Sorbonne University

Master M2 Statistics and Learning

Application of deep learning algorithms for endometriosis detection on ultrasound videos

Author:
Olena Verbytska

Supervisor:
Baptiste Gregorutti

October 15, 2024

Contents

Acknowledgments

I would like to start by warmly thanking Claire Boyer, co-director of the M2 Statistics master's program, for her psychological support throughout this academic year. She helped me recognize my depression and connected me with specialists who helped me get back on my feet. Thanks to our joint efforts, I am able to complete my studies without repeating the year and perform my best during the internship.
I am sincerely grateful to Baptiste Gregorutti, my mentor, for accepting me into this internship, guiding me, and offering his support. I also extend my thanks to Dr. Martine Valiere for her project idea and dedication.
I would like to thank Gerard Biau and Xavier Fresquet for their encouragement and motivation.
Special thanks to Emilio, a collaborator on this project, for bringing a lively atmosphere to the office with his amazing music, inspiration, humor, engaging discussions, and serious attitude toward the project.
Thanks to Alicia and Clotilde for their responsiveness, support of my ideas, generosity, and the enjoyable time we spent together, whether working or playing games.
Finally, I want to express my gratitude to Etienne, Adrien, Nicolas, Nora, Rakhee, and Julien for sharing their professional experience, their wonderful sense of humor, and their genuine kindness.

Chapter 1

Introduction

What is endometriosis and how it may be diagnosed

Endometriosis is a condition where cells similar to those in the lining of the uterus (endometrium) grow outside the uterus. It affects an estimated 5%-10% of women and adolescent girls of reproductive age (15-49 years) and up to 50% of infertile women [5]. This is a chronic disease associated with severe, life-impacting pain during periods, sexual intercourse, bowel movements and/or urination, chronic pelvic pain, abdominal bloating, nausea, fatigue, and sometimes depression, anxiety, and infertility.
Endometriosis is not easily diagnosed and can present symptoms common to other conditions. Proper diagnosis requires a multidisciplinary team involving gynecologists, midwives, and radiologists. Delays from symptom onset to diagnosis can range from 4 to 11 years [1]. A definitive diagnosis is made through laparoscopic inspection followed by histologic confirmation after a biopsy, which is both highly invasive and expensive.
MRI (magnetic resonance imaging) is another widely used imaging technique, providing a series of consecutive slices that sometimes offer better contrast, but most patients use an ultrasound to detect endometriosis [3]. In particular, while MRI may have limitations related to cost and accessibility, medical ultrasonography is a risk-free, low-cost, ultrasound-based procedure. Its main drawback, however, is the high dependency on the expertise of the clinical staff for diagnostic effectiveness. Ultrasonography is used as a supplementary examination in suspected cases, to monitor disease progression and the response to medical treatments, and to ensure adequate preparation for laparoscopy.

How artificial intelligence may help in diagnosis of endometriosis

Artificial intelligence (AI) has emerged as a powerful tool in the diagnosis and management of endometriosis. With its ability to analyze large datasets and detect patterns that may not be visible to the human eye, AI can improve diagnostic accuracy, reduce delays, and potentially alleviate the need for highly invasive procedures such as laparoscopy.
Machine learning algorithms can be trained on medical imaging data, such as ultrasound and MRI scans, to identify the characteristic features of endometriosis. These models are particularly effective in analyzing subtle abnormalities in tissue that might be overlooked during manual inspection, leading to earlier and more accurate detection. In cases where ultrasonography is heavily dependent on the clinician's experience, AI models can help standardize the interpretation, thus reducing variability between practitioners and improving diagnostic confidence.
One area where AI excels is in the automated interpretation of ultrasound images. By training AI systems with vast datasets of labeled ultrasound images, researchers can develop models that are able to classify and segment areas of interest, such as endometriotic lesions, with high precision. This can reduce the time it takes to diagnose endometriosis, making the process more efficient and accessible, especially in areas with limited access to highly skilled radiologists.
AI is also being applied to patient data beyond imaging. By analyzing clinical records, symptom patterns, and patient histories, AI can help identify high-risk patients earlier in the diagnostic process. Predictive algorithms are being developed to assess the likelihood of endometriosis based on a combination of factors, such as the severity and frequency of symptoms, personal medical history, and genetic predispositions.
The real challenge is that AI models require large, diverse datasets to improve accuracy, and rigorous clinical validation is necessary to ensure their effectiveness in real-world settings. Nonetheless, as AI continues to evolve, it holds the promise of revolutionizing the way endometriosis is diagnosed and managed, potentially reducing the time from symptom onset to diagnosis and improving patient outcomes.

ECHOPICT project

The project was initiated following discussions between Dr. Valiere and the Sorbonne Center for Artificial Intelligence (SCAI).
Dr. Valiere is an independent specialist in medical imaging, with a focus on gynecology and fertility explorations. She has a particular interest in the detection of endometriosis, a condition often linked to infertility [5]. Dr. Valiere's goal is to raise awareness of endometriosis and its diagnosis among radiologists, midwives, general practitioners, and women. As such, the development of a tool dedicated to aiding in the medical diagnosis of endometriosis lesions is both timely and essential.
The SCAI is an institute dedicated to research and education in artificial intelligence. It distinguishes itself by fostering interdisciplinary collaboration among researchers, students, and industry professionals. The SCAI undertakes projects that integrate artificial intelligence across various fields such as health, the environment, technology, and education.
ECHOPICT aims to develop an AI-powered computer tool to assist in diagnosing endometriosis. The final product will be an application that provides real-time predictions for different types of endometriosis during ultrasound scans.
The two proposed internships were a first step in this project. I worked in collaboration with Emilio Picard, who was responsible for the development of a deep neural network for classification of rectal nodules of endometriosis. He trained different models from vision transformers to CNNs, used explainable machine learning tools to control the models' performances. The aim of my internship is to implement and evaluate an object detection deep neural network, which will be able to detect rectal nodules effectively.

Report structure

The following three chapters correspond to three main topics in my internship: theory, application, and feasibility analysis of ECHOPICT project. In Chapter 2, we first give rigorous definition of a convolutional deep neural network, describing its layers in detail. Then we make an overview of the applications of convolutional neural networks for classification and object detection tasks. The concept of transfer learning and fine-tuning is then introduced, emphasizing how pre-trained models can be adapted to specific tasks with minimal adjustments, enhancing performance while minimizing computational costs.
This is followed by an examination of region-based convolutional neural networks, where we explore how these models identify and localize objects within images. The discussion covers important architectures, including Fast R-CNN and Faster R-CNN, each offering improvements in detection speed and accuracy.
The next section of the report focuses on the YOLO (You Only Look Once) family of models. Starting with the original YOLOv1, it tracks the evolution of the model through subsequent versions, each time improving efficiency and performance in real-time object detection. Specific attention is given to YOLOv5 and YOLOv8, highlighting their advanced features and limitations.
The report then transitions to the application of deep neural networks in the context of endometriosis detection (Chapter 3). Firstly, we describe the dataset used, detailing the data collection, annotation, and preprocessing methods applied to optimize the training process. Then we discuss how model performance was evaluated, ensuring the robustness of the results. The training outcomes for both YOLO and Faster R-CNN models are analyzed.
The next key highlight of the report is the development of the ECHOPICT tool, an AI-driven solution designed for endometriosis detection. The report outlines the tool's development process, the integration of AI technologies, and the results of a feasibility study that assessed its effectiveness. The tool is presented as a promising advancement for medical personnel, offering improved diagnostic capabilities and the potential to streamline the detection of endometriosis in clinical practice.
Finally, the report concludes with a reflection on the findings, discussing the successes and challenges encountered in application. It offers recommendations for future research and emphasizes the potential of AI technologies to transform endometriosis diagnosis, ultimately improving patient outcomes.

Chapter 2

Deep neural networks for object detection

Deep neural networks are essential in modern AI because they mimic the way the human brain learns and processes information. They can identify patterns and make decisions without needing explicit programming, making them powerful tools for tasks like image recognition, speech processing, and decision-making. Deep neural networks are adaptive, improving their performance over time through learning. Their ability to handle large, unstructured datasets allows them to solve complex problems across various industries.
When applying deep learning to visual data, it is not only important to classify objects but also to detect their precise locations. This challenge is known as the object detection problem, which can be effectively addressed using deep convolutional neural networks (CNNs). In medical imaging, object detection is crucial for localizing lesions, as it can significantly expedite the initiation of treatment, making the process faster and more efficient. Object detection algorithms can be broadly categorized into two types: traditional object detection algorithms based on image processing and those based on convolutional neural networks. CNN-based methods are highly efficient compared to traditional machine learning techniques for common computer vision tasks such as image classification, object detection, and image segmentation.

