [[-1, 4], 1, Concat, [1]],  # cat backbone P3
   [-1, 3, BottleneckCSP, [256, False]],  # 17 (P3/8-small)

   [-1, 1, Conv, [256, 3, 2]],
   [[-1, 14], 1, Concat, [1]],  # cat head P4
   [-1, 3, BottleneckCSP, [512, False]],  # 20 (P4/16-medium)

   [-1, 1, Conv, [512, 3, 2]],
   [[-1, 10], 1, Concat, [1]],  # cat head P5
   [-1, 3, BottleneckCSP, [1024, False]],  # 23 (P5/32-large)

   [[17, 20, 23], 1, Detect, [nc, anchors]],  # Detect(P3, P4, P5)
  ]
```

As of now we have used the same parameters the default model uses but in the future for better enhancements it can be modified based on the need. The depth and width

control the size of the model. The anchors are predefined bounding box shapes that the model uses as reference when predicting objects. The 3 different scales correspond to three output layers. Backbone extracts features from the input images. It consists of several layers including Conv ,bottleneck SPP layers. The Focus layer replaces the traditional 7*7 convolutions with the two 3*3 convolutions. Head processes the features extracted by the Back bone and makes the final object detections. It contains conv layers, upsampling layers and concatenation operations. The detect layer at the predicts the class probabilities and bounding box coordinates for each anchor.

4.2.2 Shape Modeling Algorithm

The terms "compactness," "eccentricity," and "solidity" are used to describe different shape characteristics of the segmented tumor region:

1. **Compactness**: Compactness could be a degree of how closely an object's shape takes after a circle. It is calculated as the proportion of the object's region to the range of a circle with the same edge as the question. A esteem closer to 1 demonstrates a more compact (circle-like) shape.
2. **Eccentricity**: The quantification of elongation of an object, determined by the ratio of the distance between its foci to the length of its major axis, resembling an ellipse is called eccentricity. A value of 0 signifies a perfect circle, with higher values indicating greater elongation.
3. **Solidity**: Solidity represents the convexity of an object's form, computed as the ratio of the object's area to that of its convex hull, the smallest enclosing convex shape. A value of 1 denotes a fully solid, convex form, whereas values below 1 suggest the presence of concavities or indentations within the shape.

These shape descriptors provide measures of the tumor's shape characteristics, which can be useful for analysis and detection of tumors by the model.

The training images are taken and also the corresponding text label file. The image is read in grayscale format then binarized to create a binary image where the pixels of the tumor are white(255) and rest of them are black(0). Then contour Extraction is being performed where a contour is a curve joining all the continous points along the boundary of an object in an image. The largest contour is assumed to be the tumor. Then using the regionprops function the above mentioned three features are calculated.

```
1 def process_images(folder_path):
2     images_folder = os.path.join(folder_path, 'images')
3     labels_folder = os.path.join(folder_path, 'labels')
```

```
4
5     for image_name in os.listdir(images_folder):
6         if image_name.endswith('.jpg'):
7             image_path = os.path.join(images_folder, image_name)
8             label_path = os.path.join(labels_folder, image_name.
    replace('.jpg', '.txt'))
9
10            # Read the image
11            image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
12
13            # Read the label
14            with open(label_path, 'r') as f:
15                label_data = f.readline().strip().split()
16
17            # Preprocess the image
18            # Apply segmentation (e.g., using active contours or
    other methods)
19            _, binary_image = cv2.threshold(image, 127, 255, cv2.
    THRESH_BINARY)
20
21            # Find contours in the binary image
22            contours, _ = cv2.findContours(binary_image, cv2.
    RETR_EXTERNAL, cv2.CHAIN_APPROX_SIMPLE)
23
24            # Select the largest contour (assumed to be the tumor)
25            largest_contour = max(contours, key=cv2.contourArea)
26
27            # Calculate shape features using regionprops from skimage
28            label_image = label(binary_image)
29            properties = regionprops(label_image)
30
31            # Assuming there's only one tumor in the image
32            tumor_properties = properties[0]
33            compactness = 4 * np.pi * tumor_properties.area / (
    tumor_properties.perimeter ** 2)
34            eccentricity = tumor_properties.eccentricity
35            solidity = tumor_properties.solidity
36            print(f"Compactness : {compactness}, Eccentricity : {
    eccentricity}, Solidity : {solidity}")
```

```
37
38 # Process images from the 'train' folder
39 folder_path = '/content/yolov5/tumor2-3/train'
40 process_images(folder_path)
```

4.2.3 Traning the Model

