LUMBAR DISC HERNIATION AUTOMATIC DETECTION AND CLASSIFICATION BASE ON DEEP LEARNING SAGITTAL VIEW OF MRI

BY

CHAYANIT ARDONK

A PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF BACHELOR OF ENGINEERING IN BIOMEDICAL ENGINEERING
KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG
ACADEMIC YEAR 2023

SCHOOL OF ENGINEERING

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

PROJECT CERTIFICATE

Project Title	Lumbar disc herniation automatic detection and classification base on deep learning using sagittal view of MRI
Student Name	Miss Chayanit Ardonk Student ID. 63011131
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Signed: _____ (Prof. Dr. Chuchart Pintavirooj)
Project Co-Advisor	Signed: _____ (Dr. Siriporn Thitivaraporn)
Committee	Signed: _____ (Prof. Dr. Chuchart Pintavirooj)
Committee	Signed: _____ (Asst. Prof. Dr. Treesukon Treebupachatsakul)

Committee Signed: _____

(Asst. Prof. Dr. Kasama Srirussamee)

Head of Department Signed: _____

(Assoc. Prof. Dr. Sarinporn Visitsattapongs)

Project Title	Lumbar disc herniation automatic detection and classification base on deep learning using sagittal view of MRI
Student Name	Miss Chayanit Ardonk
Degree	Bachelor of Engineering in Biomedical Engineering
Project Advisor	Prof. Dr. Chuchart Pintavirooj
	Dr. Siriporn Thitivaraporn Radiologist
Academic Years	2023

ABSTRACT

Lumbar disc prolapse in the spinal canal is a very common clinical condition which puts the lumbar spine in a position of the highest risk for lower back disorder among patients due to intense pain. In that case, an early and exact diagnostics of disc hernies is one of the fundamentals preconditions of successful diagnosis and treatment methods. To a certain extent; the numbers can be hard to decipher and the process in total quite a laborous, predictably, when it comes to tiny herniations. Thus, it implies the current deep learning models which are the computer-aided diagnosis of disc herniation.

The most common application of deep learning algorithms (YOLOv5 family) is object recognition – among which medical image analysis also belongs to. The algorithms have shown excellent results in many real-life tasks. YOLOV5 is the advanced real time object detecting algorithm that is more popular thanks to high accuracy and fast inference rate being at the large base. Through teaming up the abilities of YOLOv5 with each other, a tool is designed that can contribute more effectively and precisely to identifying lumbar disc hernias for using in MRI scans

The goal of this work is to build son of YOLOv5 model and to use it in order to visualize disc herniations in MRI images. The model will be trained by the repetitive process of viewing datasets of MRI images labelled. Go along with it, it will learn all required capabilities as accurate as it is possible to detect and localize the spinal disc herniations. Closely related to it, the system shows real-time detection of disc herniation and the device with sophisticated features will simplify the diagnosis in each practical setting.

This project intends to solve the problem that radiologists face when interpreting manual MRI images for the purpose of analyzing the presence of a herniated lumbar disc or not. The newly created system, with the application of the new technology, which makes use of the deep learning and the real time detection, is able to upgrade greatly the diagnostic process and increase greatly the accuracy and that results to the fast treatment decision.

ACKNOWLEDGEMENTS

I would like to convey my sincere appreciation to Prof. Dr. Chuchart Pintavirooj, my mentor throughout this project, and Dr. Siriporn Thitivaraporn, a radiologist, who were so generous with their time, attention, and checking the scan readings and tagging of disc herniation. They have been truly a bonafide savior who helped me to complete this project with flying colors.

I am grateful for the opportunity to share my appreciation to the Biomedical Engineering Department, who played a vital role in giving the research facilities and equipment that helped me to complete this project.

Finally, I would like to register my sincere thanks to all persons who in their own capability have contributed to the success of this work. I am especially grateful to those whose names are not mentioned in this thesis. They have made a valuable contribution.

Miss Chayanit Ardonk

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 Clinical problems

Lower back pain is a condition that is often related to the skeletal and muscular system. The word lumbar region defines the discomfort or stiffness in the lower back muscles, which starts from the lower edge of the rib cage (costal margin) and ends at the lower edge of the buttocks (inferior gluteal fold). In some cases, it may also be associated with radiating pain in the leg called as sciatica [1]. Low back pain is one of the most common reasons for work absence and disability globally. According to the claim, low back pain has been identified as one of the most prevalent reasons for seeking medical care in the United States. The fact that 60-80% of all people may experience at least one episode of lower back pain during their lifetime is also implied [2].

In lower back pain patients, disc herniation is one of the most common causative factors. Disc herniation occurs when the soft cushioning discs in between the vertebrae in your spine are ruptured or bulge, resulting in the compression of nearby nerves and subsequently causing pain. Back pain , numbness , weakness and other symptoms are common features of this condition [3].

In recent years, the deep learning algorithms have became the dominant technique in terms of the application in measurement of medical images. Among the different medical imaging modalities, e.g. MRI, and deep learning which proves to be effective and attain good precision. The most frequently used method of doctor examination of the spine pathology is the Magnetic Resonance Imaging (MRI) special technique that enables them to identify vertebrae abnormalities or pathology of the spinal cord, and detect disc herniation which frequently causes sciatica,

determines different reasons for low back pain, sciatica and other types of back pain [4]. While an MRI may be successful in detecting a disc herniation, however, the diagnosis may be tricky since the discs are tiny in diameter and time-consuming analysis is required to examine large volumes of MRI images. Actually, the application of deep learning algorithms in the detection process is the main remedy to the problem of spectra overload in the systems. [5] Deep learning method has been used for improving the quality of analysis of MRI images in terms of color and speed. Through training of deep learning models in interpreting MRI images and automatically detecting the presence of disc herniation at lumbar sites with high accuracy is possible.

Deep learning algorithms, one of them being convolutional neural network (CNNs), have an interesting feature that is they can review lots of visual data and draw remarkably complex patterns. Trends of AI advancement across the field have enabled machines to automatically identify the features that composes MRI scans, as well as precisely recognize the undetected changes such as disc herniation that are linked to the disease state. The algorithms working here only pay attention to very small bulges which can slip by even the human eyes sometimes. Deep learning algorithms, especially YOLO-v5 are an important element, when it comes to real-time object detection in AR/VR applications. Compared to the previous methods that may be too demanding in terms of data preparation or being slow in the process, YOLOv5 was built with the speed of detection in perspective. With its architecture, it can thus handle images or video frames by the millisecond, making it right for real-time applications. Consequently, this paper is intending to conduct YOLOv5 model capable of detecting disc herniation and its real-time scans detection compared to image-based outputs. This is very important as far as clinical implications are

concerned because it can compress the diagnosis process, and this time-frame will be extremely delicate for interventions and improving patient care.

1.2 Objective of the study

1.2.1 To develop a customized YOLOv5 model that is specifically designed to detect disc herniation and accurately identify the five lumbar intervertebral discs (L1-L5) in MRI scans.

1.2.2 To achieve high accuracy in detecting disc herniation instances within MRI scans. It should be capable of precisely localizing the affected area, aiding in accurate diagnosis and treatment planning.

1.2.3 To optimize the YOLOv5 model for real-time detection. This will enable the model to analyze MRI scans in real-time, providing instant detection results as the scans are being acquired. Real-time detection can significantly speed up the diagnosis process and facilitate timely medical interventions.

1.3 Scope of the study

1.3.1 Collect a comprehensive dataset of MRI scans with disc herniation.

1.3.2 Developing, customizing, and training the YOLOv5 model specifically for disc herniation detection and localization of the lumbar intervertebral discs.

1.3.3 Assessing the performance of the developed YOLOv5 model through various evaluation metrics such as precision, recall, F1 score, and mean average precision (mAP).

1.3.4 Real-Time Detection: Implementing real-time detection capabilities using the YOLOv5 model, enabling efficient and instantaneous identification of disc herniation and localization of the intervertebral discs during the MRI scan analysis.

1.3.5 Limitations: Identifying and discussing the limitations of the proposed model and its application. This may include challenges related to small or subtle disc herniations and variations in image quality.

CHAPTER 2

LITERATURE REVIEW AND MATERIALS

2.1 Definition of MRI scans

Magnetic Resonance Image (MRI) is a human medical imaging modality that uses a solid magnet, radionic waves, along with the computer to produce very precise and cross-sectional images of the internal organs. It delivers a very distinct, intrusive, and non-invasive visualization of the organs, tissues, and structures, which makes a heavy contribution to the cases of diagnostic and monitoring of a variety of medical conditions.

MRI scans can be very valuable for detecting anomalies in unusual areas involving soft tissues such as the brain, spinal cord, joints, muscles, and organs like the heart, liver and kidneys. The images which are generated contribute to the analysis of structure and function of the imaged section as well as diseases or abnormality which may be present there [6].

2.2 Herniated disc

A disc herniation (also called ruptured or slipped disc) is a condition that occurs when a disc between vertebrae (or spinal bones) of the spine (the rubbery pads that act as shock absorbers) move out of its original position. The discs are positioned between the spinal bones like a cushion to absorb the shock and give the spine the flexibility that it requires. If a disc is herniated, it is not only that the jelly-like material inside the disc pushes though to the outer side of it.

When a disc gets herniated, it can lead to occurrence of various symptoms because it can press nearby spinal nerve. Symptoms could vary in cases of either anterior or posterior disc

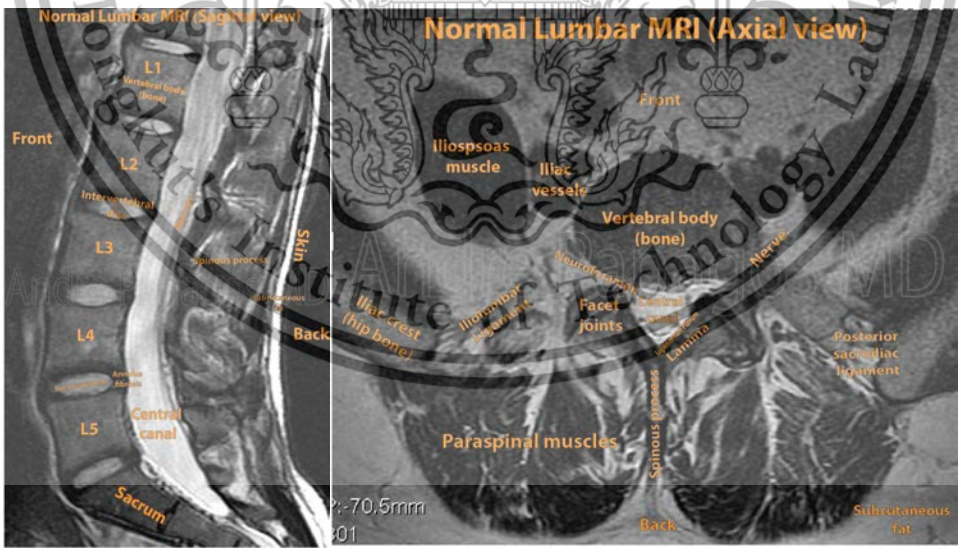
herniation. Subsequent damage may occur if the disc herniates into the spinal cord. One of the common indications that people give about a herniated disc is presented [1].

1. back or neck pain: the herniated disc itself may cause localized pain in the affected area of the spine.
2. Radiculopathy: This term describes the pain that radiates along the path of a compressed nerve. For example, if a herniated disc occurs in the lower back, it can cause pain, numbness, and tingling those travels down the leg, known as sciatica.
3. Muscle weakness: Nerve compression can lead to weakness in the muscles that are supplied by the affected nerve. This weakness may result in difficulty lifting objects, walking, or performing other activities.
4. Changes in sensation: You may experience tingling or numbness in the area supplied by the affected nerve.
5. Bowel or bladder dysfunction: In rare cases where the herniated disc severely compresses the nerves, it can lead to bowel or bladder dysfunction. This requires immediate medical attention.