1 Definitions

1.1 Deep neural network

Neural networks are functions $\mathbf{y} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ with learnable parameters $\boldsymbol{\phi}$ that map multivariate inputs $\mathbf{x} \in \mathbb{R}^{D_i}$ to multivariate outputs $\mathbf{y} \in \mathbb{R}^{D_o}$ using hidden layers with non-linear activation functions $\{a_j[\bullet]\}_{j=1}^{D}$. Deep neural network is a neural network with more than one hidden layer. Let us consider an example of deep neural network using notations from [20].

Definition 1.1.1 (**Multilayer perceptron (MLP)**)**.** Let us describe the vector of hidden units at layer k as $\mathbf{h}_k$, the vector of biases (intercepts) that contribute to hidden layer $(k+1)$ as $\boldsymbol{\beta}_k$, and the weights (slopes) that are applied to the k-th layer and contribute to the $(k+1)$-th layer as $\boldsymbol{\Omega}_k$. A multilayer perceptron $\mathbf{y} = \mathbf{f}[\mathbf{x}, \boldsymbol{\phi}]$ with K layers can be written as:

$$
\begin{aligned}
\mathbf{h}_1 &= a_1[\boldsymbol{\beta}_0 + \boldsymbol{\Omega}_0 \mathbf{x}] \\
\mathbf{h}_2 &= a_2[\boldsymbol{\beta}_1 + \boldsymbol{\Omega}_1 \mathbf{h}_1] \\
\mathbf{h}_3 &= a_3[\boldsymbol{\beta}_2 + \boldsymbol{\Omega}_2 \mathbf{h}_2] \\
&\vdots \\
\mathbf{h}_K &= a_K[\boldsymbol{\beta}_{K-1} + \boldsymbol{\Omega}_{K-1} \mathbf{h}_{K-1}] \\
\mathbf{y} &= \boldsymbol{\beta}_K + \boldsymbol{\Omega}_K \mathbf{h}_K.
\end{aligned}
$$

Each hidden layer in MLP is called fully connected or dense, because each neuron h_{ij} from the hidden layer $\mathbf{h}_i$ is connected to every other neuron in adjacent layers. Functions like ReLU $(\max(0, \bullet))$, Sigmoid $(1/(1+e^{-\bullet}))$, or Tanh $(\tanh(\bullet))$ are usually used as activation functions in fully connected layers, enabling them to learn complex patterns.

The parameters $\boldsymbol{\phi}$ of this model comprise all of these bias vectors and weight matrices: $\boldsymbol{\phi} = \{\boldsymbol{\beta}_k, \boldsymbol{\Omega}_k\}_{k=0}^{K}$.
If the k-th layer has D_k hidden units, then the bias vector $\boldsymbol{\beta}_{k-1}$ will be of size D_k. The last bias vector $\boldsymbol{\beta}_K$ has the size D_o of the output. The first weight matrix $\boldsymbol{\Omega}_0$ has size $D_1 \times D_i$, where D_i is the size of the input. The last weight matrix $\boldsymbol{\Omega}_K$ is $D_o \times D_K$, and the remaining matrices $\boldsymbol{\Omega}_k$ are $D_{k+1} \times D_k$.

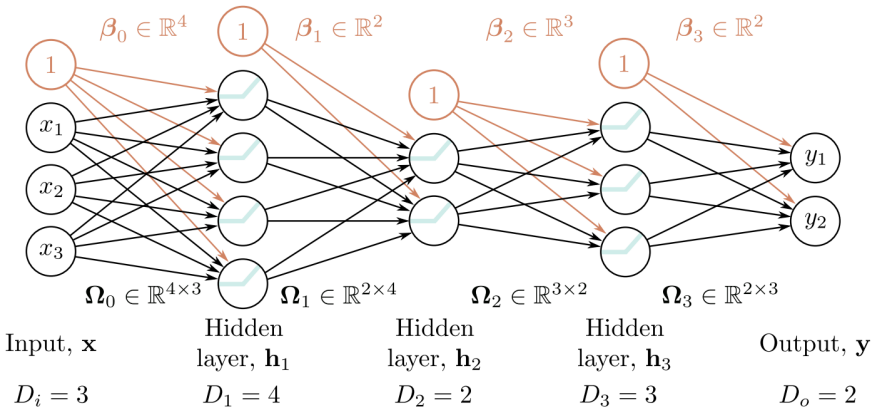


Figure 2.1: An example of deep neural network with three fully connected hidden layers with ReLU activation function [20].

The learning process in a deep neural network involves training the model to recognize patterns and make accurate predictions by adjusting its internal parameters $\boldsymbol{\phi}$. This process starts with an input layer, where data (such as images or text) is fed into the network. The data is then passed through multiple hidden layers, each consisting of neurons that perform calculations based on the input and the current weights, applying activation functions to introduce non-linearity.
As the data moves through these layers, the network makes predictions, which are compared to the actual outcomes using a loss function to measure the error. A technique called backpropagation is used to calculate how much each weight contributed to the error, and then an optimization algorithm like gradient descent adjusts the weights to minimize the error. This process of forward propagation, error calculation, optimization, and weight adjustment is repeated over many iterations, gradually improving the network's performance.

1.2 Convolutional neural network

Convolutional neural network (CNN) is a deep neural network that have been designed for image processing tasks, such as image recognition, classification, and segmentation. While MLPs consist of fully connected layers CNNs use convolutional layers that apply filters to detect edges, textures, and patterns. CNNs preserve spatial relationships in the input data (like the position of pixels in an image), while MLPs treat all input features as independent, losing any inherent spatial structure.
CNNs can be composed of these types of layers: convolutional (for feature extraction), pooling, and fully connected (for classification). Each layer is 3-dimensional: it has its height, width, and depth, which is the number of channels in an image or the number of feature maps in a hidden layer.

Convolutional layer

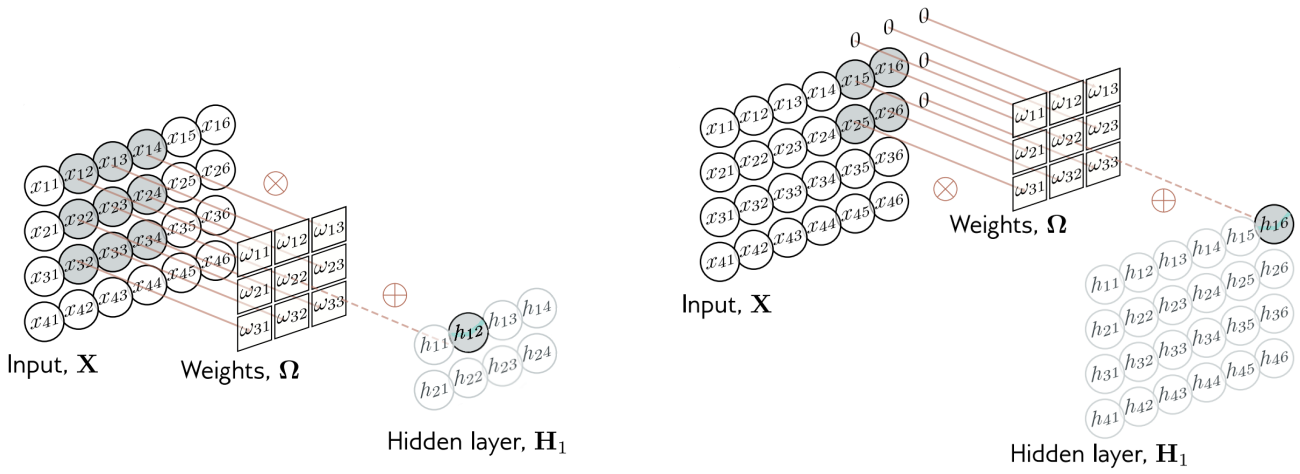
The convolutional layer is the fundamental building block of a convolutional neural network, responsible for the majority of calculations within the network. It was first introduced by Fukushima and Miyake in 1982 for pattern recognition in images [8], and then the idea was formalized over the following years by LeCun and Bengio [15].
Let us present the definition of convolutional layer adapted from [20].