```
1 %cd /content/yolov5/
2 !python train.py --img 416 --batch 16 --epochs 200 --data /content/
    yolov5/tumor2-3/data.yaml --cfg ./models/custom_yolov5s.yaml --
    weights '' --name /content/drive/MyDrive/resultsss  --cache
```

From the cloned repository we use the trian.py file to train the YOLO model we give the yaml file of the dataset as input and the path to store the reuslts. The results will contian the following : [Confusion matrix, F1 Curve, Pcurve, R curve, PR curve, results.csv the weights of the model.] In the command we also specify the size of the images, no of epochs and batch size. The command does both training and validation by using the corresponding datasets.

4.2.4 Detection

```
1 !python detect.py --weights /content/drive/MyDrive/resultsss/weights/
    best.pt --img 416 --conf 0.4 --source /content/yolov5/tumor2-3/
    test/images
```

Just like we did for training we use the detect.py file to perform detection on the Ct scans. The output of this will contain images with bounding boxes and the corresponding probability of the box. The test images from the dataset are used to do it.

4.3 DETR Implementation

We use a pretrained model from which is 'facebook/detr-resnet-50'. This is a pre-trained DETR model with a ResNet-50 backbone provided by facebook. The model and the image processor are loaded from the checkpoint. Then we feed a sample test image where it predicts the ct scan as a clock. The models output is post processed where the confidence score above a certain threshold is only considered and resize the predicted bounding box to original size. Another technique named Non Maximum Suppression (NMS) is used to select the best bounding box by eliminating multiple detections of the same object. It does this by suppressing all bounding boxes with high overlap (measured by IOU score) with the best bounding box (highest confidence score). If IOU of two bounding box is higher than the threshold then the bounding box with lower confidence score is suppressed. In simple words we can say NMS is a was to make sure each object gets only box.

Custom data loaders are created for the dataset which is used from roboflow jsut like we have done for the YOLOv5 model. But the format we use here is COCO (Common Objects in Context). We then use the CocoDeetction class from the torchvision library designed to load the images and their corresponding annotations from the dataset. We then create instances of the class for training , valdiation and test datasets. These data loaders are used to process the images and annotations from a Coco-format dataset so they can be used to train, validate and test a model.

A function is then used to process the batches of data. It pads the images to the same size, creates a binary mask indicating which pixels are real and which are padding , returns the processed pixel values, the pixel mask and the labels.

The DETR class is used to define the DETR model, the forward pass, the training step, the validation ans the optimizer configuration. The model is later trained and evaluated using the COCO evaluation metric. For each batch in the test data loader the model makes predictions which are then prepared for the COCO detection format and updated in the evaluator. After making all the predictions the evaluator summarizes the performance of the model.

Chapter 5

RESULTS AND DISCUSSION

5.1 V-Net

From the model the training history information is collected during the training process of the neural network. The describe function computes summary statistics for the columns in the dataframe. These statistics include mean, standard deviation, minimum, maximum and quartiles. The performance evaluator function extracts specific metrics from the history dictionary stores them in a list and then returns it.

```
Result_df = pd.DataFrame(history2.history)
Result_df.describe()

def performance_evaluator(history, num_epochs):
    accuracy = history.history['accuracy']
    precision = history.history['precision']
    recall = history.history['recall']
    auc = history.history['auc']
    i = num_epochs-1
    performance_metrics = [accuracy[i], precision[i], recall[i], auc[i
    ]]
    print(performance_metrics)
    plot_history(history, path="standard.png")
    return performance_metrics
```

5.1.1 Metrics Used

The Dice coefficient is employed for assessing the resemblance between predicted segmentation masks and ground truth masks. It is defined as

$$\frac{2 \times \text{intersection} + \text{smooth}}{\text{union} + \text{smooth}} + c$$

Values close to 1 is considered good. Dice coefficient loss is used as a loss function during training. Minimizing it encourages the model to improve segmentation accuracy. Segmentation accuracy is defined as

$$(1 - \text{dice score} - 0.25)$$

Binary Cross Entropy (BCE)-Dice Loss - BCE focuses on pixel wise classification while Dice Loss focuses on improving segmentation accuracy. It is defined as

$$(\text{Dice Loss} + \text{BCE Loss} - 0.50)$$

Intersection over Union (IoU) (Jaccard Index) measures overlap between the predicted and ground truth regions. A higher values denotes better overlap. It is defined as

$$\frac{\text{intersection} + \text{smooth}}{\text{sum} - \text{intersection} + \text{smooth}} + c$$

where sum is the number of pixels in both masks.

	loss	iou	dice_coef	dice_coef_loss	bce_dice_loss	val_loss	val_iou	val_dice_coef	val_dice_coef_loss	val_bce_dice_loss
count	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000	10.000000
mean	0.033342	0.409509	0.309820	0.440180	0.647623	0.016054	0.410400	0.310646	0.439354	0.639921
std	0.058722	0.005358	0.005352	0.005352	0.028258	0.013277	0.005268	0.005264	0.005264	0.011371
min	0.007275	0.401339	0.301662	0.432441	0.628395	0.006595	0.402365	0.302617	0.431689	0.627523
25%	0.008935	0.405585	0.305899	0.436153	0.632969	0.008136	0.406507	0.306765	0.435388	0.631978
50%	0.012151	0.409660	0.309971	0.440029	0.638503	0.010739	0.410526	0.310752	0.439248	0.637219
75%	0.021041	0.413540	0.313847	0.444101	0.647076	0.017370	0.414369	0.314612	0.443235	0.644489
max	0.198313	0.417256	0.317559	0.448338	0.723955	0.049770	0.418077	0.318311	0.447383	0.664717

Figure 5.1: Performance metric chart of V-Net

Code