2.3 Basic Anatomy of lumbar spine of MRI scans

Examining the anatomy of the lumbar spine on the magnetic resonance images (MRI) scans involves identifying the following structures. The key anatomical structures that could be seen on lumbar spine MRI scans include 1) curtains; Lumbar spine consists of 5 vertebrae namely L1 to L5 numbered from top to bottom. These vertebrae play a role in the structural support and the covering which protects the spinal cord and the nerves. 2) Disc; there is a disc between each pair of vertebrae. These sponging discs act as shock absorbers providing cushioning between the

vertebrae. On MRI, the discs appear as black, round structures whose outer part is brighter (annulus fibrosus) than the inner softer part (nucleus pulposus). 3) Spinal cord, it is a long, cylindrical bundle of nerves, which starting from the base of the brain and goes till the lower part of the back. It runs within the protected tube formed by the bony column that surrounds it is. The cord is typically shown as a tubular structure in MRI with the cerebrospinal fluid surrounding it. 4) Nerve Roots; the spinal nerves spring from the spinal cord and exit from the spinal canal through the foramina layers. Through these nerve roots, sensory and motor information travels in the opposite direction from the spinal cord to the body and vice versa. On the MRI, nerves look like ridiculous structures exiting the spinal canal. 5) Facet Joints: These joints, which are known as zygapophyseal joints, are the paired ones seen on the posterior aspect of the vertebrae. They provide steady and flexible movement for the spinal area. On MRI, the facet joints appear as small, smooth, round, or oval structures.6) Muscles and Soft Tissues: An MRI scan may also detect soft tissues adjacent to the spine such as sides paraspinal muscles, ligaments, and blood vessels. (Figure 1)



(A) **(B)**

Figure 1. The anatomical structures of lumbar spine: sagittal view **(A)** axial view **(B)** [7].

2.3.1 Sagittal view of Intervertebral disc herniation MRI

In the process of arising, the grey matter initiates to bubble away from the central canal (D). At the stage of disc degradation, when the wall of the disc is torn, the inside of the disc starts leaking into the central canal leading to an extrusion (E). During a degenerative disc herniation, a small amount of material herniates and can extrude while a larger amount of fluid gets squeezed out and could spread altogether (F). The more the disc material, which moves to the central canal somehow, the more likely than the symptoms, such severe pain, weakness, tingling and numbness, would happen (Figure 2).

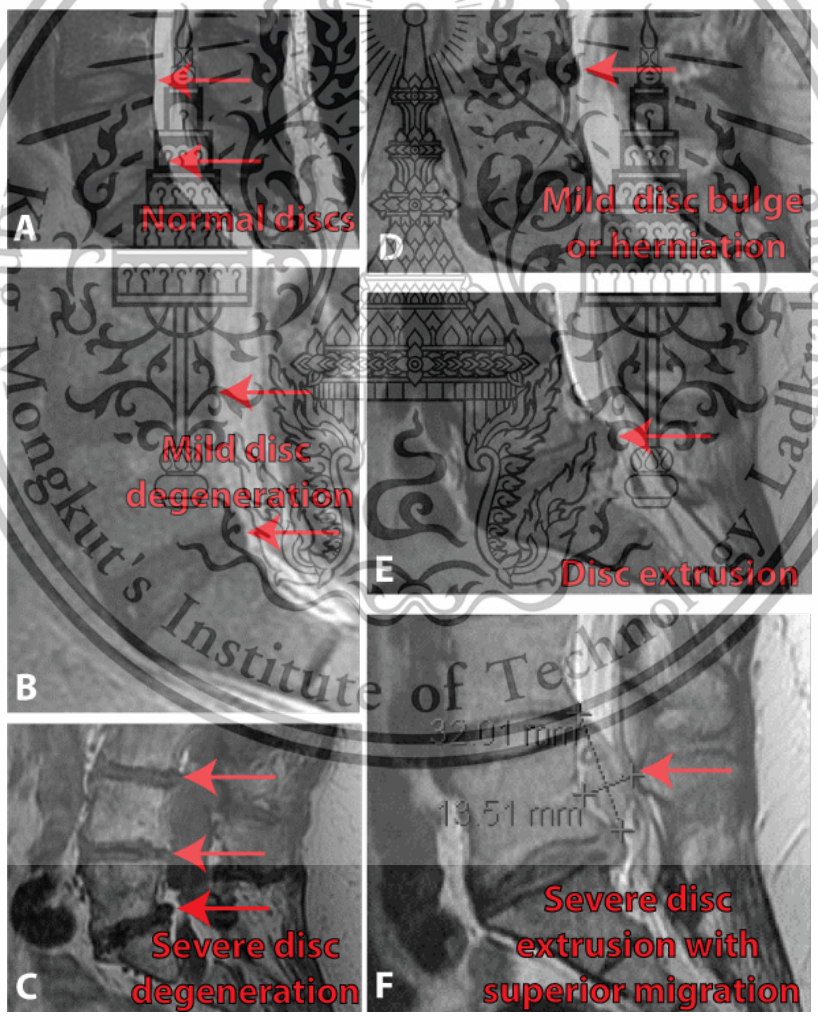


Figure 2. Different types of disc herniation in sagittal view **(A),(B),(C),(D),(E),(F)** [7]

2.4 Deep learning

Deep learning which is a subfield of machine learning addresses the issues of amount and complexity of data by using artificial neural networks that can extract meaningful patterns and features [8]. In medical image analysis, the most important issue is reading images of different angiels which are taken with different angles of view. Nevertheless, the use of deep learning algorithms has proven to be able to solve this problem as they have the ability to learn and recognize any kind of complex patterns, and can make a right diagnosis for it. Machine learning with deep structure classifiers and models are AI-based techniques that grant the AI to detect and arrange elements and patterns from sample data..

2.5 YOLOv5

YoloV5 object detection algorithm is what our team of Ultralytics engineers came up with, and this is a part of the big YOLO family. YOLOv5 has achieved a remarkable gain of popularity by combining both speed and accuracy, which is ushered in starting from June 2020 where it was first launched. YOLOv5's library of models includes YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l, and YOLOv5x, whose depth and the feature map width vary to adequately satisfy different situations. Figure 3 illustrates the network architecture of YOLOv5, which comprises four main components: Backbone, Input, Head, and Neck.[9]

This Input module gets an image and prepares it for better analysis. Here is the process: adaptive image scaling and mosaic data enhancement are among the vital preprocessing steps. In adaptive image scaling, the size of images is normalized to 640 × 640 and, due to mosaic data processing, four images are cropped, scaled and put at random arrangement.

The Backbone bit of the network plays the role of the primary feature selector. It employs convolutional layers for feature extracting from the image inputted. YOLOv5's backbone combines Conv module, C3 module, and SPPF module, which are pretrained on the Imagenet dataset. The Conv block has convolution layer, batch normalization, and SiLU activation. C3 module that employs CSPNet model and has multiple branches acts as a bottleneck. The SPPF module uses the serial maxpooling to perform multi-scale data fusion and create a feature map that increases the receptive field size.

Feature extraction, as done by the backbone, is combined with strong representation learning by the Neck component. YOLOv5 uses the Feature Pyramid Network (FPN) as well as the Pixel Aggregation Network (PAN) structures in the Neck section. The FPN structure carries semantic information form top to bottom via upsampling and the PAN structure does the opposite by transmitting location information from bottom to top via downsampling. The union of these structures offers not only large-scale feature fusion but also the retention of both the large-scale and small-scale target feature information.

The Head component is accountable for forecasting the localization and class probabilities of objects in an input image. The bounding box loss function used here is CIOU loss, and Non-Maximum Suppression (NMS) is utilized to separate multi-object boxes from that of final predicted image.

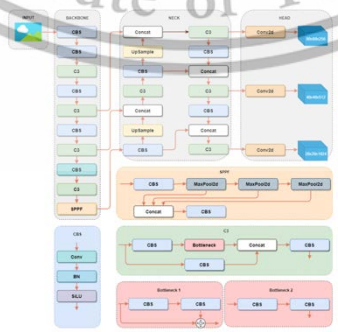


Figure 3 The YOLOv5 structure [15]

2.5.1 Model size

YOLOv5 is available in four different sizes: I size labels the shirts as small (s), medium (m) , large (l) and extra-large (xl) like in 34. The term the size of the model is used to indicate the number of adjustable parameters it has at its disposal. Typically, the models with more parameters cause them to do more efficiently due to the possibility of having many parameters to use for the data While on the other hand, the larger the model the training time grows longer and the inference times also becomes deeper.

If the goals of the system are set up for the real-time feature, it is suggested to choose the small or medium-sized models. These sets of models, however, are fine-tuned and mix speed and functionality, which makes them a really good option for tasks that request fast inference times. The size of specific model is contingent upon the needs of an application and the manner in which the application will exploit the computational resources. Developer can make a choice of the didactic model size, among the other parameters, by considering some issues related to the performance, run time, and available resources [10] .

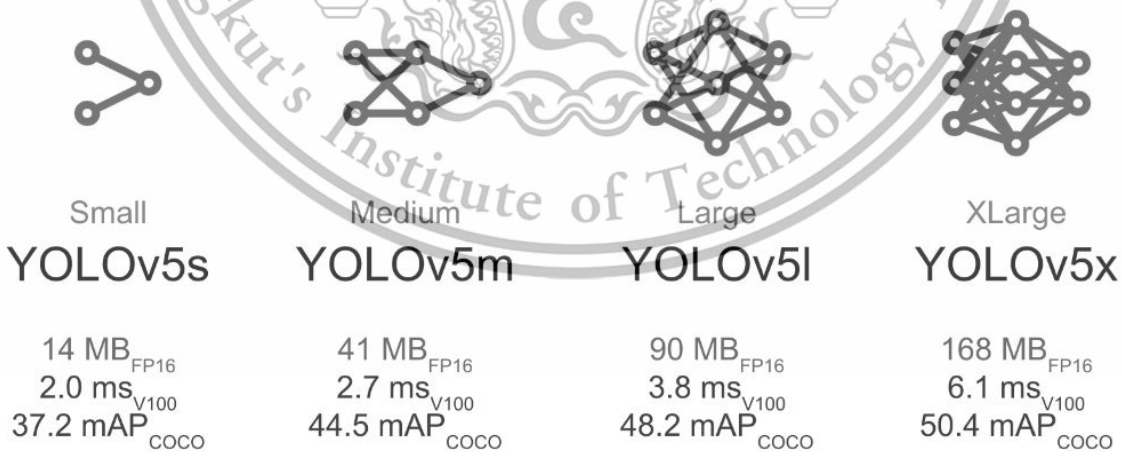


Figure 4 YOLOv5 model size [10]

.

CHAPTER III

METHODOLOGY

3.1 Introduction

This chapter presents the methodology employed for the automatic detection of Lumbar disc herniation (LDH) from sagittal plane MRI scans. The chapter encompasses a detailed description of the various steps involved in the process, including data preprocessing, model training, evaluation, and real-time prediction. Figures, graphs, and data analysis will be incorporated to provide a comprehensive understanding of the methodology. Each step will be thoroughly discussed, highlighting the techniques and algorithms used to enhance the accuracy and efficiency of the LDH detection.

3.2 Material

The original dataset has been collected by Universitas Multimedia Nusantara and Liverpool John Moores University in 2019 and is available on Mendeley Data [9]. It comprises MRI scans of over 500 non-identified patients experiencing low back pain. This data set contains slices of individual images captured in both the sagittal and axial planes for 48345 slices with a 320 x 320 pixels resolution. It provides both T1-weighted and T2-weighted scans. In the sagittal view, all images cover the five lumbar vertebrae (L1, L2, L3, L4, L5) and the sacrum (S1) [11].

3.2.1 Collecting data images.

This study concentrates explicitly on the sagittal view of MRI scans, as this view allows for the visualization of at least the last seven vertebrae and the sacrum. The focus is primarily on

the T2-weighted images obtained from the sagittal view. T2-weighted images are preferred because they enhance the contrast between water (appearing as white) in the central canal and bone (appearing as black). In contrast, T1-weighted images show less pronounced differentiation between water and bone, as both are represented in black tones. The files named "T2_TSE_,SAG_384" in all folders are collected for further analysis. ImageJ software converts these images from the. ima format to .jpeg format as Figure 5 and Figure 6.