Definition 1.2.1 (Convolutional layer). A $K \times K$ kernel (or filter) $\mathbf{\Omega} = \{\omega_{mn}\}_{m=1,n=1}^{K,K}$ applied to a 2D input that consists of elements (pixels) $\{x_{kl}\}_{k=1,l=1}^{H_i,W_i}$ computes a single layer of hidden units h_{ij} as:

$$h_{ij} = a\left[\beta + \sum_{m=1}^{K}\sum_{n=1}^{K}\omega_{mn}x_{i+m-1,j+n-1}\right],$$

where often $a[\bullet] = \max(0, \bullet)$ is a rectified linear unit (ReLU) activation function. The output of kernel applied to the input layer is called a feature map or activation map. If there are D different kernels applied to the input, then the hidden layer will consist of D feature maps. The number D is called the depth of the hidden layer.

Knowing how to calculate the size of the output of a convolutional layer is crucial. When designing a CNN, understanding the output size helps in structuring the network. We should ensure that the next layers receive input of the correct dimensions, preventing errors like incompatible input sizes when transitioning between layers. For a $H_i \times W_i$ input and D kernels of size $K \times K$ we can calculate the size of resulting hidden layer by the following formula: $(H_i - K + 1) \times (W_i - K + 1) \times D$. If we add extra rows and columns around the borders (*i.e.* padding) of an input image filled with zeros, then the size of hidden layer will be $(H_i + 2p - K + 1) \times (W_i + 2p - K + 1) \times D$, where p is amount of added paddings.



(a) $p = 0$ (b) $p = 1$

Figure 2.2: Examples of convolutional layers with default stride (1, 1) without (a) and with (b) padding [20].

Another important parameter of convolutional layer is stride, which defines how much the kernel shifts across the input image after each application. Stride (s_1, s_2) means that we shift the kernel by s_1 horizontally until the end of row, then continue from the beginning, but s_2 rows lower. Finally, the size of hidden layer is

$$\left\lfloor \frac{H_i + 2p - K}{s_1} + 1 \right\rfloor \times \left\lfloor \frac{W_i + 2p - K}{s_2} + 1 \right\rfloor \times D,$$

where $\lfloor \bullet \rfloor = \max\{m \in \mathbb{Z} \mid m \leq \bullet\}$ is a floor function.

Pooling layer

The pooling layer was first seen in the LeNet article [15], and named after the publication of the AlexNet [14] paper. Pooling is commonly used between convolutional layers to downsample the input data. This helps reduce the number of learned features, thereby decreasing the risk of overfitting, when model has good performance on any subset of train set, but has difficulties with recognizing patterns, that it did not see before. On the other hand, one should be careful using too many pooling layers, since it can lead to underfitting, when a model is too simple to capture the underlying patterns in the data, resulting in poor performance on both the training and testing sets. The pooling layer divides the input feature map into a set of non-overlapping regions. Each region is then transformed into a single output value, which represents the presence of a particular feature in that region.

The most common types of pooling operations are max pooling and average pooling. In max pooling, the output value for each pooling region is simply the maximum value of the input values within that region. This allows the most salient features in each pooling region to be emphasized, while muting the less relevant ones. Max pooling helps to identify the most distinctive features of an object, such as its edges and corners, which makes it useful in CNNs for object detection tasks. In average pooling, the output value for each pooling region is the average of the input values within that region. This has the effect of preserving more information than max pooling, but may also dilute the most salient features. Average pooling is often used in CNNs for tasks such as object detection and image segmentation, where a more precise representation of the input is required.

Batch normalization

Batch normalization layers were introduced by Ioffe and Szegedy [13] in 2015. The primary purpose of batch normalization is to normalize the inputs to activation functions in the network, which helps to mitigate issues related to internal covariate shift—where the distribution of network activations changes during training.
Batch normalization is particularly useful in deep networks, where the number of layers can cause problems of vanishing gradients (the gradients of a neural network become very small as they propagate back through the layers during training, leading to minimal updates in the weights) or exploding gradients (the gradients become excessively large, causing the model weights to update drastically and leading to instability during training, making the model unable to converge). By maintaining a consistent distribution of inputs to each layer, batch normalization facilitates faster convergence and improves model performance.
In their paper, Ioffe and Szegedy proposed to normalize inputs to activation functions with their mini-batch mean and variance. A mini-batch is a small, randomly selected subset of the training data used to update the model's weights during each iteration of the training process. This approach strikes a balance between computational efficiency and stability, making training faster and less prone to getting stuck in local minima.
The mean and the variance of the mini-batch are calculated as follows:

$$\mu_B = \frac{1}{M}\sum_{i=1}^{M} x_i, \quad \sigma_B^2 = \frac{1}{M}\sum_{i=1}^{M}(x_i - \mu_B)^2,$$

where M is the size of the mini-batch, x_i represents the individual inputs within that batch.
Once the mean and variance are computed, the normalized input is obtained using the formula:

$$\hat{x_i} = \frac{x_i - \mu_B}{\sqrt{\sigma_B^2 + \varepsilon}}$$

where ε is a small constant added for numerical stability to prevent division by zero. This normalization ensures that the inputs to the activation function have zero mean and unit variance, leading to more effective learning.
In CNNs, the batch normalization transformation normalizes across the spatial dimensions (height and width) of feature maps independently for each channel. This independence allows CNNs to preserve the spatial relationships within the images while still benefiting from normalization.

Dropout

Dropout is a regularization technique designed to mitigate overfitting in neural networks, ultimately enhancing testing accuracy, sometimes at the expense of training accuracy. During the training phase, for each mini-batch in the training set, dropout layers randomly disconnect a fraction of the neurons, with a specified probability. This process prevents the network from becoming overly reliant on specific neurons, encouraging it to learn more robust features by ensuring that all neurons are treated equally during training. As a result, the network develops a more generalized representation of the data, which helps improve its performance on unseen test data. After training, dropout is typically disabled, allowing all neurons to contribute to the model's predictions, thus leveraging the full capacity of the network for inference. Overall, dropout serves as a simple yet effective method to enhance the generalization capabilities of deep learning models.

Change of depth with 1×1 kernels

Sometimes we want to change the number of channels, *i.e.* depth, between one hidden layer and the next without further spatial pooling. This is usually so we can combine the representation with another parallel computation. To accomplish this, we apply a convolution with kernel size $1 \times 1 \times D'$, where $D' \leq D$, D is depth.

Each element of the output layer is computed by taking a weighted sum of all the channels at the same position. We can repeat this multiple times with different weights to generate as many output channels as we need. The associated convolution weights have size 1×1. Hence, this is known as 1×1 convolution. Combined with a bias and activation function, it is equivalent to running the same fully connected network on the input channels at every position.

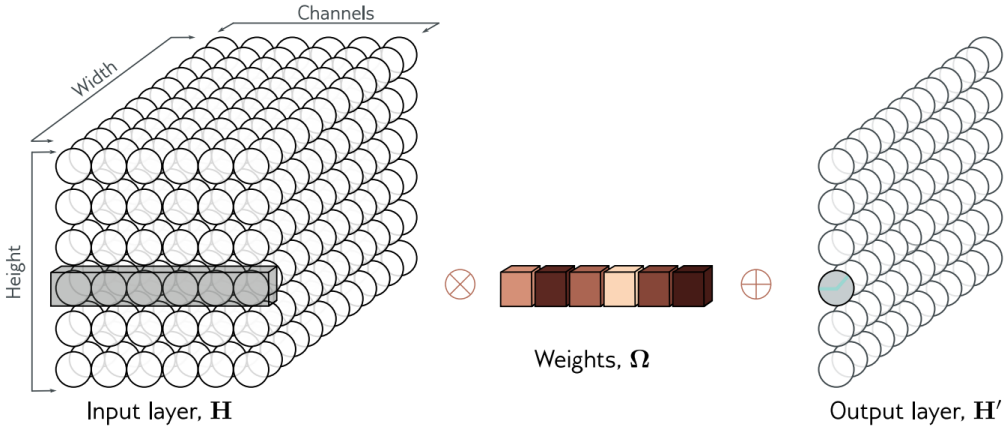


Figure 2.3: Reducing the depth of a layer with 1×1 kernel [20].