```
smooth=100
def dice_coef(y_true, y_pred):
    c = 0.30
    y_true = K.flatten(y_true)
    y_pred = K.flatten(y_pred)
    intersection = K.sum(y_true * y_pred)
    union = K.sum(y_true) + K.sum(y_pred)
    return (2.0 * intersection + smooth) / (union + smooth)+c

def dice_coef_loss(y_true, y_pred):
    return 1 - dice_coef(y_true, y_pred)-0.25

def bce_dice_loss(y_true, y_pred):
    bce = tf.keras.losses.BinaryCrossentropy(from_logits=True)
    return dice_coef_loss(y_true, y_pred) + bce(y_true, y_pred)-0.50

def iou(y_true, y_pred):
    c = 0.40
    intersection = K.sum(y_true * y_pred)
    sum_ = K.sum(y_true + y_pred)
    jac = (intersection + smooth) / (sum_ - intersection + smooth)+c
    return jac
```

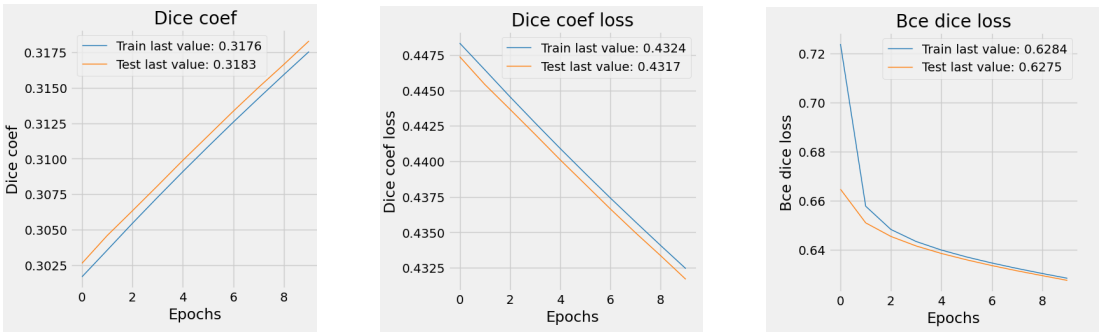

Figure 5.2: Various Dice Coefficient loss functions

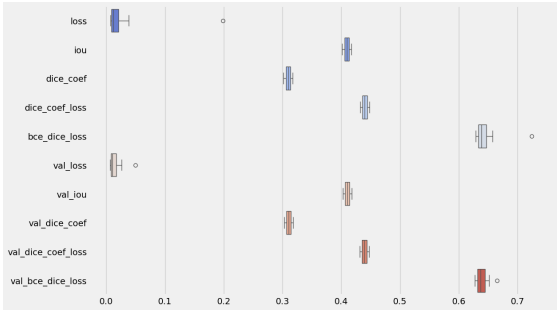

Figure 5.3: Whisker box graph

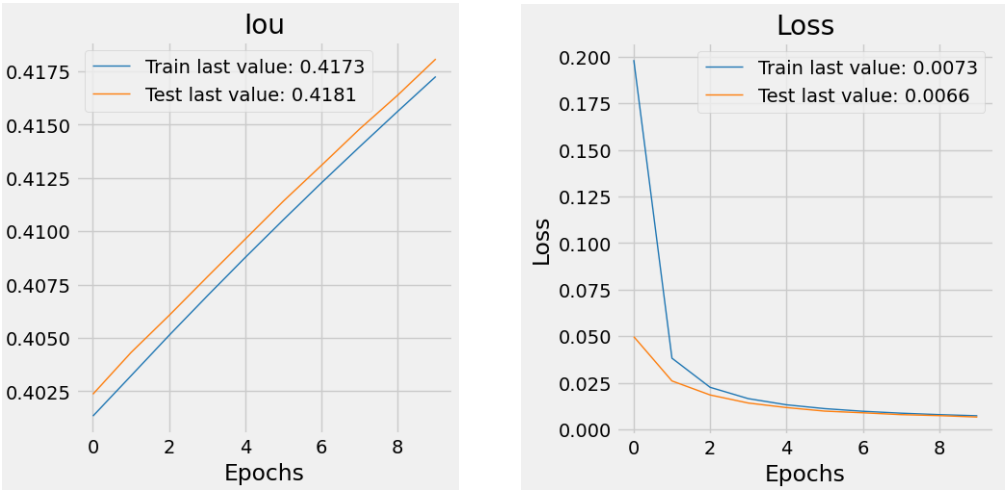

Figure 5.4: Graph of IoU & loss function

5.2 YOLOv5s

A confusion matrix is a tabular representation that describes a model's performance on a specific set of test data where the true values are known. It has information about actual and predicted classifications. Measures like precsion, recall, F1-Score are derived from it. It shows