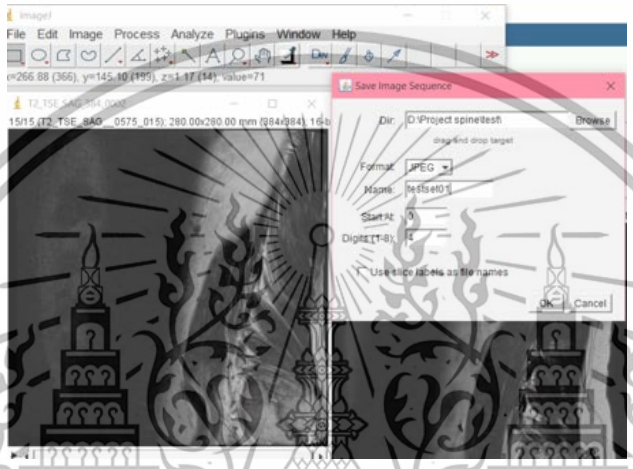


Figure 5 Convert .ima to.jpeg file

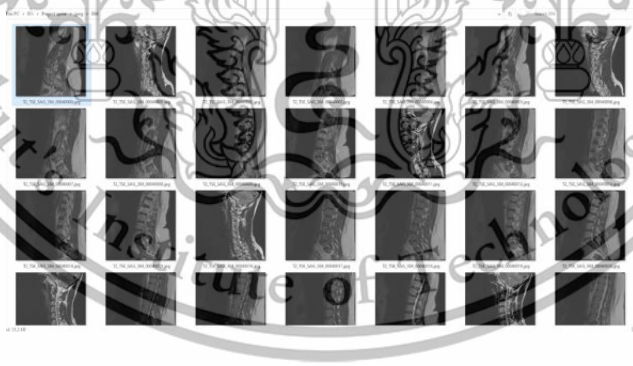


Figure 6 A MRI Lumbar Spine in Sagittal View

3.3 Environment Setup

The model was coded in Python 3.9, some libraries are imperative for its implementation. The libraries that were imported include numpy, pandas, matplotlib, opencv-python, jupyter and labelImg. We used the cloud-based platform, Google Colab, to complete the training, and validating process of the machine learning algorithms.

The PC used in this experiment boasts a MACBOOK PRO 14 model equipped with an M2 CPU, a 16-core GPU, 16GB of RAM, and 512GB SSD, while the mouse and keyboard used have the model of the Attika Blue and Advanced Gaming Keypad. This setting, therefore, is super appropriate for deep learning tasks, offering the necessary processing power with vast room for saving the model's results.

3.4 Data preprocessing

Pre-procession is therefore the ground work and primary step before training models on image data by presenting the data in a form that is suitable to the model's learning. The pictures are auto oriented and resized to uniform unite size=384 x 384 pixels. Making the sure that the model is aligned correctly, and the input is matching and consistent during the entire time of training process. The study provides us with the details of such procedures as image processing and feature extraction that are to be applied prior to training.

3.4.1 Inclusion and Exclusion criteria for MRI scans

The inclusion criteria for the MRIs scans for the image analysis were focused on gathering images that showed anatomical features. To begin with, the mid-line plane where the MRI images can be reliably captured was defined, and this facilitated the clear visualization of the central canal.

In the course of this lumbar puncture (LP), I made it a must to visualize if cerebrospinal fluid (CSF) was present, levels of nerves L1-L5, disc between S1 vertebrae, and sacrum S1. Moreover, another important prerequisite is disk bulge or herniation which has been supposed to be the cause for discogenic pathology (figure 7).

In contrast, the categories of exclusion were determined to prevent picture material that had unclear purpose and was not indispensable. The scarce MRI images that were overtly obscure and did not indicate any particular disc protrusion or the stated inclusion criteria were discarded. MRI slices placed too laterally, or spreading significantly away from the mid-line plain were also discarded if they appear figure 8.

By employing these inclusion and exclusion criteria, the study aimed to ensure that the selected MRI scans provided clear visualization of the relevant anatomical structures and exhibited the pathological condition under investigation. This rigorous selection process aimed to enhance the accuracy and reliability of the study's findings.

Figure 7. Data preparation: Examples of Inclusion image

Figure 8. Data preparation: Examples of Exclusion image

3.4.2 Data labeling and annotation

For labeling and annotation of data, a module found in the Python package- LabelImg, operates beneath a friendly user interface and generates speedy labeling and annotation. It facilitates using box drawing tools to label objects by changing their coordinate values and class descriptions. This study used 6 classes of objects. The five class labels correspond to the intervertebral disc levels starting from L1-2 making L12 right up to L45 making the last L5-S1 for the segments of L1-L2, L2-L3, L3-L4, and the last L4-L5 as L5-S1. The cervical lumbosacral degeneration class is the remaining class label and represents lumbar disc herniation (LDH). The number of cases of LDH (leukoaraiosis) on the respective MRI images varied from at least one instance to a maximum five. Samples of LDH cases were called out and marked properly on superimposed slides; as Figure 9 shows.

It is crucial to exercise caution when drawing bounding boxes during the annotation process to ensure optimal precision and accuracy. Care should be taken to accurately delineate the boundaries of the objects of interest, ensuring that the bounding box size precisely encompasses the entire target object. Inaccurate or imprecise bounding boxes can harm the performance of

subsequent tasks, such as object detection or classification. By meticulously drawing bounding boxes that precisely cover the focused objects, the risk of introducing errors or reducing the overall quality of the annotations is minimized. This attention to detail and accuracy in the annotation process contributes to higher precision and accuracy in subsequent tasks, ultimately enhancing the overall effectiveness of the trained models [12].

The labeling and annotation process was conducted under the supervision of a radiologist and underwent verification to ensure accuracy. To save the annotations, Pascal VOC (Visual Object Classes) format was employed, utilizing XML files [13]. This format captures crucial information such as image paths, class labels, and bounding box coordinates (see Appendix A). By adopting this format, the labeled and annotated data is stored in a structured manner, facilitating seamless integration and utilization for training YOLOv5 models.

Figure 9. Labeling and annotation: L1-2, L2-3, L3-4, L4-5, L5-S1and LDH

3.4.3 Coding and creating function in Python

After completing the labeling and annotation process, it is necessary to read and extract the data information using code written in a Python notebook environment, Jupyter, to prepare it for training in a YOLOv5 model as the following steps (see code Appendix B).

3.4.3.1 Getting a list of XML files in Python

This step is important to identify and retrieve the XML files that contain label information or other relevant data. This step involves using the **glob** module or any suitable method to obtain a list of XML files present in a specific directory. It is necessary to identify and access the XML files that contain the desired information for further processing.

3.4.3.2 Read and extract labels data from XML file

In this step, the XML files are read and parsed to extract the relevant label data. This involves accessing specific XML elements or attributes that contain the desired label information. The **extract_text** function takes an XML file as input, extracts object detection-related information including image name, width and height of the image, name of the object, and the corresponding bounding box coordinates (xmin, xmax, ymin, ymax) and returns this information as a list as Figure 10.

```
[['194m3.jpg', '384', '384', 'L5S', '181', '230', '248', '288'],
 ['194m3.jpg', '384', '384', 'LDH', '227', '244', '245', '262'],
 ['194m3.jpg', '384', '384', 'L45', '170', '226', '204', '227'],
 ['194m3.jpg', '384', '384', 'L34', '170', '223', '150', '170'],
 ['194m3.jpg', '384', '384', 'L23', '174', '225', '94', '117'],
 ['194m3.jpg', '384', '384', 'L12', '188', '232', '45', '67'],
 ['314m2.jpg', '384', '384', 'L5S', '229', '271', '283', '333'],
 ['314m2.jpg', '384', '384', 'L45', '198', '248', '238', '279'],
 ['314m2.jpg', '384', '384', 'L34', '194', '246', '190', '215'],
 ['314m2.jpg', '384', '384', 'L23', '201', '249', '134', '161'],
 ['314m2.jpg', '384', '384', 'L12', '217', '259', '86', '112'],
 ['314m2.jpg', '384', '384', 'LDH', '243', '258', '238', '256'],
 ['377m1.jpg', '384', '384', 'L5S', '242', '281', '275', '320'],
 ['377m1.jpg', '384', '384', 'L45', '232', '276', '235', '262'],
 ['377m1.jpg', '384', '384', 'L34', '232', '277', '186', '210'],
 ['377m1.jpg', '384', '384', 'L23', '239', '284', '137', '162'],
 ['377m1.jpg', '384', '384', 'L12', '254', '298', '87', '115'],
 ['377m1.jpg', '384', '384', 'LDH', '279', '290', '278', '291'],
 ['sag384p0709.jpg', '384', '384', 'L5S', '204', '250', '259', '296'],
```

Figure 10. list of extracted information

3.4.3.3 Convert labels information into pandas DataFrame

The extracted label data from the XML files is converted into a structured format, pandas DataFrame. This allows for easier manipulation, analysis, and transformation of the dat a as figure 11.

	filename	width	height	name	xmin	xmax	ymin	ymax
0	194m3.jpg	384	384	L5S	181	230	248	288
1	194m3.jpg	384	384	LDH	227	244	245	262
2	194m3.jpg	384	384	L45	170	226	204	227
3	194m3.jpg	384	384	L34	170	223	150	170
4	194m3.jpg	384	384	L23	174	225	94	117
5	194m3.jpg	384	384	L12	188	232	45	67
6	314m2.jpg	384	384	L5S	229	271	283	333
7	314m2.jpg	384	384	L45	198	248	238	279
8	314m2.jpg	384	384	L34	194	246	190	215
9	314m2.jpg	384	384	L23	201	249	134	161

Figure 11. Pandas DataFrame data

3.4.3.4 Preparing Label for YOLO model

In YOLO model, the bounding box coordinates are normalized to values between 0 and 1 relative to the image dimensions. This normalization allows YOLO to handle objects of different sizes and aspect ratios effectively. The center point of the bounding box is an important parameter in YOLO because it represents the spatial location of the object within the image. By calculating the center point coordinates (center_x and center_y), it can express the object's position relative to the image size [14]. The center coordinates are obtained by taking the average of the minimum and maximum coordinates in the x and y directions and then dividing them by

the respective image dimensions (width and height). Dividing by the image dimensions ensures that the center coordinates are normalized values between 0 and 1 [15] as the following formulas.

- $center_x = \frac{\frac{x_{min}+x_{max}}{2}}{width\ of\ the\ image}$
- $center_y = \frac{\frac{y_{min}+y_{max}}{2}}{height\ of\ the\ image}$
- $w = \frac{x_{max} - x_{min}}{width\ of\ the\ image}$
- $h = \frac{y_{max} - y_{min}}{height\ of\ the\ image}$

This information is crucial for the subsequent steps of YOLO, such as assigning bounding box predictions and class probabilities to specific grid cells as Figure 12.

	filename	width	height	name	xmin	xmax	ymin	ymax	center_x	center_y	w	h
0	194m3.jpg	384	384	L5S	181	230	248	288	0.535156	0.697917	0.127604	0.104167
1	194m3.jpg	384	384	LDH	227	244	245	262	0.613281	0.660156	0.044271	0.044271
2	194m3.jpg	384	384	L45	170	226	204	227	0.515625	0.561198	0.145833	0.059896
3	194m3.jpg	384	384	L34	170	223	150	170	0.511719	0.416667	0.138021	0.052083
4	194m3.jpg	384	384	L23	174	225	94	117	0.519531	0.274740	0.132812	0.059896

Figure 12. Yolo label Conversion

3.4.3.5 Split data into train and test sets

The dataset, comprising images and the The dataset, comprising images and their corresponding labels, is divided into separate training and testing sets. This partitioning is essential for evaluating the performance of the trained model on unseen data. The code snippet divides a DataFrame containing image filenames into training and test sets. Approximately 80% of the

filenames are randomly selected for the training set, while the remaining 20% are assigned to the test set as Figure 13. This split allows for separate datasets to be used for training and evaluating models or algorithms.

```
1 train_df.head()
```

	filename	width	height	name	xmin	xmax	ymin	ymax	center_x	center_y	w	h
0	194m3.jpg	384	384	L5S	181	230	248	288	0.535156	0.697917	0.127604	0.104167
1	194m3.jpg	384	384	LDH	227	244	245	262	0.613281	0.660156	0.044271	0.044271
2	194m3.jpg	384	384	L45	170	226	204	227	0.515625	0.561198	0.145833	0.059896
3	194m3.jpg	384	384	L34	170	223	150	170	0.511719	0.416667	0.138021	0.052083
4	194m3.jpg	384	384	L23	174	225	94	117	0.519531	0.274740	0.132812	0.059896

```
1 test_df.head()
```

	filename	width	height	name	xmin	xmax	ymin	ymax	center_x	center_y	w	h
6	314m2.jpg	384	384	L5S	229	271	283	333	0.651042	0.802083	0.109375	0.130208
7	314m2.jpg	384	384	L45	198	248	238	279	0.580729	0.673177	0.130208	0.106771
8	314m2.jpg	384	384	L34	194	246	190	215	0.572917	0.527344	0.135417	0.065104
9	314m2.jpg	384	384	L23	201	249	134	161	0.585938	0.384115	0.125000	0.070312
10	314m2.jpg	384	384	L12	217	259	86	112	0.619792	0.257812	0.109375	0.067708

Figure 13. train and test set data

3.4.3.6 Label encoding to objects

In this step, the object labels are encoded or mapped to numerical values. The purpose of label encoding is to convert categorical object names into numeric labels, which can be useful for various tasks such as data analysis, machine learning algorithms, or organizing and representing categorical data in a numerical format. The object names being encoded are: 'LDH', 'L12', 'L23', 'L34', 'L45', and 'L5S'. These names are associated with certain

objects or categories, and the code assigns numeric labels to each of these names as the following numeric labels to the object names

'LDH' is encoded as 0

'L12' is encoded as 1

'L23' is encoded as 2

'L34' is encoded as 3

'L45' is encoded as 4

'L5S' is encoded as 5

3.4.3.7 Create test and train folders

Folders or directories are created to organize the images and label files separately for the training and testing sets. This ensures a structured arrangement of the data for subsequent steps.