2 Application of convolutional neural networks

Image classification

Much of the early work in deep learning for computer vision centered around image classification using the ImageNet dataset, which consists of 1,281,167 training images, 50,000 validation images, and 100,000 test images, each labeled with one of 1,000 possible categories. Typically, methods resize the input images to a standard size. For example, a common input for the network is a 224×224 RGB image, and the output is a probability distribution across the 1,000 classes. The challenge lies in the large number of classes, which exhibit significant variation. In 2011, before the use of deep networks, the best method had a 25% error rate for placing the correct class in the top five predictions (a top-5 error) [12].
In 2012, AlexNet [14] (Figure 2.6) became the first convolutional neural network to perform well in this domain. AlexNet achieved a top-5 error rate of 16.4% and a top-1 error rate of 38.1%, a significant improvement over previous methods and a breakthrough that demonstrated the power of deep learning, propelling modern AI research.
The VGG network [26] (Figure 2.9) also targeted the ImageNet classification task, achieving better results with a 6.8% top-5 error rate and a 23.7% top-1 error rate. The key difference between AlexNet and VGG was the depth of the network. While AlexNet used 7 hidden layers, VGG used 19, demonstrating the importance of network depth in achieving higher performance, a trend observed for several years in deep learning research.

Object detection

In recent years, deep learning has led to significant advancements in object detection algorithms. Current popular methods for object detection can be categorized into two main types. The first category consists of region proposal-based algorithms, such as R-CNN [10], Fast R-CNN [9], and Faster R-CNN [24], which will be considered in the Section 4. These are two-stage approaches that first generate region proposals using heuristic methods like selective search [27] or CNN-based techniques, and then perform classification and regression on the proposed regions.
The second category includes one-stage algorithms like YOLO [23] (Section 5) and SSD [18], which use a CNN to directly predict object categories and positions without generating region proposals.
Two-stage algorithms first extract image features using a CNN backbone, identify possible candidate regions from the feature map, and then use sliding window operations on these regions to determine object categories and locations more precisely. In contrast, one-stage algorithms skip the region proposal step, performing feature extraction, object classification, and position regression in a single convolutional network. While their accuracy is slightly lower than that of two-stage methods, they offer significantly faster processing speeds.

3 Transfer learning and fine-tuning

Training large machine learning models from scratch often requires vast amounts of labeled data, significant computational resources, and extensive time. These challenges can be prohibitive, particularly in medical domain, where acquiring large datasets is difficult or expensive.

To address these issues, techniques such as transfer learning and fine-tuning have emerged as effective strategies for leveraging existing knowledge and enhancing model performance without the need for extensive retraining.

Transfer learning involves taking a model that has been pre-trained on a large dataset for a specific task and adapting it for a different but related task. The pre-trained model serves as a starting point, allowing to benefit from the knowledge encoded in its weights, which means that the model has already learned valuable features and representations that can be useful for the new task. For example, models like AlexNet, VGG, and ResNet, which were trained on the ImageNet dataset containing over 1,000 classes, can be used for real-world image classification or object detection. The pre-training allows them to recognize a wide array of visual features. By leveraging these models, specialists can quickly adapt them to the tasks, such as medical image classification or object detection, with minimal additional data.

The logical way to increase the model's performance is to re-train or fine-tune the weights. Fine-tuning enables the model to adapt to the nuances of the target domain without starting from scratch, which significantly reduces the time and resources required for training. During fine-tuning, the weights of chosen layers are adjusted to better align with the specific characteristics of the new data. In CNNs, the convolutional, pooling, and some of fully connected layers are usually "frozen", *i.e.* learning rate equal to 0, and the rest of fully connected layers are adapted to the task and fine-tuned using a smaller learning rate to ensure that the existing learned features are preserved while still allowing for necessary adjustments.

The challenges of training large models from scratch can be effectively mitigated through the use of pre-trained models, transfer learning, and fine-tuning. Using the knowledge gained from extensive training on large datasets, practitioners can achieve state-of-the-art performance on a variety of tasks while minimizing the resources and time required for model development.

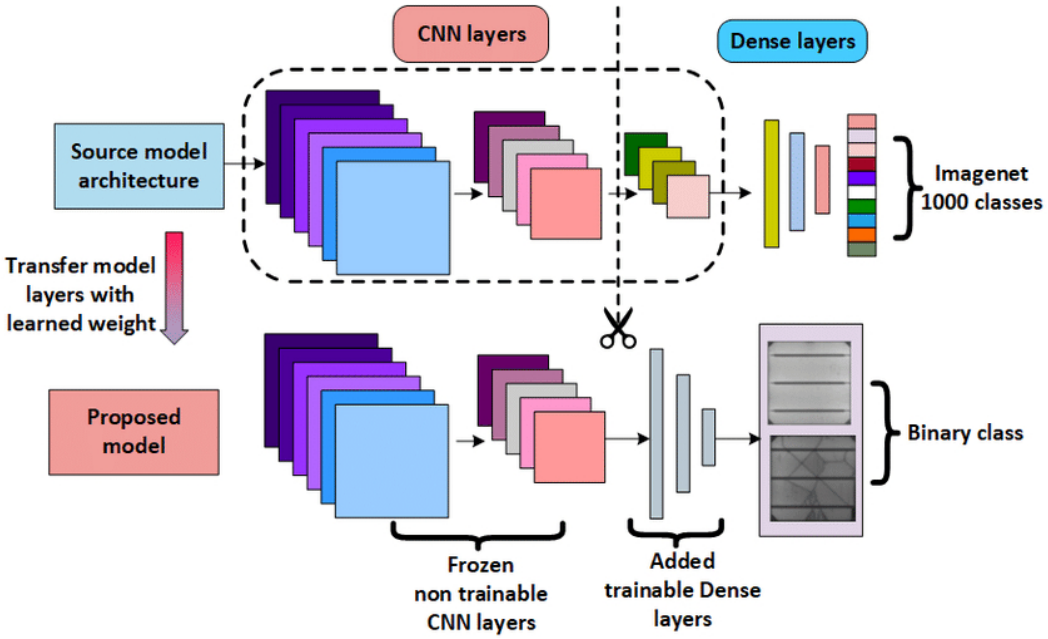


Figure 2.4: An example of transfer learning and fine-tuning [21].

4 Region-based convolutional neural networks

4.1 Concept

In 2014, Girshick et al. proposed the R-CNN (Regions with Convolutional Neural Networks) model [10]. It uses a selective search algorithm to create region proposals. Then a convolutional neural network is applied for classification and detection of objects. It can more effectively extract the features of the image than the traditional hand-made solution.

Selective search

Selective search is an algorithm that generates region proposals (Regions of Interest or RoIs) that potentially contain objects, reducing the need to exhaustively search the entire image. It operates by combining the strengths of both exhaustive search and segmentation, ensuring a balance between computational efficiency and high object recall.
The input image is first segmented into multiple regions based on color and texture similarity. For example, Felzenszwalb's efficient graph-based segmentation can be used for the task. This method considers the image as an undirected graph, where each pixel is a node and edges connect neighboring pixels. The weight of an edge represents the dissimilarity between the connected pixels. Each pixel is considered as segment and then adjacent segments are iteratively merged based on the edge weights. Merging continues until the difference between regions is smaller then some threshold. This algorithm is fast and efficient since it applies greedy approach, which is concentrated on local decisions and leads to the same segmentation that depends only on the threshold.
After the hierarchical grouping is completed, selective search generates around 1,000 to 2,000 region proposals. Each proposal is a bounding box that potentially contains an object of interest.