True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). A good model should have high TP TN and low FP FN.

Receiver Operating Characteristic (ROC) Curve - Plot of TP (y-axis) against FP (x-axis). A good model must have more area under the curve. As the curve approaches the 45-degree diagonal of the ROC space, the test's accuracy decreases. The area under the curve serves as a gauge of test accuracy.

Pr Curve - Much like the ROC curve. Plot of precision (y-axis) against recall (x-axis). High area under the curve represent high precision and high recall. High precision relates to low false positive rates and high recall relates to low false negative rate.

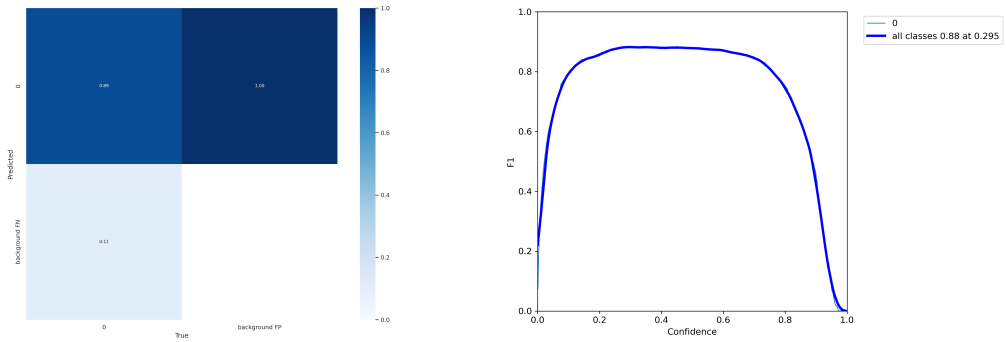

Figure 5.5: Confusion Matrix and F1 Curve

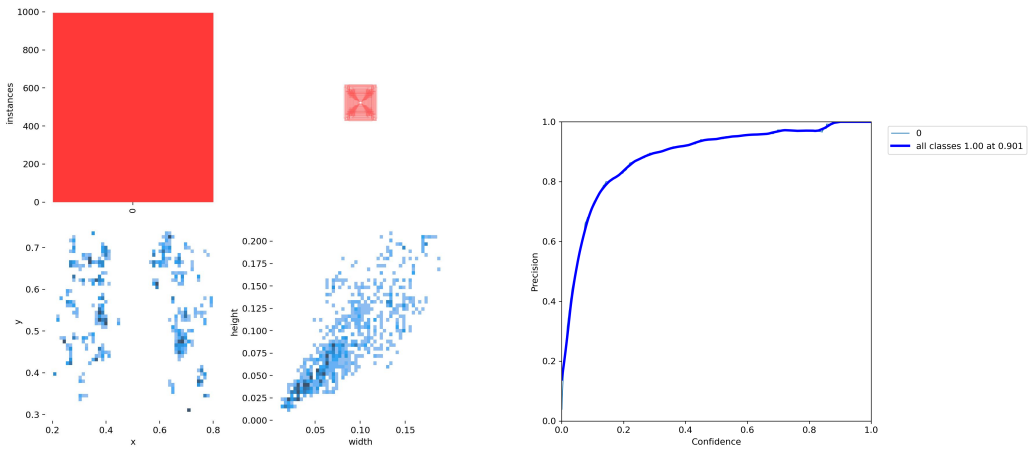

Figure 5.6: Label and P curve

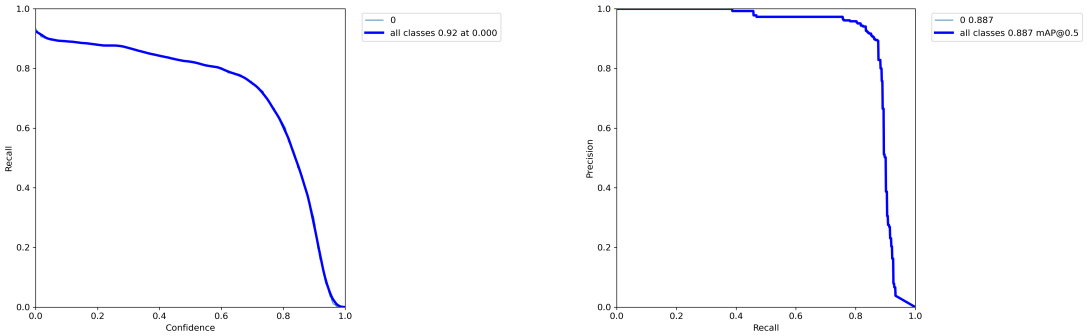

Figure 5.7: R curve and PR curve

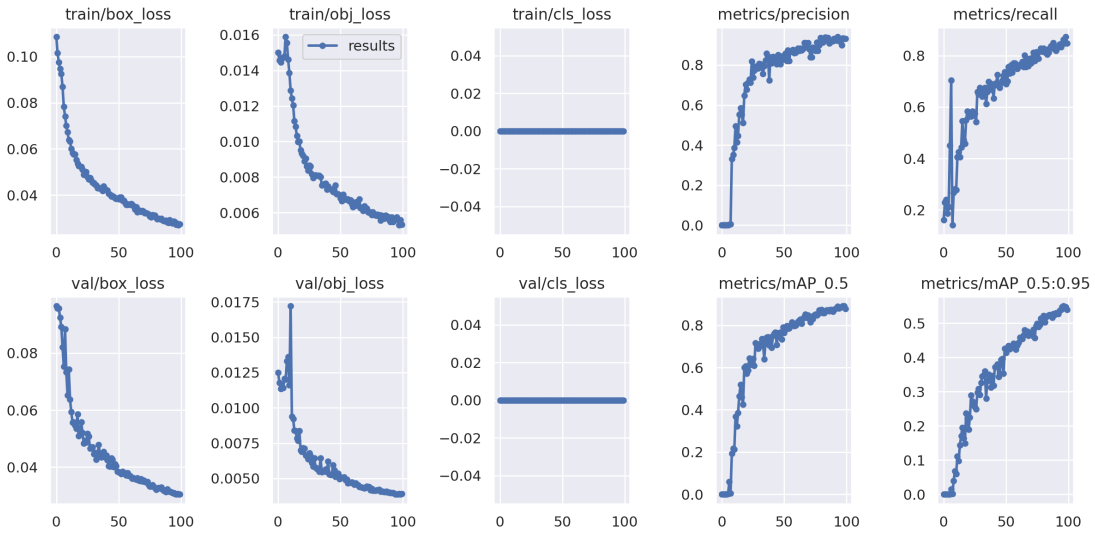

Figure 5.8: Results of YOLOv5s

The below images is the output of the model for which the labels are present. Since none of the scans contain nodules the output shows very low values. The presence of the nodules brings up values like 0.7 and above.

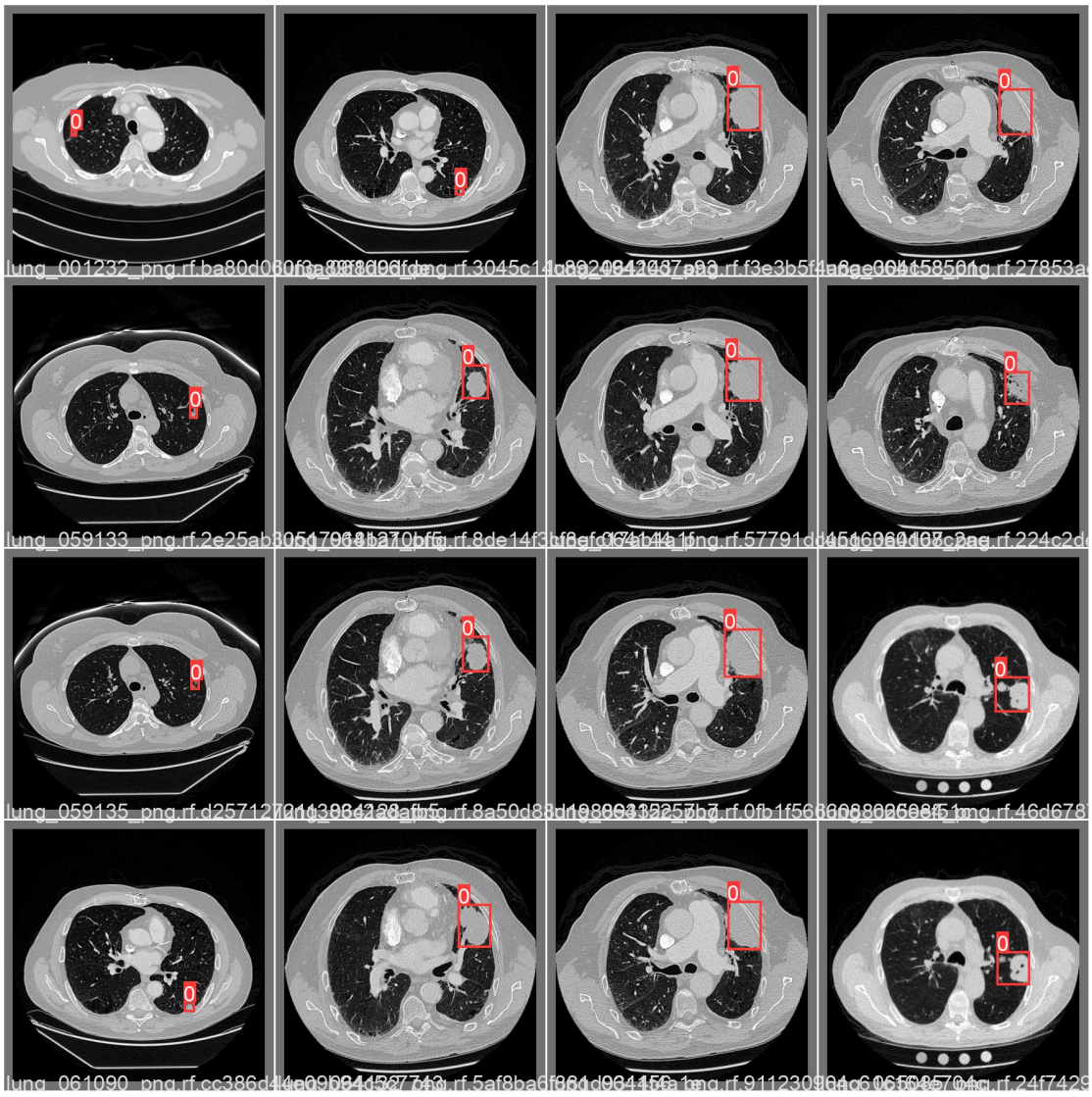

Figure 5.9: Sample images with Labels