3.4.3.8 Create functions to move train and test images and label text in their folder

A train folder function is created to automate the process of moving the training images and their corresponding label files into the designated training folder. This helps to streamline the data preparation workflow and ensure consistency. As for a test folder function is created to move the testing images and their associated label files into the designated testing folder. This helps to organize the data and facilitate the evaluation of the model's performance on unseen data.

3.5 Training YOLO Model

For training YOLOv5 model. The first crucial step is to obtain the YAML file. The YAML file works as a setup file that tells the training process different options, parameters, and other settlings. The help section indicates the classes that are needed for the YOLOv5 model to know how to do training. There is a train file, validate file, six classes and titles. The names in this list designated as 'LDH', 'L12', 'L23', 'L34', 'L45', and 'L5S'. Each name was encoded using the Label encoding process, leading to a unique numberic code for each label.

Besides of that, Google Colab is used to train the YOLOv5 model as the favorite training platform. Ensuring accessibility to the YOLOv5 repository is crucial, which can be achieved by downloading it from the provided link: https://github.com/ultralytics/yolov5.gho Furthermore, students should ensure they have created the missing YAML file and folders (train and test) at the specified location within the Google Drive cloud. Thus, these fundamental actions are of great significance for the successful execution of the training.

After that step comes running the program (see Appendix C). The YOLOv5 model will go through the training stage using our provided dataset and settings. While training machine will be learning to recognize and localize the objects in the images through the use of provided annotation. One hundred epochs stand - the number of the whole dataset a model does through it. More epochs creates opportunities for model to assimilate more patterns and thus continue on a decent learning path until 100. The byproduct of that training process is an optimal YOLOv5 model, which is able to make the object detection from new, not seen images right.

3.5.1 Model Evaluation

Several evaluation metrics and techniques were employed to evaluate the performance of the YOLOv5 model as the following precision, recall, accuracy, F1 score and mean average precision (mAP).

3.6 Prediction and Live Testing from YOLOv5 Model

This process involves utilizing trained models, YOLOv5, to analyze input data and generate predictions in the form of bounding boxes around the detected objects, along with corresponding class labels and confidence scores within an image or video. These predictions help to understand and interpret the visual content, automate tasks, extract meaningful information, and make informed decisions based on the detected objects. To generate prediction, the following steps are involved (see code in Appendix D).

3.6.1 Load the trained model.

The first is to load the YAML file. The class labels are extracted and used in subsequent steps on the model's predictions by accessing the' names' field. Secondly, the YOLO model is being loaded using the OpenCV library's deep neural network module (cv2.dnn), setting the preferable backend, and target device, this step is necessary before using the model to perform object detection on images or videos.

3.6.2 Prepare the input data.

By loading an input image that make the prediction on and converting the image into the square image (array). Converting the image into a square matrix ensures that the input to the YOLO

model has consistent dimensions. It is important because YOLO expects the input image to be in a specific size and aspect ratio to produce accurate predictions.

3.6.3 Getting predictions from the YOLO model

YOLO prediction comes from the model by using a square image array. yolo.setInput(blob): Use the blob, that is array of squares (nxn) as the input for YOLO model. The command preds = yolo.forward() computes the forward passes of the YOLO neural network to get the model outputs. Therefore, the function forward () makes the computations for the predictions resulting from the input. After the detection process, the preds vector contains the objects that were detected, with the box coordinates, accuracy scores, and class labels being among the variables.

Following the output line, print(preds.shape) displays the dimensions of the Preds array. Here, the shape of the data of age is (1, 25200, 11) the Figure 14. The axis with samples' or images' number at the left end represents the first dimension. Here are the two options: either one photo or other types of samples therefore size becomes 1. The second axis can be used to show the number of bounding boxes or detections which is the maximum number. YOLO is of size 25200 which means, it can detect up to 25200 image boundaries on the pictured image. The third dimension is related to the number of components and values that exist for each searched area. This is what happened next. The data consists of 11 elements and the first five of these are boxes coordinates which are in the format (center x, center y, w, h, confidence scores), and the next six elements are class probabilities corresponding to the names of the six objects (L1, L23, L34, L45, L5S, LDH).

```
1 print(preds.shape)
```

```
(1, 25200, 11)
```

Figure 14. Prediction shape

3.6.4 Non-Maximum Suppression

Non-Maximum Suppression (NMS) is a post-processing method used in object detection assignments whereby redundant and overlapping bounding boxes that give out low-confident and poor detection accuracy are deleted to enhance the accuracy and speed of object detection algorithms by choosing the most confident and non-overlapping bounding box predictions. The format (1, 25200, 11) clearly shows that the model is having 25,200 bounding box predictions and each is represented by the 11 elements. Use NMS on the set of predicted bounding boxes to get rid of all overlapping and repeated detections.

Step 3 involves all the detections with a confidence score less than 0.4 or a maximum class probability less than 0.25. It then uses NMS to eliminate the overlaps among the boxes and finally, it outputs a unique and confident detections.

NMS produces its final index array with the indices of selected bounding boxes as represented in Figure 15. Numbers [140, 132, 21, 127, 95, 13, 78, 92 and 61] are the indices of the bounding boxes that are selected after the Non-Maximum Suppression (NMS) algorithm is applied. These indices denote the places in the initial set of probable detections that have been selected to finalize the non-overlapping and confident object detections as a disjoint set.

Every number in arrays represents the corresponding bounding box of the initial list of detection outcomes. These Indices in turn can be used to fetch bounding boxes from the original list for the processing or visualization.

```
1 index
2
```

```
array([140, 132,  21, 127,  95,  13,  78,  92,  61],
```

Figure 15. The indices after NMS

3.6.5 Drawing the bounding box for model predictions.

This step involves drawing bounding boxes around the detected objects in the image and displaying the corresponding class name and confidence score percentage. This help in interpretating and understanding of the object detection algorithm's outcomes. The code generates green color of bounding boxes, pink-colored text for the class labels, with a thickness of 1, displayed on a white rectangular background.

3.6.6 Object detection from the image with YOLO

The final object detection results for a sample image are displayed in Figure 16. (a) represents the original image, while (b) showcases the image with object predictions generated by YOLO.

(a) (b)

Figure 16. Object detection from the image with YOLO; a) original image b) object predictions

3.7 Real Time Objects Detection with YOLOv5

3.7.1 Create YOLO prediction module.

The YOLO prediction module in Python facilitates real-time object detection using YOLO. It can be easily imported and integrated into various projects, enabling convenient usage and seamless integration of YOLO-based object detection functionalities (see code Appendix E).

The video showcasing MRI slices is played in a sequence starting from the most lateral position and progressing towards the other most lateral position. These slices are utilized as input for the YOLO model, enabling real-time object detection throughout the duration of the video. The detection results are displayed by drawing bounding boxes around the detected objects. Each object is represented by a distinct color, with the LDH objects identified by the green bounding boxes as Figure 17. Link for viewing real-time detection is below:

https://drive.google.com/file/d/1Y8M6rZShJ_XYWw-P4AkvdrlIpyViTunH/view?usp=sharing

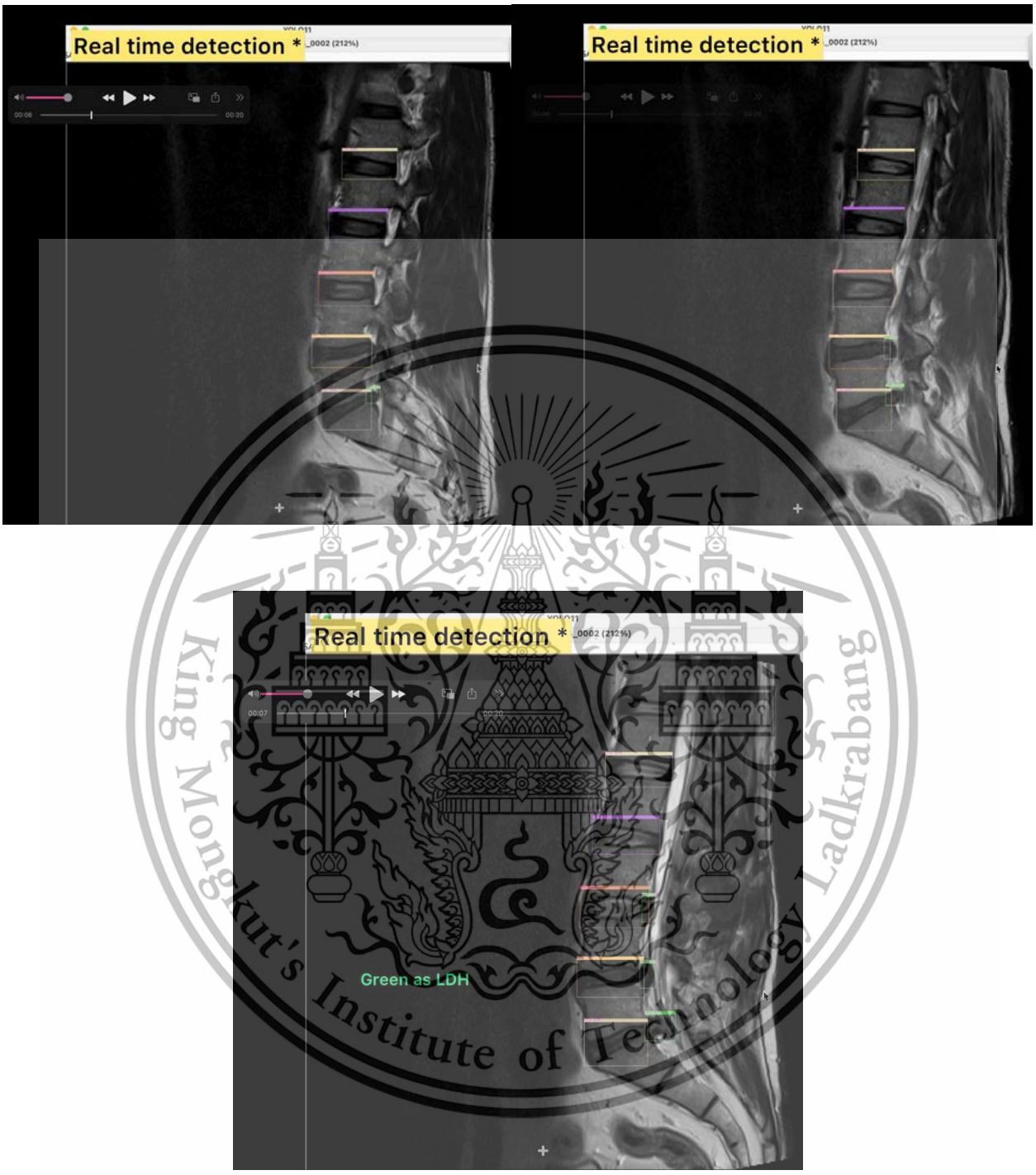


Figure 17. Example captures of the real-time detection video.

CHAPTER 4

EXPERIMENTAL RESULT

4.1 Introduction

This chapter focuses on the results of the trained YOLOv5 model and the evaluation of its performance using various metric graphs, including box losses. The metric graphs will illustrate the progression of the model during training, highlighting the changes in box losses over time. Additionally, the chapter will provide visual representations of the figures and graphs, allowing for a better understanding of the training process and the model's performance. Through the analysis of these metric graphs and figures, valuable insights into the effectiveness and accuracy of the YOLOv5 model will be gained, contributing to the overall evaluation and validation of the approach.