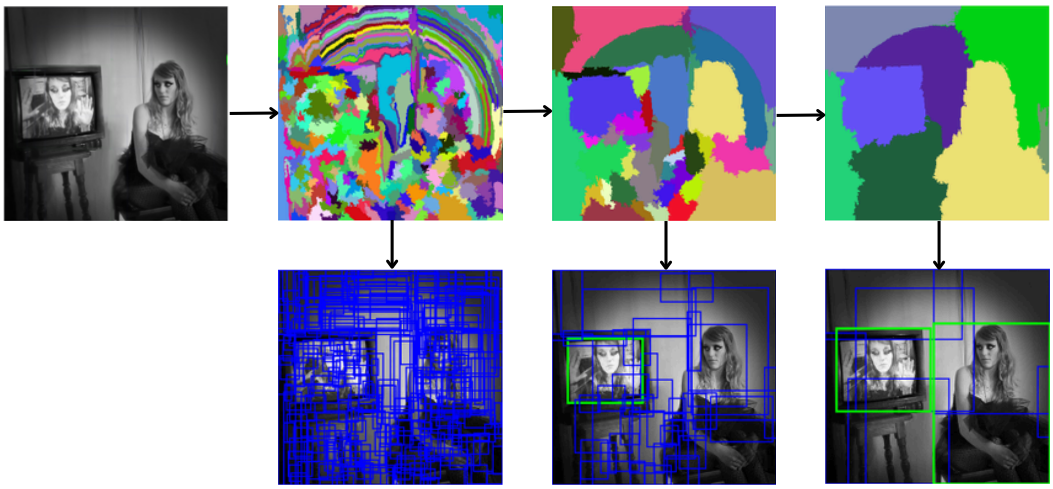


Figure 2.5: Example of results of Felzenszwalb's efficient graph-based segmentation and selective search region proposals [27].

Selective search generates region proposals without requiring the use of sliding windows, which would be computationally expensive. However, one of its main drawbacks is the slow speed, especially when applied to large images or complex scenes.

CNN for feature extraction

After the selective search algorithm generates region proposals, the R-CNN takes each region, resizes it to a fixed size (227×227) to match the input size expected by the network, and feeds it to a CNN, such as AlexNet [14]. This CNN is pre-trained on the ILSVRC 2012 dataset for image classification and serves to extract features from each proposed region. Convolutional layers capture the spatial and hierarchical features of the object in the region, and these are passed to fully connected layers for further processing.

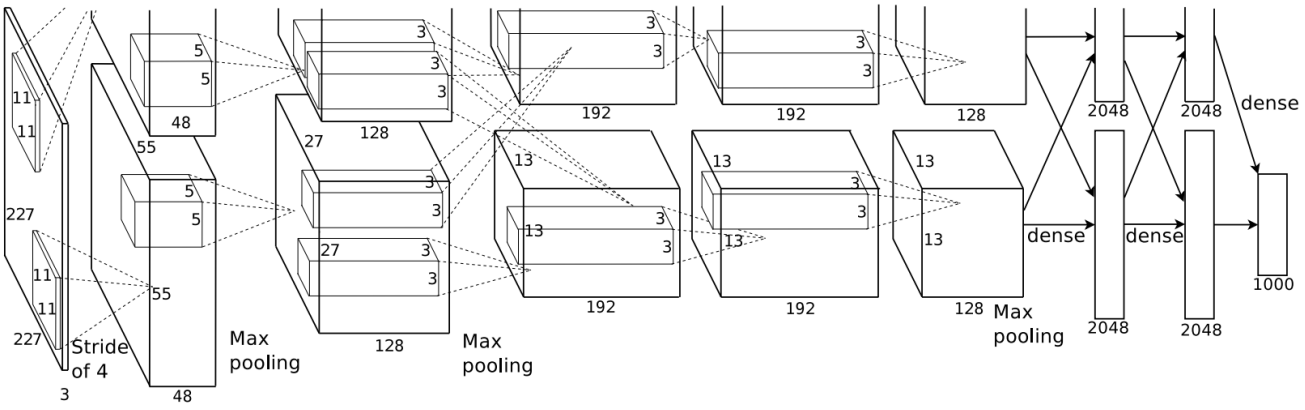


Figure 2.6: AlexNet architecture for two GPUs [14] with correct image size.

Classification and bounding box regression

The extracted features from each region are then fed into a classifier (here it is a linear Support Vector Machine), to determine the presence and type of object within each proposed region. Additionally, a bounding box regression model is employed to refine the predicted bounding boxes for improved localization. This step results in class labels and adjusted bounding boxes for each of the proposed regions.

Non-maximum suppression

After predictions are done, non-maximum suppression (NMS) is applied to the predicted bounding boxes. NMS is a crucial technique employed in object detection algorithms to eliminate redundant bounding boxes and retain only the most significant ones. The process begins with the algorithm generating a set of candidate bounding boxes, each associated with a confidence score that indicates the likelihood of containing an object.
To perform NMS, the algorithm first selects the bounding box with the highest confidence score. This box is considered the best candidate and is marked for retention. Subsequently, all other boxes that exhibit a significant overlap with this selected box, typically measured using the IoU metric, are suppressed. The threshold for suppression is predefined, with common values ranging from 0.3 to 0.7; if the IoU between the selected box and any other box exceeds this threshold, the overlapping box is discarded.
The process is iteratively repeated for the remaining bounding boxes. Each time, the algorithm selects the box with the highest score from the reduced set and suppresses any boxes that overlap with it significantly. This continues until all bounding boxes have been evaluated. The outcome of NMS is a refined set of bounding boxes that maximizes the detection of objects while minimizing redundancy, thus improving the overall accuracy of the object detection system.

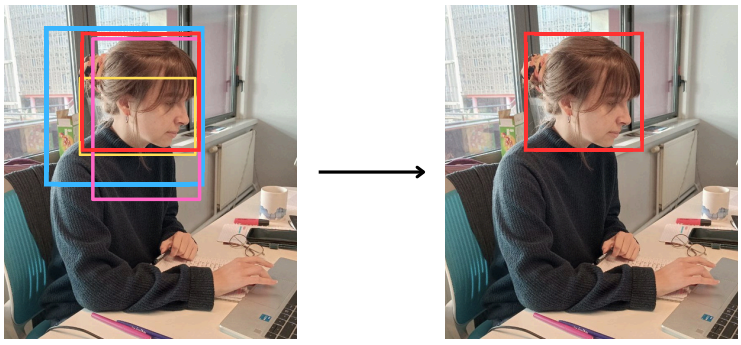


Figure 2.7: Non-maximum suppression result.

4.2 Fast R-CNN

Fast R-CNN was presented in 2015 by Ross Girshick [9]. The framework was proposed as an improvement over R-CNN, which suffers from slow processing speeds, as it requires running the CNN on each of the 2,000 region proposals individually, making it computationally expensive. Additionally, R-CNN relies on resizing region proposals, which may lead to a loss of important spatial information. Instead of performing training for object detection in three steps, Fast R-CNN combines them into single-stage training procedure (Figure 2.8), using a multi-task loss. While R-CNN feeds each region proposal separately into the CNN, Fast R-CNN processes the entire image using a CNN to create a feature map. Region proposals are then applied on top of this shared feature map, which significantly reduces the computational cost. Fast R-CNN introduces the Region of Interest (RoI) pooling layer. This layer extracts fixed-size feature maps for each region proposal from the shared feature map, ensuring uniform input size for the fully connected layers. The feature maps are passed through fully connected layers and two parallel output layers: one for softmax classification and one for bounding box regression.

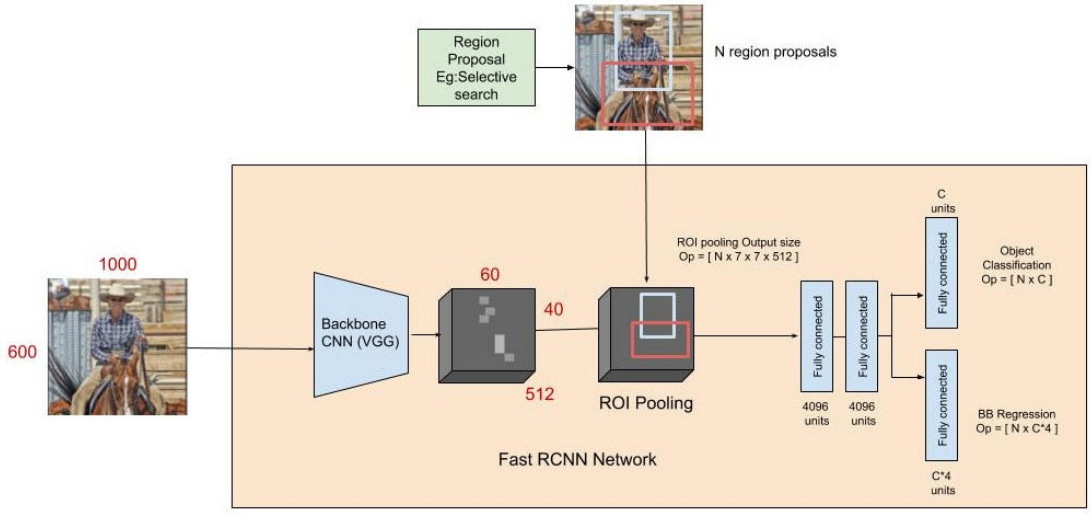


Figure 2.8: Fast R-CNN architecture overview [2].