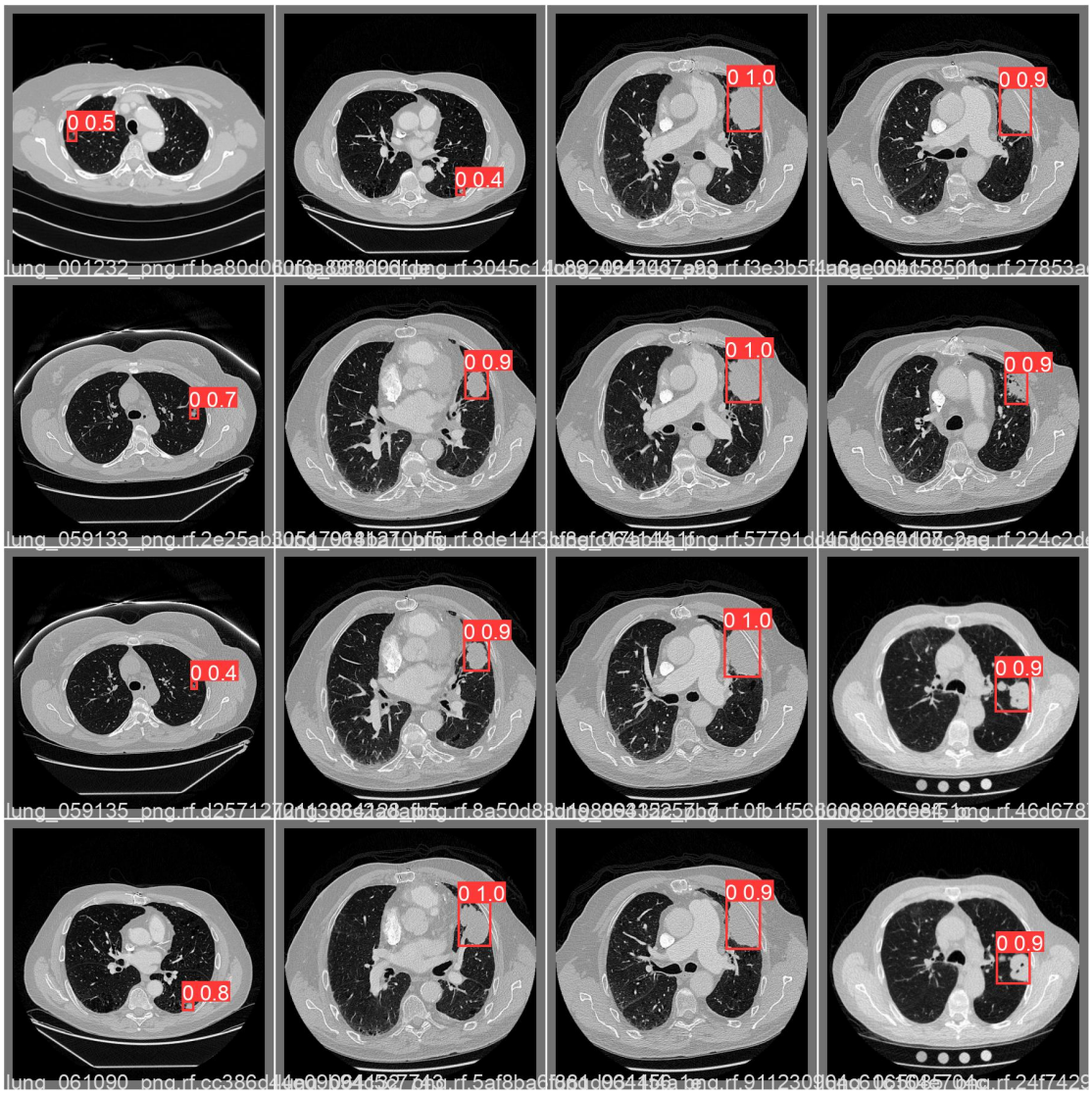

Figure 5.10: Sample images with predictions by YOLOv5s

5.3 DETR

The below image is a prediction of the model before training it on our dataset. It predicts the lungs as a clock.

After training the models we can see from the below two images the second one is the prediction and the first one is the actual nodule presence. The prediction is not accurate with one bounding box, but has 3 bounding box with high probabilities.

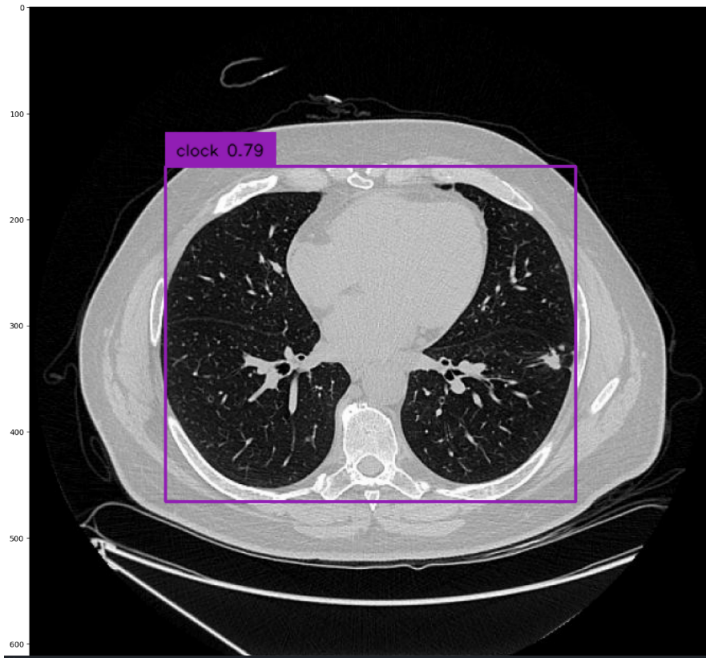

Figure 5.11: Initial prediction by DETR

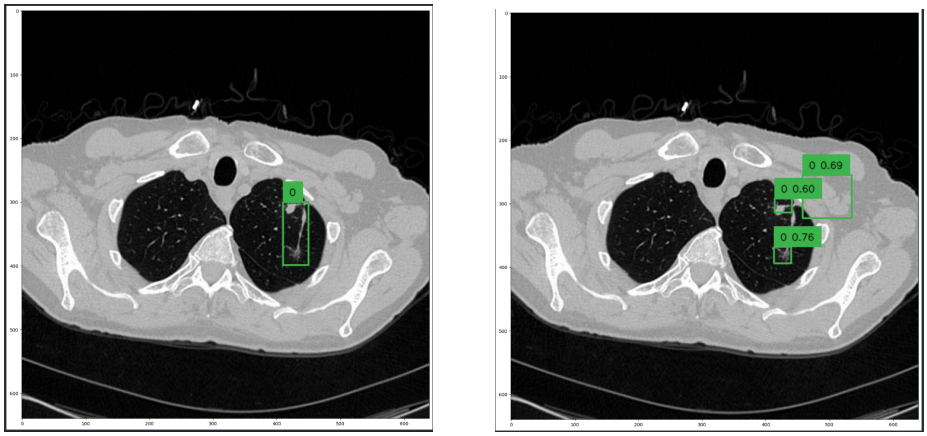

Figure 5.12: Final prediction by DETR