4.2 Datasets

The study was based on the MRI dataset that was T2-weighted, from the midline of the low back region, precisely looking at the placement of six-seven low lumbar vertebrae, inclusive of the sacrum. The main purpose of this dataset was to build a model of YOLO that would be be effective enough to detect disc herniation. The data was distributed into set training and test which were 80% and 20%, respectively. Labelimg package written in python was used since the output format. It is .txt file, which is the custom format of YOLOv5 PyTorch. We have compiled 2123 annotations in total, each image annotated with a nearly 7.07 average of annotation. There were 623 instances of attended classes for the LDH, while the L12, L23, L34, L45, and L5S had 299, 302, 300, 299, and 300 cases, respectively. An X graph in the Figure shows an annotation annotation per class proportional distribution in the dataset. Despite the lack of L2LDH information, around 300 instances had low dietary heterogeneity. However, all the study relied on

from the number of L5S and it still presented 600 instances of LDH. The disproportionate importance to LDH can be explained through the purposes of of the research that is aimed for visual identification lumbar disc herniation because it is a small portion of image and has the difficulty identification. The figure shown in 18 (B) shows a statistical illustration of bounding box distribution within the dataset, it indicates where each bounding box is located and the size for each bounding box. So as to beef up the white areas may it be objects positions and sizes, there should be enough variability for better model recognition. Figures 18 (C) and (D) for tightening box position and size distributions are shown in Figure 19. It is clearly seen where bounding boxes occur in the dataset. This analysis helps identify any potential biases or imbalances within the dataset, ensuring that the model can accurately detect objects regardless of their size or position. Such considerations are essential to ensure the model's robustness and generalizability.

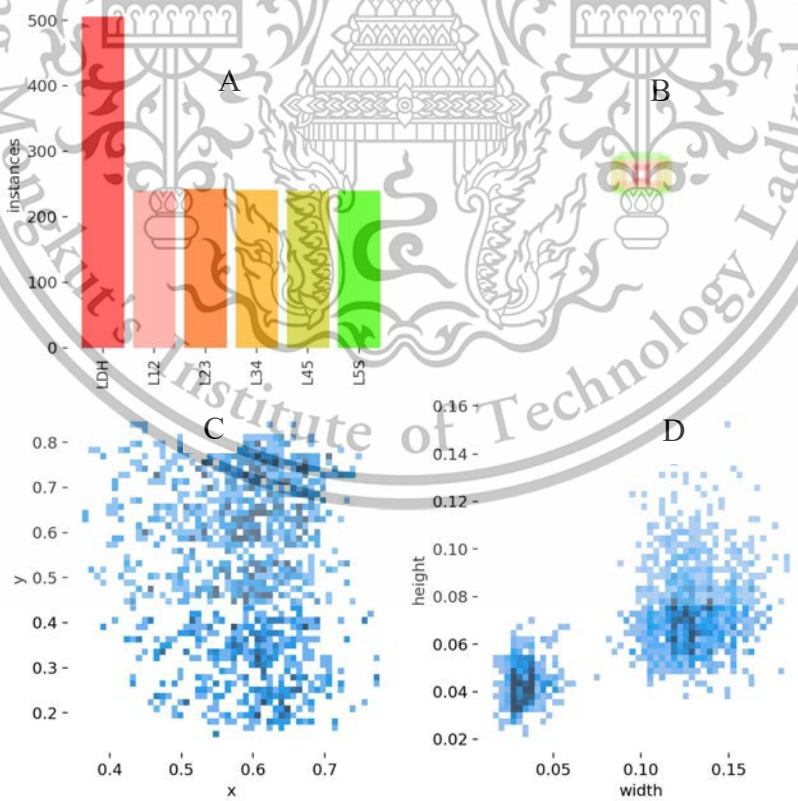


Figure 18. Visualized Dataset Results

4.3 Training results

4.3.1 Testing batch

This section presents the results obtained from evaluating the YOLOv5 model for object detection. The predicted batches of the YOLOv5 comprise the predicted bounding boxes, class labels, and confidence scores produced. These predictions represent the model's estimations of object positions and classes within the respective validation or testing batch. Conversely, the actual batches refer to the ground truth or actual labels of the objects in the same batch. These labels provide precise information about the object positions and class annotations. The comparison between the actual and predicted batches of three batch sets is shown in Figure 19, 20, 21.

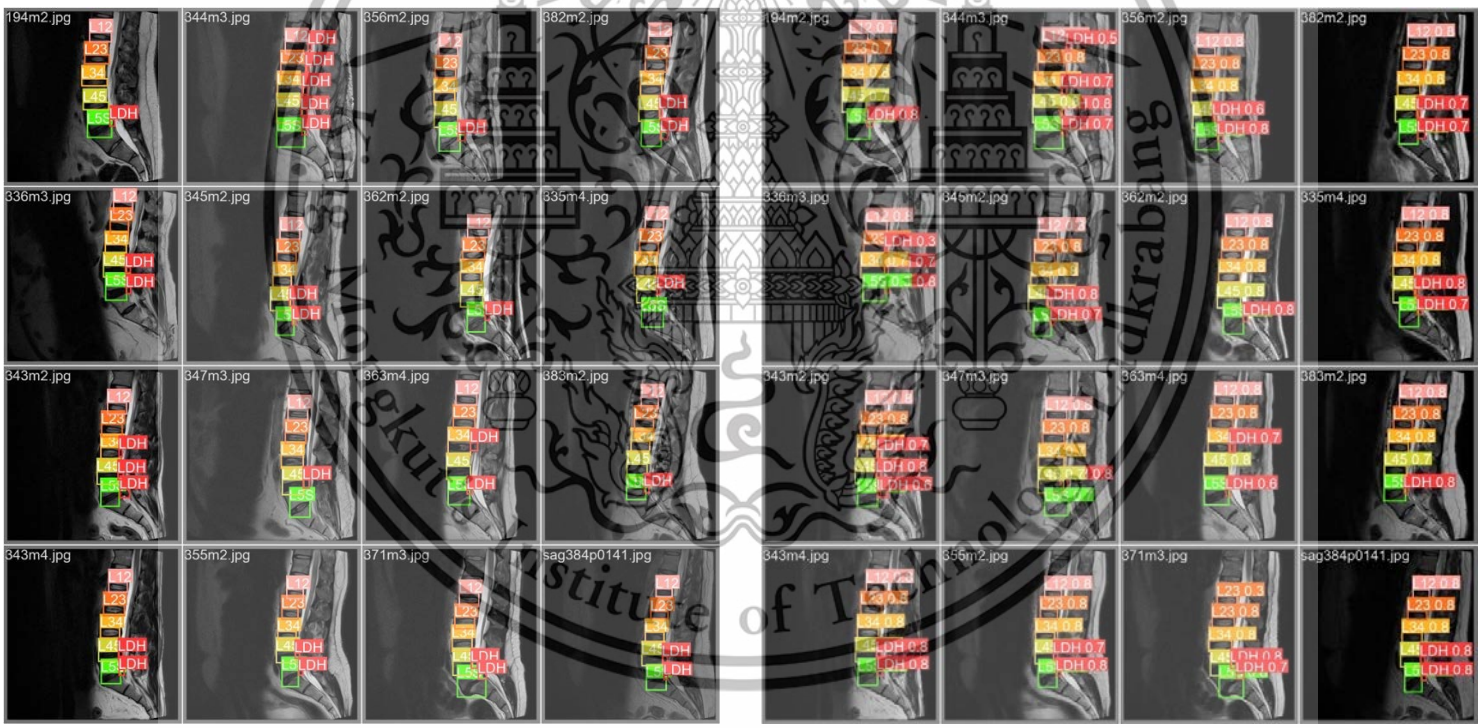
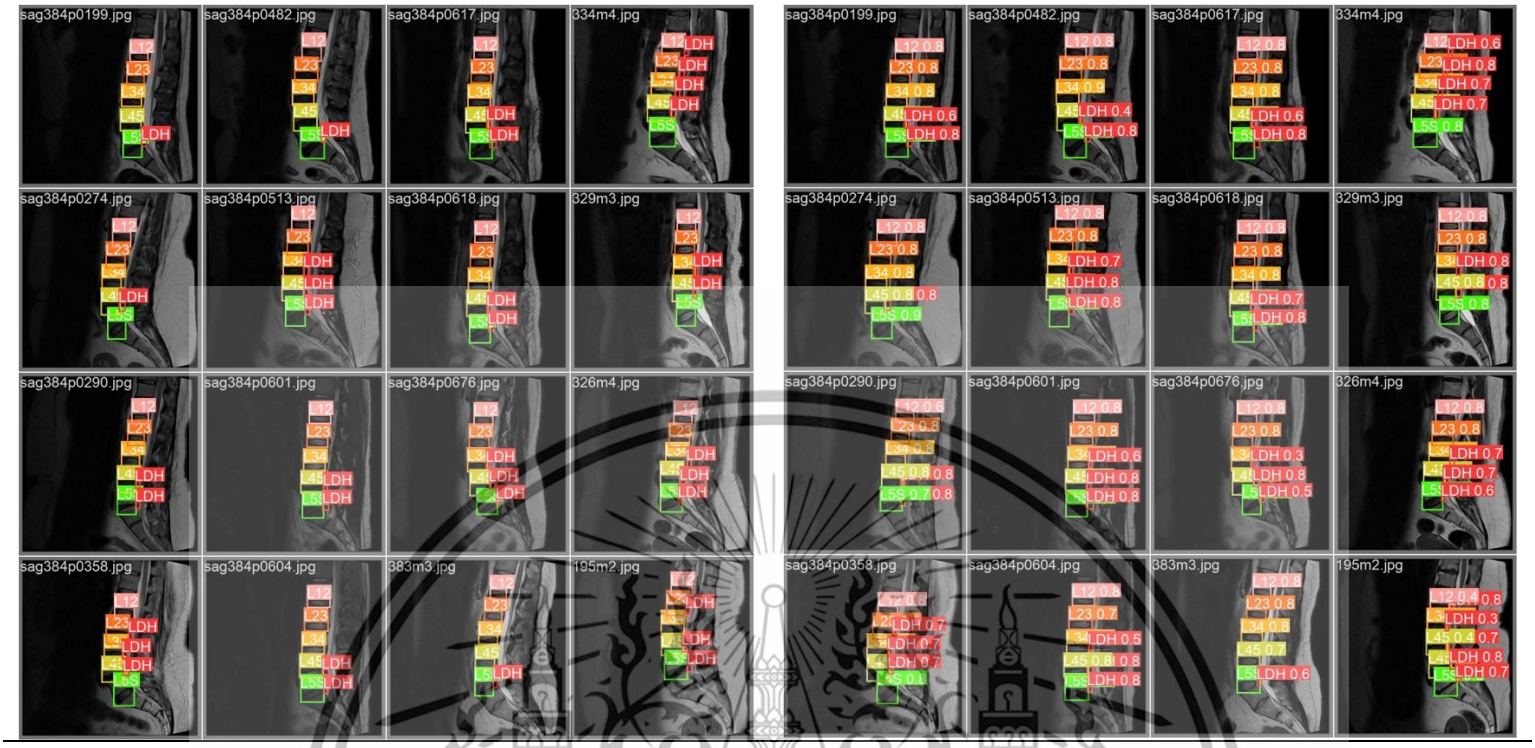