Fast R-CNN used fine-tuned layers starting from Conv3-1 in VGG16 (Figure 2.9) to generate the mapped feature block of the RoI on the corresponding feature map. Then ROI pooling is used to convert feature tiles to a fixed size and send them to the fully connected layer for classification and positioning. Fast R-CNN uses softmax loss and smooth L1 loss to jointly train classification and bounding box regression in the training process.

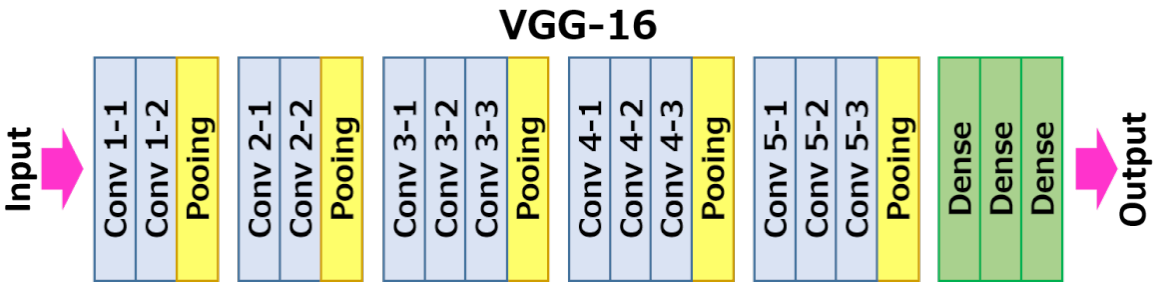
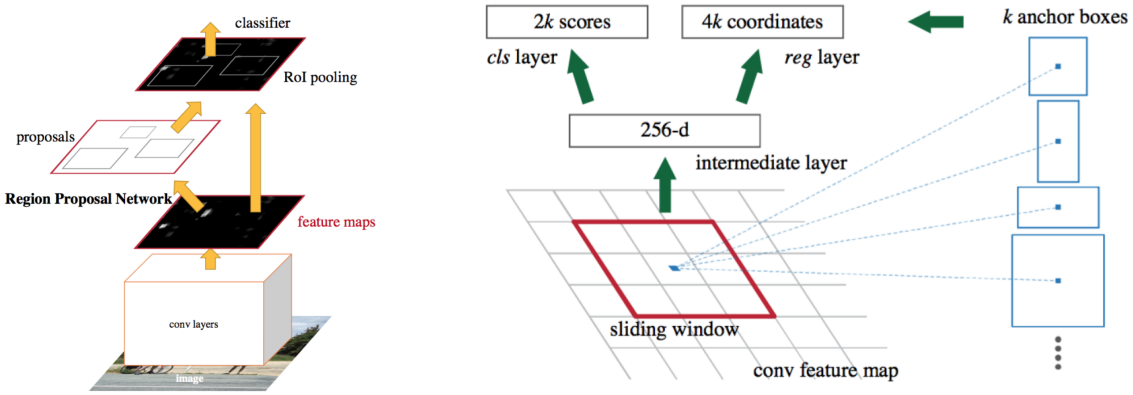


Figure 2.9: VGG16 architecture [11], [26].

This architecture significantly reduces the time required for training and inference by avoiding repeated computations. Additionally, it improves accuracy since the feature extraction is performed on the entire image rather than individual regions. Fast R-CNN achieves superior results on benchmark datasets like PASCAL VOC, offering a considerable speed-up compared to R-CNN without sacrificing detection performance.

4.3 Faster R-CNN

Faster R-CNN [24] is a state-of-the-art object detection framework, another masterpiece of the author Ross Girshick after Fast R-CNN. It also uses VGG-16 as the backbone of the network.
Faster R-CNN further improves the efficiency of Fast R-CNN by eliminating the need for an external region proposal algorithm, like selective search. To obtain these candidate frames more efficiently, Faster R-CNN has added a neural network region proposal network (RPN) that extracts edges (Figure 2.10b). As part of the Faster R-CNN model, it is trained together with the entire model. Realizing the integration of candidate frame extraction into the deep network, RPN can learn how to generate high-quality proposed regions, reduce the number of proposed regions learned from the data, and still maintain the accuracy of object detection.



(a) Faster R-CNN architecture [24].

(b) Region proposal network [24].

Figure 2.10: Faster R-CNN architecture components.

Both the RPN and the detection network, which performs classification and bounding box regression, share the same convolutional layers, significantly reducing computation time (Figure 2.10a). The RPN generates region proposals based on predefined k anchor boxes of different scales and aspect ratios, allowing it to handle objects of varying sizes more effectively. In the original article authors used 3 scales and 3 aspect ratios, yielding $k = 9$ anchors at each sliding position.
Each anchor is labeled as positive or negative. It is positive if the anchor (or anchors) has the highest intersection over union (IoU) overlap with a ground-truth box, or an anchor that has an IoU overlap higher than 0.7 with any ground-truth box. The anchor is negative if it is non-positive and its IoU ratio is lower than 0.3 for all ground-truth boxes. Anchors that are neither positive nor negative do not contribute to the training objective.
Then we can define the multi-task loss function for an image as:

$$L(\{p_i\},\{t_i\}) = \frac{1}{N_{\text{cls}}}\sum_i L_{\text{cls}}(p_i, p_i^*) + \lambda\frac{1}{N_{\text{reg}}}\sum_i p_i^* L_{\text{reg}}(t_i, t_i^*),$$

where, i is the index of an anchor in a mini-batch and p_i is the predicted probability of anchor i being an object. The ground-truth label p_i^* is 1 if the anchor is positive, and is 0 if the anchor is negative. t_i is a vector representing the 4 parameterized coordinates of the predicted bounding box, and t_i^* is that of the ground-truth box associated with a positive anchor. The classification loss is a log loss (cross-entropy) for binary classification:

$$L_{\text{cls}}(p_i, p_i^*) = -(p_i^* \log(p_i) + (1 - p_i^*)\log(1 - p_i^*)).$$

The regression loss is a smooth L1 loss:

$$L_{\text{reg}}(t_i, t_i^*) = \begin{cases} 0.5(t_i - t_i^*)^2 & \text{if } |t_i - t_i^*| < 1, \\ |t_i - t_i^*| - 0.5 & \text{otherwise.} \end{cases}$$

Thus, by using multi-task loss and sharing convolutional layers between the RPN and the detection network, Faster R-CNN achieves both accurate object detection and reduced computational overhead, making it a powerful approach for real-time detection tasks.

5 You Only Look Once (YOLO)

YOLO (You Only Look Once) is a real-time object detection system that was introduced by Redmon et al. in 2015 [23]. The primary idea behind YOLO is to frame object detection as a single regression problem, straight from image pixels to bounding box coordinates and class probabilities, making it extremely fast compared to traditional object detection algorithms. The algorithm runs over the entire image in a single forward pass, making it faster than region proposal-based methods such as Faster R-CNN. YOLO has evolved through multiple versions, each improving on the previous one, and has seen contributions from various researchers and organizations over time.

5.1 YOLOv1

Working Mechanism of YOLOv1

YOLOv1 divides the input image into an $S \times S$ grid. Each grid cell is responsible for predicting a fixed number B of bounding boxes, along with a confidence score and the conditional class probabilities $\mathbb{P}(\text{Class}_i|\text{Object})$ for C object classes. The final output for each grid cell contains $5B + C$ values: $(x, y, w, h, \text{confidence})$ for each bounding box and C probabilities, where

- (x, y) are the coordinates of the center of the box, predicted relative to the bounds of the grid cell;
- (w, h) are the width and height, normalized to the whole image;
- confidence $= \mathbb{P}(\text{Object}) \cdot \text{IOU}^{\text{truth}}_{\text{pred}}$, *i.e.* it reflects the probability that a bounding box contains an object and the accuracy of its location.