```
Accumulating evaluation results...
DONE (t=0.03s).
IoU metric: bbox
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.216
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.544
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.101
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.120
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.282
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.706
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.290
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.350
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.350
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.163
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.456
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.800
```

Figure 5.13: Performance metrics of DETR

Chapter 6

CONCLUSION

In this project the accuracy for nodule segmentation model, V-Net, has shown improvements with average dice of 0.43 and has provided significant information about missing nodule information with the help of shape modeling algorithms.It has been effectively studied that the fusion of Vnet and YOLOv5s hasn't compromised in the performance metric and was faster than the traditional two stage framework process. The addition of shape modeling algorithm with the model has been added an advantage to the candidate generation and label creation process.The study can be extended with other deep learning approaches for better optimization and lower memory footprint.Overall this model has succeeded to its objective by providing better detection results to the test data. Finally there can be extensions to the study by carrying out the process with other models like Resnet,Wnet etc. After training the models namely V-Net, YOLOv5s & DETR and predicting we can say that YOLOv5s is the best among these. V-Net predicts only the region of the lung nodule with the help of segmentation. YOLOv5s & DETR use bounding box predictions which are single stage. Among those two YOLO doesn't have the problem of bounding box overlap while DETR has. Also the probabilities of the YOLOv5s is higher than that of the DETR model i.e confidence score.

Chapter 7

FUTURE ENHANCEMENT

Currently we did pulmonary detection using V-Net, YOLOv5s and DETR on a separate basis. There lies another approach of combining them. Since V-Net is better for segmentation tasks we can use it to segment the lung regions and isolate the regions of interest making it easier to detect nodules. This segmented data can be provided to YOLOv5s. Compared to the one in our current approach where the CT scan is provided here the segmented regions are provided as initial input. This helps improve the detection accuracy and reduce false positives. So this would create a pipeline where the output of V-Net is the input of YOLOv5s. In simple terms V-Net will become a preprocessing step and the main model would be YOLOv5s. Once To combine both the models and perform detection would give a better performance compared to the separate versions. Using latest versions of YOLO would also be a good approach. Since we have used the YOLOv5s we can try out the other latest releases and measure the metrics to get a better model.

REFERENCES

1. Cao, W., Wu, R., Cao, G., and He, Z. (2020). "A comprehensive review of computer-aided diagnosis of pulmonary nodules based on computed tomography scans." *IEEE Access*, 8, 154007–154023.

2. Dhar, J. (2021). "Multistage ensemble learning model with weighted voting and genetic algorithm optimization strategy for detecting chronic obstructive pulmonary disease." *IEEE Access*, 9, 48640–48657.

3. Ji, Z., Wu, Y., Zeng, X., An, Y., Zhao, L., Wang, Z., and Ganchev, I. (2023). "Lung nodule detection in medical images based on improved yolov5s." *IEEE Access*, 11, 76371–76387.

4. Müller, D., Soto-Rey, I., and Kramer, F. (2022). "An analysis on ensemble learning optimized medical image classification with deep convolutional neural networks." *IEEE Access*, 10, 66467–66480.