Figure 19 Testing batch 1



Actual Prediction

Figure 20 Testing batch 2

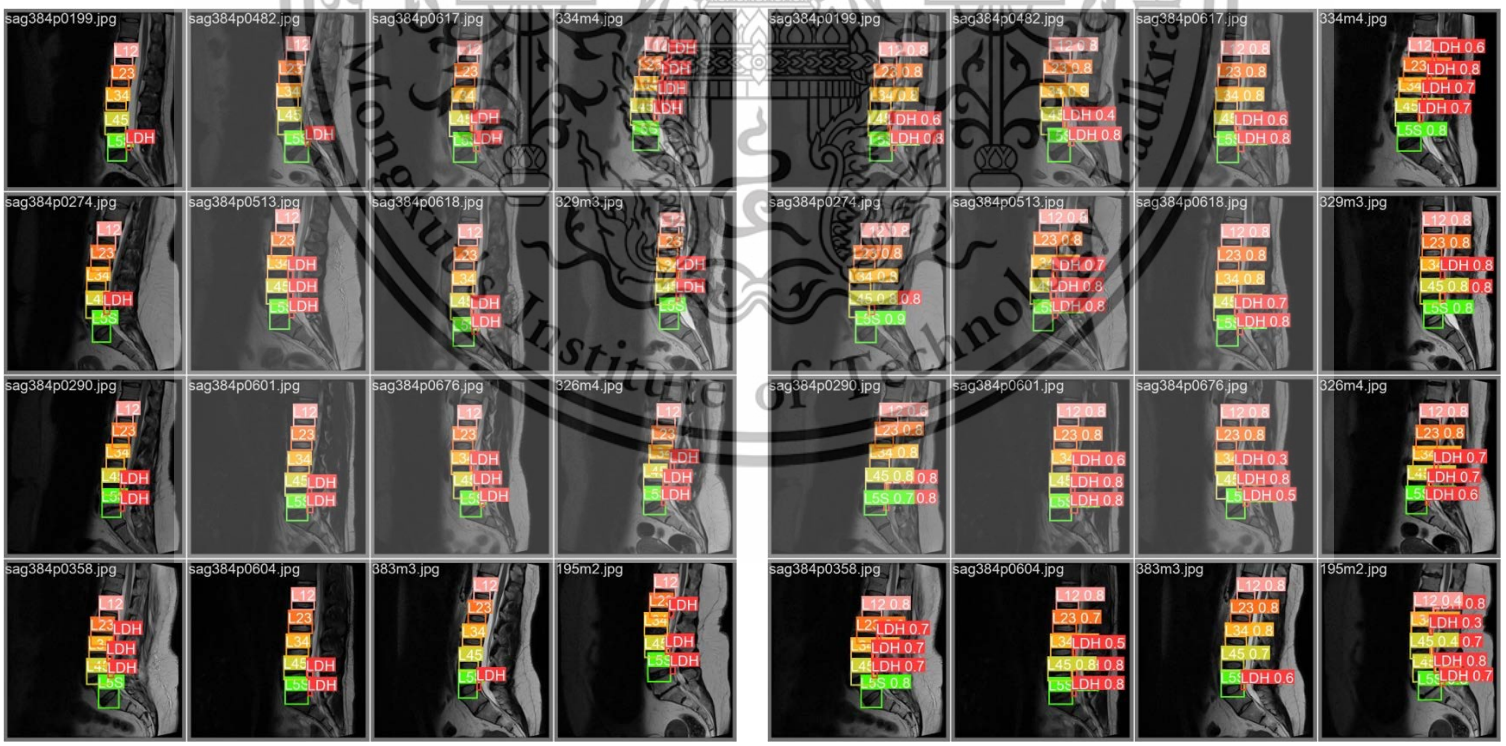


Figure 21 Testing batch 2

4.3.2 Confusion matrix

The figure 22 confirmation matrix provided by enhanced YOLOv5 model sheds light on the accuracy level of the prediction outcome when the dataset is used with six different labels related to the magnetic resonance images (MRI scans). The matrix is another visualized form where the model's prediction is compared to the real values of data. Each row of the matrix, being associated with the real label, is parallel with each column, on the other hand, representing the prediction category. The numbers in the diagonal part of the matrix tell us the true positive values. It means the model correctly concludes the corresponding label. These details are crucial for the model performance assessment and determining its ability to make precision forecasts.

The validation process run on the different frameworks was recorded via the confusion matrix method. YOLOv5 achieved pretty high accuracy averagely, hitting a value higher than 96% for object detection. In addition, LDH class reasonably reported a precise figure of 0.95, which described the exact rate of correctly detected LDH items. The model, although appears to have a primarily focus on LDH, also exhibited significant accuracy ratios for the other specified categories, and the ratios were 0.97 for L12, 0.97 for L23, 0.98 for L34, and 0.98 for L5S.

These results indicate YOLOv5 model power in providing precise detection of objects in the MRI scans. Apart from that, the feature of high precision across various categories demonstrate the model's capability and robust performance especially when tackling different object types. Matrix confusion ranks among the most important of the testing and analytical tools for determining model's forecasting and diagnostic capabilities and judging its condition in terms of accuracy.

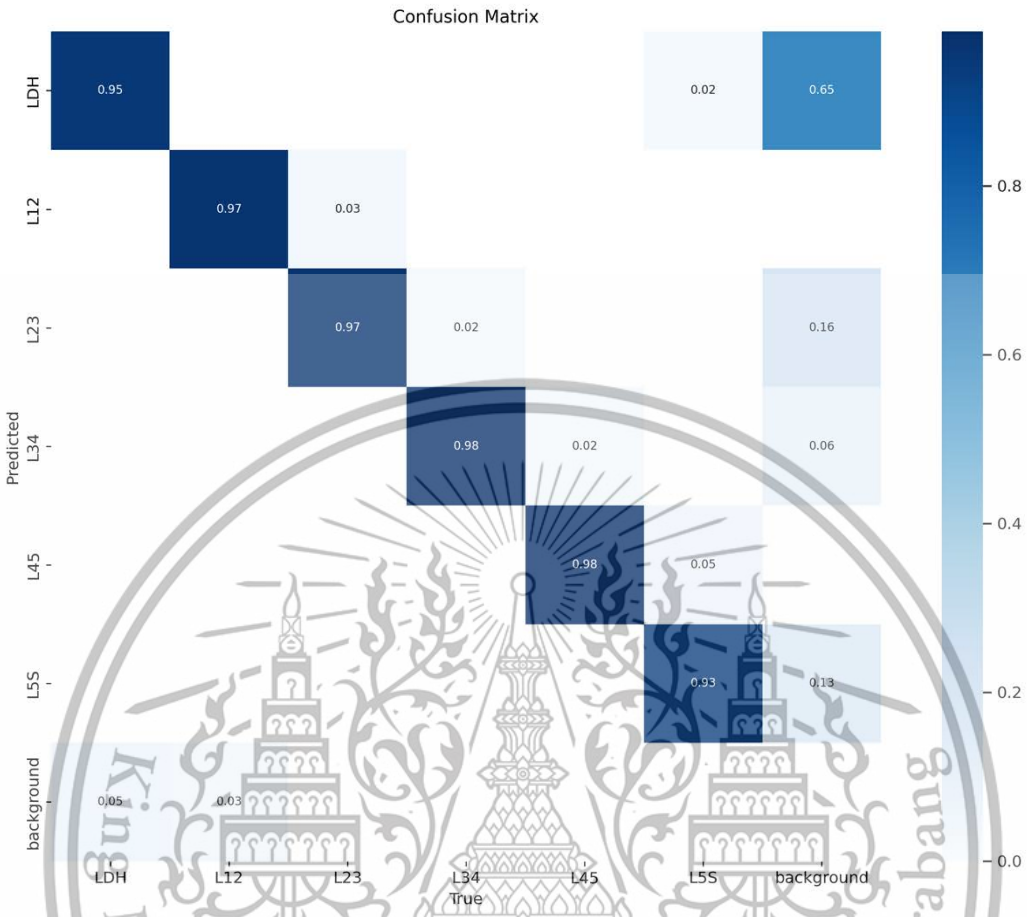


Figure 22. Confusion Matrix

4.3.3 Performance of Model over time

Throughout the training process for 100 epochs, the graphs representing the loss give the information which is useful for learning how the object detection model is doing. The box loss functional decreases uniformly at each epoch, and as the model learns, the bounding boxes become more precise in object location. The result is the model's performance is getting better and better at detecting accurate object boundaries as it can be seen from Figure 23 (A). In the same way, the figure 23 (B) that scrutinizes the model's ability to detect object slopes downwards. This suggests that the model starts to have an increased ability of telling apart the boxes with objects inside and

those with nothing. Reduced object loss denotes the model's capability to successfully differentiate and identify objects in the image.

The class loss, which is the performance indicator of figure 23 (C) for the model's object classification also gets worse and worse. Hence, this means that the model is becoming more and more competent at assigning accurate labels to detected objects, which eventually results in improved classification accuracy.

As the training goes on through 100 epochs the plot displaying all three loss values illustrates the model's continuous success in accurately localizing objects, detecting the objects presence, and making a correct classification. Such evolution is an additional factor that results in the improvement in general task performance for object detection.

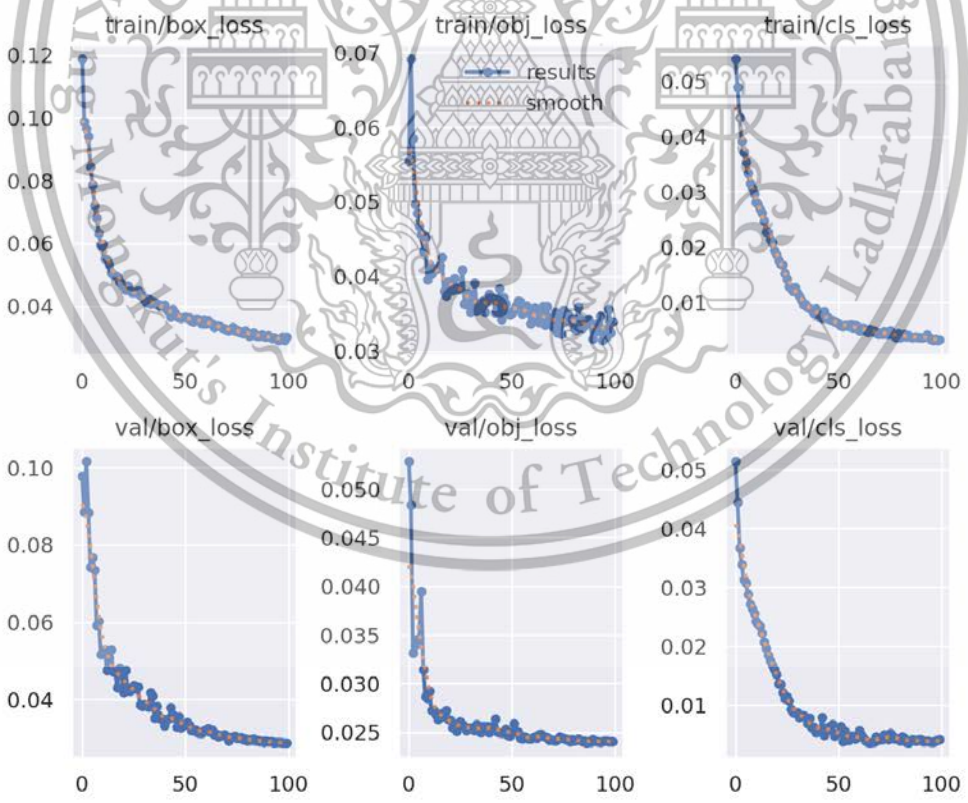


Figure 23. Performance of Training Progress : Losses

Figure 24 illustrates the object detection model's performance metrics over the training. The "metrics/precision" metric shows a steady increase from the beginning to the end of the training, indicating an improvement in the model's ability to make accurate optimistic predictions. The "metrics/recall" metric also demonstrates an upward trend, suggesting that the model becomes more effective at capturing a higher proportion of actual positive instances as training progresses.

The "mAP_0.5" metric significantly rises throughout training, improving the model's overall detection performance. This implies that the model achieves higher average precision across different IoU thresholds, particularly at the IoU threshold of 0.5.

Furthermore, the "metrics_0.5:0.95" metric, which considers a range of IoU thresholds, also demonstrates a positive trend, indicating that the model performs well across various IoU thresholds from 0.5 to 0.95. This suggests that the model maintains consistent and accurate detections across different levels of object overlap.

Overall, the graph results indicate that the model's precision, recall, and mAP metrics consistently improve over 100 epochs of training. This suggests that the model becomes more accurate, comprehensive, and effective at detecting and classifying objects as the training progresses.

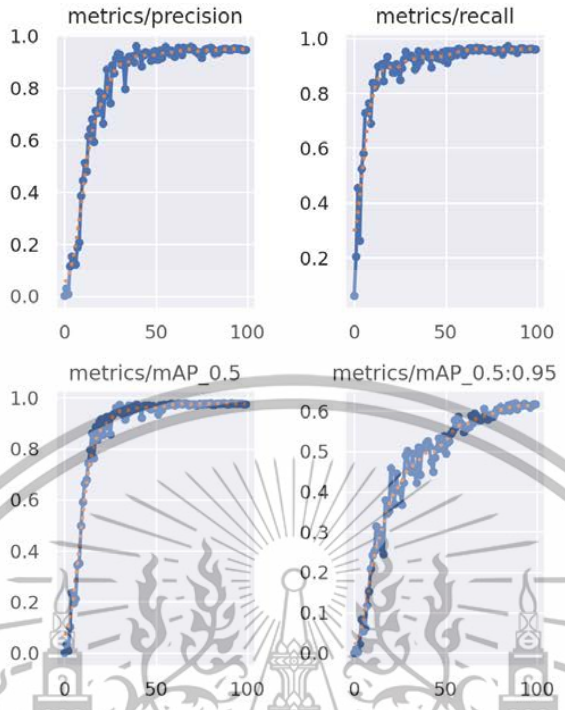


Figure 24 Performance of Training Progress: Metrics

4.3.3 Performance results of Yolov5 model

Table 1. Performance results of Yolov5 model

	Precision	Recall	F1 Score	Accuracy
LDH	0.886	0.955	0.896	0.963
L12	0.98	0.907	0.964	
L23	0.933	0.95	0.957	
L34	0.967	0.983	0.971	
L45	0.95[	0.976	0.963	
L5S	0.983	0.967	0.966	
All	0.951	0.95	0.952	

According to the table 1, the experimental results of the object detection using the YOLOv5 model demonstrate impressive performance across various evaluation metrics. The implementation can achieve a precision (P) of 0.951, a recall (R) of 0.955, and a mean average precision at an IoU 0.5 (mAP50) of 0.978.These metrics indicate the model's ability to accurately identify and classify objects in the given images. On the other hand the model displays also a high F1 score (harmonic mean of the precision and recall) which calculates the accuracy overall of the model. By reaching these measures, YOLOv5 model shows itself to be a reliable instrument for detecting objects with high preciseness and a good recall rate, and so a tool of choice for various object detection purposes.