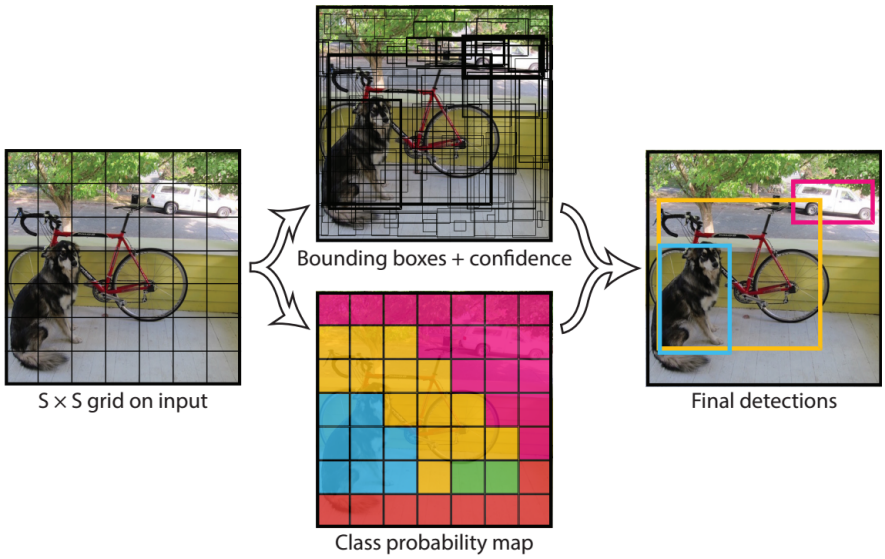


Figure 2.11: YOLO divides the image into an $S \times S$ grid. Each grid cell is responsible for predicting two bounding boxes. Only one object is predicted per grid cell [23].

As shown in Figure 2.11, if the center of an object falls within a grid cell, that grid cell is responsible for detecting it. The predictions are encoded as an $S \times S \times (5B + C)$ tensor. For evaluating YOLO on PASCAL VOC dataset, authors used S = 7, B = 2. PASCAL VOC has 20 labelled classes, so C = 20. The final prediction is a $7 \times 7 \times 30$ tensor (Figure 2.12).
The YOLO network takes a 448×448 RGB image as input and passes it through 24 convolutional layers. These layers progressively reduce the spatial dimensions using stride=(2, 2) and max pooling while increasing the number of channels. The final convolutional layer has a size of 7×7 with 1024 channels, which is then flattened into a vector. A fully connected layer maps this vector to 4096 values, followed by another fully connected layer that produces the output. After the network produces its predictions, a heuristic method removes bounding boxes with low confidence scores and applies non-maximum suppression to keep only the highest-confidence box for each detected object.

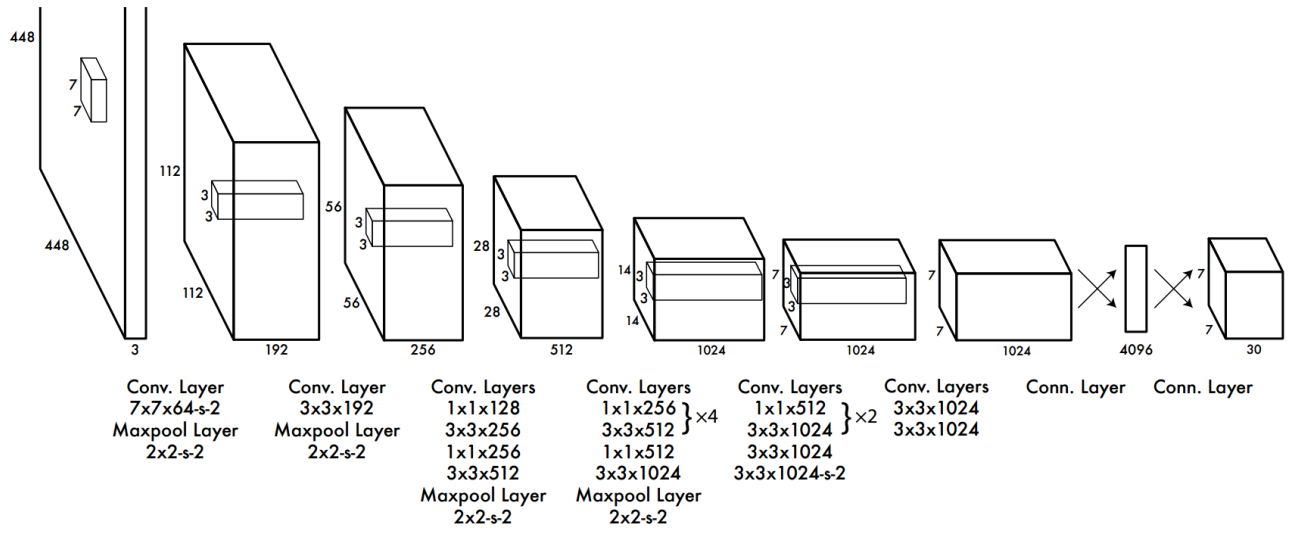


Figure 2.12: The architecture of YOLOv1. The detection network consists of 24 convolutional layers followed by 2 fully connected layers [23].

YOLO loss function

The YOLO loss function is a weighted sum of three components, which helps the network learn both accurate localization and classification:

$$\mathcal{L} = \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 + \left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \tag{2.1}$$

$$+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 + \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \tag{2.2}$$

$$+ \sum_{i=0}^{S^2} \mathbb{1}_{i}^{\text{obj}} \sum_{c \in \text{classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 \tag{2.3}$$

The sum-squared errors are chosen because of ease of optimization. Localization loss (2.1) measures the error in predicting the bounding box coordinates (x, y, w, h). However, it is important to handle the errors for large and small bounding boxes in a different way. Therefore, the square root of the width and height can improve the situation. Confidence loss (2.2) captures the error between the predicted confidence score and the actual IoU for the bounding box. And finally, classification loss (2.3) measures the error between the predicted and actual class probabilities for the detected object.
The parameters λ_{coord} and λ_{noobj} are introduced to address the problem of model instability. Since many grid cells in each image do not contain any objects, the predicted confidence scores of those cells are tending to zero during training, which often gives too much power to the gradient from cells that do contain objects. To fix the problem, the authors increase the localization loss and decrease the confidence loss for the bounding boxes that do not contain objects, putting $\lambda_{\text{coord}} = 5$ and $\lambda_{\text{noobj}} = 0.5$.

Limitations of YOLOv1

YOLOv1 struggles with small objects or objects that are close to each other because of the coarse grid division. YOLOv1 uses a fixed number of bounding boxes per grid cell, limiting the precision of object localization, especially for objects that are not well aligned with the grid.

5.2 Some of later versions

YOLOv2

YOLOv2, introduced by Joseph Redmon and Ali Farhadi in 2017 [22], made significant improvements over YOLOv1. The most notable advancement was the introduction of the anchor mechanism, inspired by Faster R-CNN. Instead of predicting bounding boxes directly, as in YOLOv1, YOLOv2 used predefined anchor boxes to handle objects of varying sizes and aspect ratios more effectively.
In addition, YOLOv2 added several other improvements such as batch normalization to stabilize and speed up training, high-resolution classifier, and multi-scale training. High-resolution classifier involves first training the classification network at higher resolution, and then fine-tuning for detection. Multi-scale training allow the model to generalize better to different input resolutions.
Compared to YOLOv1, YOLOv2 achieved higher accuracy and better handling of small objects, while still maintaining real-time performance.

YOLOv3

YOLOv3, introduced in 2018, further built on the previous versions by using a deeper network architecture called Darknet-53. YOLOv3 improved feature extraction through a deeper architecture, allowing the model to capture more complex patterns. Use of binary cross-entropy loss for class predictions, which better handled multi-label classification problems.
These improvements allowed YOLOv3 to detect objects with greater precision and robustness, particularly small objects, while retaining the speed advantage of previous versions.