The F1 curve, Figure 25, for all classes shows that at a threshold value of 0.517, the F1 score is 0.95 for each class. This is reflected in the high performance and accuracy standards which the model maintains for all the classes. A good score of 0.95 indicates that the model correctly makes a trade-off and that it has a high precision and high recall for each class at this threshold.

This result implies that the model has a strong ability to accurately classify instances belonging to different classes, with a high level of confidence. It shows the model performance in all the classes and gives an account of how accurately it captures needed features and predictions from every single class. The F1 score of 0.95 indicates the model's efficiency in which it can handle in almost all situations and different object detection tasks.

The Precision-Recall (PR) curve for all classes shows that the model achieves a high level of performance with a precision of 0.978 and a mean Average Precision (mAP) of 0.5. A precision of 0.978 indicates that the model makes accurate positive predictions for the given classes, minimizing false positives. This high precision score suggests that the model has a low rate of incorrectly classifying instances as positive. The PR curve and the high precision and mAP scores

indicate that the model performs well in accurately classifying instances and maintaining a balance between precision and recall for all classes shown as Figure 26.

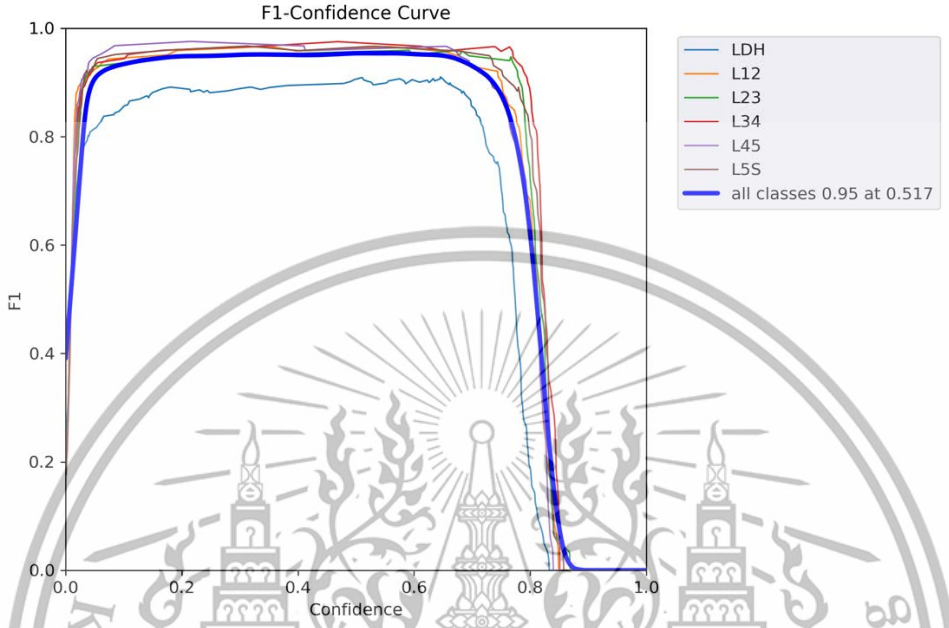


Figure 25. F1-confident curve

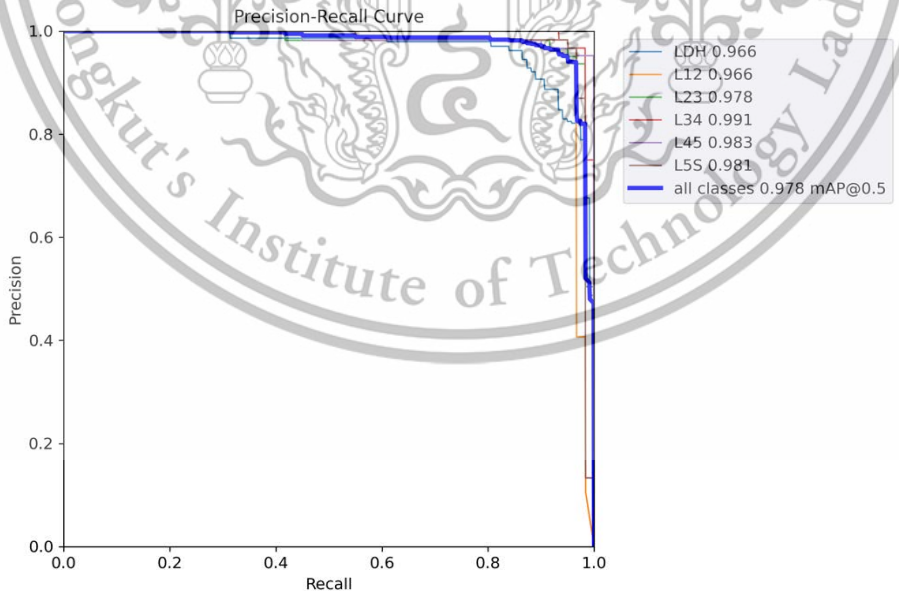


Figure 26. Precision-Recall Curve

CHAPTER V

CONCLUSION

5.1 Introduction

In this chapter, the discussion and conclusion will provide a comprehensive overview of the study, including the findings, limitations, and implications of the developed automatic lumbar disc herniation detection approach. This will contribute to the existing body of knowledge in the field and pave the way for further research and advancements in the automated diagnosis of disc herniation using deep learning techniques.

5.2 Discussion

The system presented very high scores of precision, recall, and mAP50 for various classes which made it good in object detection.Specifically, the "LDH" class, which focuses on detecting lumbar disc herniation, showed promising results with a precision of 0.886 and recall of 0.907. The model was also seen to have performed well on the other classes, such as "L12," "L23," "L34," "L45," and "L5S", precision scores of 0.959 to 0.983. The model's general performance is proven by both mAP50 and mAP50-95 scores, demonstrating that the model is capable of extensive meticulous image analysis.

Beside the evaluation metrics, the loss values during training provide valuable insights into the model's learning progress. The box loss, object loss, and class loss continuously reduced over the epochs of training which can be an indicator of successfully optimization process and a good training. Correlations of the loss values indicate the performance of the model not only in the identification of objects locations, but also sequence of differentiating between the objects and the background regions and classifying them.

The combination of high evaluation metrics and decreasing loss values suggests that the YOLOv5 model effectively learned to detect and classify objects in the given dataset. These outcomes provide us with much-needed assurance that our model not only generalizes well but also predicts accurately on unseen data.

In addition to the evaluation results, real-time detection was also performed using the trained YOLOv5 model. This involved applying the model to live video streams or recorded videos to detect objects in real-time. The model was able to process each frame of the video and provide predictions for the presence and location of objects.

The real-time detection capability of the YOLOv5 model opens up possibilities for various practical applications where timely object detection is crucial. However, it is important to consider the computational requirements and hardware limitations when deploying the model for real-time detection, as it may require sufficient processing power and resources to maintain the desired performance.

Despite the promising results, there are several limitations to consider. Firstly, the performance of the improved YOLOv5 model heavily relies on the quality and diversity of the training data. Insufficient or biased training data may affect the model's generalization ability and introduce errors in object detection.

Secondly, the model's performance may vary depending on the complexity and variability of the MRI scans. Certain challenging conditions, such as low-resolution scans, artifacts, or overlapping structures, can pose difficulties for accurate object detection.

The accurate detection of small disc herniations requires the expertise and experience of radiologists or medical professionals in interpreting the MRI scans. The subjective nature of interpretation and the potential for inter-observer variability can introduce limitations in the

consistent detection of small herniations. Detecting small disc herniations alone may not be sufficient for clinical decision-making. The significance and clinical relevance of a small herniation may depend on various factors, including the patient's symptoms, medical history, and overall clinical context. Therefore, accurate detection should be complemented by a comprehensive clinical assessment.

Addressing these limitations requires ongoing research and development in imaging techniques, advanced algorithms, and comprehensive validation studies. The collaboration between radiologists, researchers, and medical professionals is essential to overcome these limitations and improve the accuracy and reliability of small disc herniation detection in MRI scans.

5.3 Conclusion

In conclusion, the implementation of YOLO for disc herniation detection in MRI scans offers several advantages for faster and accurate diagnosis. The use of YOLO enables real-time detection, allowing for efficient and timely identification of disc herniation abnormalities. By leveraging deep learning algorithms, YOLO exhibits high precision and recall metrics, resulting in reliable and consistent detection performance.

Table 2 : Overall Performance results of YOLOv5 model.

Class	Precision	Recall	F1 Score	Accuracy	mAP@0.5	mAP@0.5:0.95
all	0.951	0.955	0.896	0.963	0.978	0.62
LDH	0.886	0.907	0.964	0.95	0.966	0.473
L12	0.98	0.95	0.957	0.97	0.966	0.656
L23	0.933	0.983	0.971	0.97	0.978	0.652
L34	0.967	0.976	0.963	0.98	0.991	0.69
L45	0.959	0.967	0.966	0.98	0.983	0.638
L5S	0.983	0.95	0.952	0.93	0.981	0.61

The results of our study demonstrate the effectiveness of the YOLOv5 model in detecting disc herniation across different label classes, such as LDH, L12, L23, L34, L45, and L5S. The model achieved impressive accuracy rates, with all classes surpassing 0.95 and an overall mAP50 score of 0.978. This indicates the model's ability to accurately localize and classify disc herniation instances within MRI scans.

The utilization of the sagittal plane in our approach further enhances the accuracy of the detection, as it provides a comprehensive view of the spine and facilitates precise localization of disc herniation. However, it is important to acknowledge the limitation of focusing solely on the sagittal plane, as disc herniation can present differently in other planes, such as the axial plane. Incorporating multiple imaging planes, when available, can enhance the overall accuracy and diagnostic capabilities.

While our study focuses on the application of YOLOv5 for disc herniation detection, it is essential to consider the limitations of the model and the dataset used. The small size of some disc herniation instances may pose challenges in detection, particularly in cases where the abnormalities are subtle or not well-defined. Additionally, the performance of the model may be influenced by factors such as image quality, variations in patient positioning, and the presence of artifacts.

Despite these limitations, the integration of YOLOv5 as a tool for disc herniation detection holds great promise in the field of medical imaging. The ability to rapidly and accurately identify disc herniation abnormalities can significantly aid in faster diagnosis, treatment planning, and improved patient outcomes. Further research and development in this area, along with the incorporation of multi-plane analysis, can further enhance the performance and applicability of YOLO-based models for disc herniation detection in clinical practice

REFERENCES

1. Chaiwat, C., Thitivaraporn, S., Pintavirooj, C., & Kaewduangdee, S. (2013). Lumbar Disc Herniation: Symptoms, Diagnosis, and Treatment. Journal of Orthopaedic Surgery, 21(2), 223–233. Available at: https://www.rama.mahidol.ac.th/ortho/sites/default/files/public/file/pdf/low_back_chaiwat55.pdf
2. Urits, I., Burshtein, A., Sharma, M., Testa, L., Gold, P. A., Orhurhu, V., Viswanath, O., Jones, M. R., Sidransky, M. A., Spektor, B., & Kaye, A. D. (2019). Low Back Pain, a Comprehensive Review: Pathophysiology, Diagnosis, and Treatment. Current Pain and Headache Reports, 23(3). doi: 10.1007/s11916-019-0757-1
3. Manchikanti L, Singh V, Datta S, Cohen SP, Hirsch JA. Comprehensive review of epidemiology, scope, and impact of spinal pain. Pain Physician. 2009;12(4):E35-E70.
4. Winegar, B., Kay, M., & Taljanovic, M. S. (2020). Magnetic resonance imaging of the spine. Polish Journal of Radiology, 85(1), 550-574. doi: 10.5114/pjr.2020.99887
5. Litjens, G., Kooi, T., Bejnordi, B. E., Setio, A. A. A., Ciompi, F., Ghafoorian, M., ... & Sánchez, C. I. (2017). A survey on deep learning in medical image analysis. Medical image analysis, 42, 60-88.
6. Vingle [Username: cityxray]. Available at: https://www.vingle.net/cityxray?cv=1
7. Healthcare Extreme. (n.d.). How to Read MRI Lumbar Spine in 8 Easy Steps. Retrieved from https://healthcareextreme.com/how-to-read-mri-lumbar-spine-in-8-easy-steps/
8. Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep Learning (Vol. 1). MIT Press.

9. Arifando, R., Eto, S., & Wada, C. (2023). Improved YOLOv5-Based Lightweight Object Detection Algorithm for People with Visual Impairment to Detect Buses. Applied Sciences, 13(9), 5802. doi: 10.3390/app13095802
10. Mohamed, A. (2021). Getting Started with YOLOv5 in a Few Minutes. Medium. Retrieved from https://medium.com/mlearning-ai/getting-started-with-yolov5-in-a-few-minutes-80ea2ff25c3c
11. Sudirman, S., Al Kafri, A., Natalia, F., Meidia, H., Afriliana, N., Al-Rashdan, W., Bashtawi, M., & Al-Jumaily, M. (2019). Lumbar Spine MRI Dataset. Mendeley Data, V2. doi: 10.17632/k57fr854j2.2
12. VISo.ai. (n.d.). LabelImg for Image Annotation. Retrieved from https://viso.ai/computer-vision/labelimg-for-image-annotation/
13. Section. (n.d.). Understanding Pascal VOC Dataset. Retrieved from https://www.section.io/engineering-education/understanding-pascal-voc-dataset/
14. Mohamed, A. (2021). The Practical Guide for Object Detection with YOLOv5 Algorithm. Towards Data Science. Retrieved from https://towardsdatascience.com/the-practical-guide-for-object-detection-with-yolov5-algorithm-74c04aac4843
15. Mahmoud, A. (2021). Getting Started with YOLOv5 in a Few Minutes. Medium. Retrieved from https://medium.com/mlearning-ai/getting-started-with-yolov5-in-a-few-minutes-80ea2ff25c3c

APPENDIX A

CODE EXAMPLE OF XML FILE IN LABELLING IMAGE

```
<annotation>
	<folder>selected</folder>
	<filename>194m1.jpg</filename>
	<path>/Users/jajalyngirly/Documents/thesis/yolo_object_detection/1_datapreparation/selected/194m1.jpg</path>
	<source>
		<database>Unknown</database>
	</source>
	<size>
		<width>384</width>
		<height>384</height>
		<depth>1</depth>
	</size>
	<segmented>0</segmented>
	<object>
		<name>L5S</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>181</xmin>
			<ymin>248</ymin>
			<xmax>232</xmax>
			<ymax>290</ymax>
		</bndbox>
	</object>
	<object>
		<name>LDH</name>
		<pose>Unspecified</pose>
		<truncated>0</truncated>
		<difficult>0</difficult>
		<bndbox>
			<xmin>226</xmin>
```

<ymin>244</ymin>
<xmax>241</xmax>
<ymax>260</ymax>
</bndbox>
</object>
<object>
<name>L45</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>169</xmin>
<ymin>200</ymin>
<xmax>228</xmax>
<ymax>227</ymax>
</bndbox>
</object>
<object>
<name>L34</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>170</xmin>
<ymin>148</ymin>
<xmax>227</xmax>
<ymax>170</ymax>
</bndbox>
</object>
<object>
<name>L23</name>
<pose>Unspecified</pose>
<truncated>0</truncated>
<difficult>0</difficult>
<bndbox>
<xmin>178</xmin>

```
                <ymin>96</ymin>
                <xmax>230</xmax>
                <ymax>117</ymax>
            </bndbox>
        </object>
        <object>
            <name>L23</name>
            <pose>Unspecified</pose>
            <truncated>0</truncated>
            <difficult>0</difficult>
            <bndbox>
                <xmin>188</xmin>
                <ymin>47</ymin>
                <xmax>235</xmax>
                <ymax>69</ymax>
            </bndbox>
        </object>
</annotation>
```

APPENDIX B

CODE FOR READ AND EXTRACT DATA IN PYTHON

```
import os
from glob import glob
import pandas as pd
from functools import reduce
from xml.etree import ElementTree as et

# Load all xml files and store in a list
xml_list = glob('./selected/*.xml')

# step-2 : reas xml files
# from each xml file we need to extract
# filename, size(width, height), object(name, xmin, xmax, ymin, ymax)
def extract_text(filename):
    tree = et.parse(filename)
    root = tree.getroot()

    # extract filename
    image_name = root.find('filename').text

    # width and height of the image
    width = root.find('size').find('width').text
    height = root.find('size').find('height').text
    objs = root.findall('object')
    parser = []
    for obj in objs :
        name = obj.find('name').text
        bndbox = obj.find('bndbox')
        xmin = bndbox.find('xmin').text
        xmax = bndbox.find ('xmax').text
        ymin = bndbox.find ('ymin').text
        ymax = bndbox.find ('ymax').text
        parser.append([image_name], width, height, name, xmin, xmax ,ymin ,ymax)
    return parser
```

```
import xml.etree.ElementTree as et

def extract_text(filename):
    tree = et.parse(filename)
    root = tree.getroot()

    # extract filename
    image_name = root.find('filename').text

    # width and height of the image
    width = root.find('size').find('width').text
    height = root.find('size').find('height').text

    objs = root.findall('object')
    parser = []
    for obj in objs:
        name = obj.find('name').text
        bndbox = obj.find('bndbox')
        xmin = bndbox.find('xmin').text
        xmax = bndbox.find('xmax').text
        ymin = bndbox.find('ymin').text
        ymax = bndbox.find('ymax').text
        parser.append([image_name, width, height, name, xmin, xmax, ymin, ymax])

    return parser

# Assuming you have a list of XML file names called xml_list
parser_all = list(map(extract_text, xml_list))

data = reduce(lambda x,y : x+y, parser_all)

df = pd.DataFrame (data, columns = ['filename','width', 'height', 'name', 'xmin', 'xmax', 'ymin',
'ymax'])
df.head()
df['name'].value_counts()
df.info()
```

```
#type conversion
cols = ['width', 'height', 'xmin', 'xmax', 'ymin', 'ymax']
df[cols] = df[cols].astype(int)

# Check the data types of the DataFrame
df.info()

# apply center x, center y formula
df['center_x'] = ((df['xmax'] + df['xmin']) / 2) / df['width']
df['center_y'] = ((df['ymax'] + df['ymin']) / 2) / df['height']
# w
df['w'] = (df['xmax'] - df['xmin']) / df['width']
# h
df['h'] = (df['ymax'] - df['ymin']) / df['height']
df.head ()

# split data into train and test
images = df['filename'].unique()

# 80% train and 20% test
img_df = pd.DataFrame(images,columns = ['filename'])
img_train = tuple (img_df.sample(frac= 0.8)['filename']) # shuffle and pick 80% of images

# Assign id number to object names
# Label encoding
def label_encoding(x):
    labels = {'LDH':0, 'L12':1, 'L23':2, 'L34': 3, 'L45' : 4, 'L5S': 5}
    return labels[x]

train_df.loc[:, 'id'] = train_df['name'].apply(label_encoding)
test_df.loc[:, 'id'] = test_df['name'].apply(label_encoding)

# save image and labels in text file
import os
```

```
from shutil import move

train_folder = 'selected/train'
test_folder = 'selected/test'

os.makedirs(train_folder, exist_ok=True)
os.makedirs(test_folder, exist_ok=True)

cols = ['filename','id','center_x','center_y','w','h']
groupby_obj_train = train_df[cols].groupby('filename')
groupby_obj_test = test_df[cols].groupby('filename')

# groupby_obj_train.get_group('314m2.jpg').set_index('filename').to_csv('sample.txt',
index=False, header=False)
# save each image in train/test folder and repective labels in .txt

def save_data(filename, folder_path, group_obj):
    # move image
    src = os.path.join('selected',filename)
    dst = os.path.join(folder_path,filename)
    move(src,dst) # move image to destination folder

# save the labels
    text_filename = os.path.join (folder_path,
                        os.path.splitext(filename)[0]+'.txt')
    group_obj.get_group(filename).set_index('filename').to_csv(text_filename,sep
',index=False,header=False)
filename_series.apply(save_data,args=(train_folder,groupby_obj_train))
filename_series_test = pd.Series(groupby_obj_test.groups.keys())
filename_series_test.apply(save_data,args=(test_folder,groupby_obj_test))
```

APPENDIX C

TRAINING CODE IN GOOGLE COLAB

```
!python train.py --data data.yaml --weights yolov5s.pt --img 640 --batch-size 8 --name Model --epochs 100
```

APPENDIX D

CODE OF PREDICTION FROM THE YOLO MODEL

```
import cv2
import numpy as np
import os
import yaml
from yaml.loader import SafeLoader

# load YAML
with open('data.yaml',mode='r') as f:
    data_yaml = yaml.load(f,Loader=SafeLoader)

labels = data_yaml['names']
print(labels)

# Load YOLO model
yolo = cv2.dnn.readNetFromONNX('./Model6/weights/best.onnx')
yolo.setPreferableBackend(cv2.dnn.DNN_BACKEND_OPENCV)
yolo.setPreferableTarget(cv2.dnn.DNN_TARGET_CPU)

# Load the image
img = cv2.imread('./sag384p1247.jpg')
image = img.copy()
row, col, d = image.shape

#  get the YOLO predictiom from the image
# step-1 convert image into square image (array)
max_rc = max(row,col)
input_image = np.zeros((max_rc,max_rc,3),dtype=np.uint8)
input_image[0:row,0:col]=image

# step 2 get prediction from squre array
INPUT_WH_YOLO = 640
```

```
blob =
cv2.dnn.blobFromImage(input_image,1/255,(INPUT_WH_YOLO,INPUT_WH_YOLO),swapR
B=True,crop=False)
yolo.setInput(blob)
preds = yolo.forward() #detection or prediction from YOLO

print(preds.shape)



# Non Maximum Supression
#step 1 filter detection base on confidence(0.4) and probability score (0.25)
detections = preds[0]
boxes = []
confidences = []
classes = []

#width and height of the image (input_image)
image_w, image_h = input_image.shape[:2]
x_factor = image_w/INPUT_WH_YOLO
y_factor = image_h/INPUT_WH_YOLO

for i in range(len(detections)):
    row = detections[i]
    confidence = row[4] # confidence of detection and objection
    if confidence > 0.4:
        class_score = row[5:].max() #maximum probability from 6 objects
        class_id = row[5:].argmax() #get the index position at which max probablity occur

        if class_score > 0.25:
            cx,cy,w,h = row[0:4]
            # construct bounding from four values
            # left, top,width and height
            left = int((cx -0.5*w)*x_factor)
            top = int((cy -0.5*h)*y_factor)
            width = int(w*x_factor)
            height = int(h*y_factor)
```

```
        box = np.array([left,top,width,height])

        #append value into list
        confidences.append(confidence)
        boxes.append(box)
        classes.append(class_id)
# clean
boxes_np = np.array(boxes).tolist()
confidences_np =np.array(confidences).tolist()

#NMS
index = cv2.dnn.NMSBoxes(boxes_np,confidences_np,0.25,0.45).flatten()
index

# Draw the boundinf box
for ind in index:
    print(ind)
    # extract bounding box
    x,y,w,h = boxes_np[ind]
    bb_conf = int(confidences_np[ind]*100)
    classes_id = classes[ind]
    class_name = labels[classes_id]

    text = f'{class_name}: {bb_conf}%'
    print(text)

    cv2.rectangle(image,(x,y),(x+w,y+h),(0,255,0),1)
    cv2.rectangle(image,(x,y-10),(x+w,y),(255,255,255),-1)

    cv2.putText(image,text,(x,y-3),cv2.FONT_HERSHEY_PLAIN,0.7,(255,0,255,),1)

cv2.imshow('original',img)
cv2.imshow("yolo_prediction",image)
cv2.waitKey(0)
cv2.destroyAllWindows
```

APPENDIX E

CODE OF REAL-TIME DETECTION

```
import cv2
from yolo_predictions import YOLO_Pred

yolo = YOLO_Pred('./Model6/weights/best.onnx','data.yaml')

## real time object detection

cap = cv2.VideoCapture('Screen Recording 2566-06-23 at 07.16.47.mov')

while True:
    ret, frame = cap.read()
    if ret == False:
        print('unable to read video')
        break

    pred_image = yolo.predictions(frame)

    cv2.imshow('YOLO11',pred_image)
    if cv2.waitKey(10) == 27:
        break

cv2.destroyAllwindows()
cap.release()
```