YOLOv5

YOLOv5, released in 2020 by Ultralytics [28], was not an official continuation of the previous YOLO versions but became widely popular due to its ease of use and lightweight design. They made a PyTorch implementation of YOLO, which made the model more accessible to a wider range of users, added various pre-trained model sizes (nano, small, medium, large) for different use cases, and integrated modern data augmentation techniques like mosaic augmentation, which will be described in Chapter 3, Section 1.
Also Ultralytics implemented distribution focal loss (DFL) [16], which enhances localization accuracy by refining bounding box predictions using a probability distribution approach rather than directly predicting continuous values. The DFL works as follows. The bounding box coordinates (x, y, w, h) are discretized into a fixed number of bins. This allows the model to output a probability distribution over possible values for each coordinate. Instead of predicting the coordinates directly, the model predicts a classification score for each bin, indicating the likelihood that a coordinate falls within that bin. The DFL loss is calculated using the cross-entropy between the predicted probability distribution and the true distribution for each coordinate.

YOLOv8

YOLOv8, introduced by Ultralytics in 2023 [28], who focused on creating a more modular architecture and improving the model's compatibility with cloud and hardware deployment. Additionally, they enhanced YOLO's performance across detection, segmentation, and classification tasks.
The performance of YOLOv8 heavily relies on the quality and diversity of the training dataset. If the training data lacks variety or has poor quality annotations, the model may perform poorly in real-world applications. While YOLOv8 is designed to be modular and flexible, this can also make it more complex to implement and optimize. There may be trade-offs between accuracy and speed, particularly when using the largest models, which are more accurate, but may be too slow for real-time applications. Like other deep learning models, YOLOv8 can be perceived as a "black box", making it difficult to interpret how decisions are made. Understanding why certain predictions are made can be challenging, which is critical for applications in safety-critical areas. While improvements have been made in detecting small objects compared to previous YOLO versions, YOLOv8 may still struggle with very small objects or objects that are heavily occluded. Although the anchor box mechanism helps, YOLOv8 might still have difficulty with objects of unusual aspect ratios that were not well-represented in the training data.

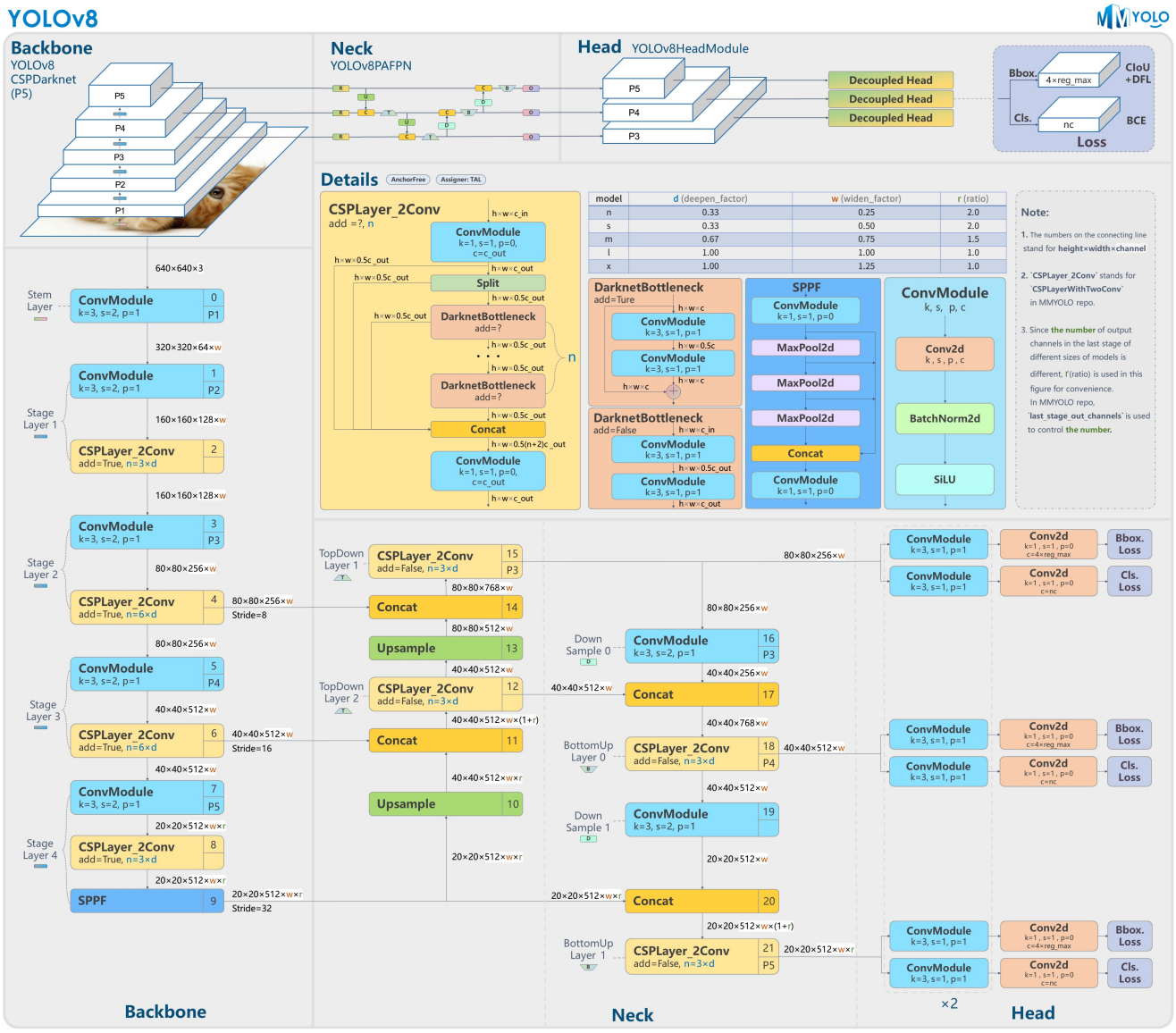


Figure 2.13: YOLOv8 modular architecture [4].

Chapter 3

Application of deep neural networks to endometriosis detection

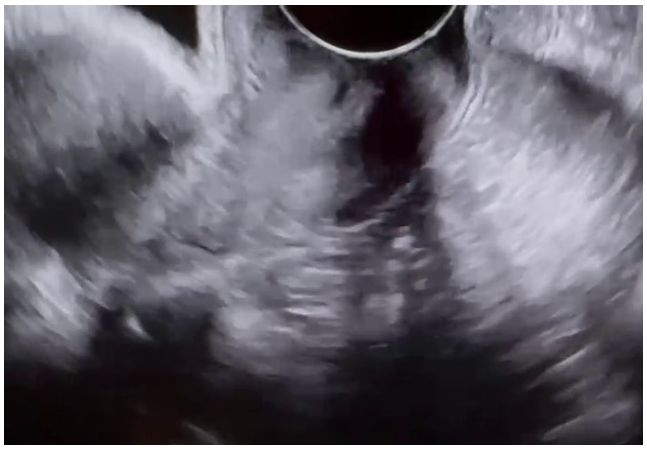
1 Data overview and preprocessing

Our dataset was collected by Dr. Valiere during the internship. It contains 21 ultrasound scan videos from 21 different patients, including 17 videos showing rectal nodules associated with endometriosis and 4 videos from healthy women. Each video lasts between 10 to 18 seconds and captures a transvaginal ultrasound scan of a patient's reproductive organs using traditional ultrasound systems. This method provides detailed images of the uterus, endometrium, ovaries, and rectum.

Data preprocessing

We split these videos into frames, and saved every fifth frame. Considering that we have little data, it is crucial to avoid duplicate images, otherwise, our model will pay more attention to the shapes of nodules that repeat more often. It leads to overfitting, which means that model has good performance on any subset of train set, but has difficulties with recognizing patterns, that it did not see before. We then manually classify all the images with the help of Dr. Valiere. Thus, we obtain 849 images with and 937 images without endometriosis (Figure 3.1).

(a) Image with endometriosis rectal nodule.



(b) Image without endometriosis.

Figure 3.1: Data examples.

For training, we use only images that contain objects, *i.e.* our final dataset consists of 849 images. This choice can be explained by the specifics of work of YOLO and Faster R-CNN. Including a significant number of images without objects in the training dataset could lead to an imbalance, where the model learns more about background features than the objects of interest. This imbalance can result in a less effective model, as it may struggle to generalize well to images containing objects.

Both YOLO and Faster R-CNN are designed to output confidence scores for object detections. If a model is trained extensively on images without objects, it may become less confident in its detections and produce more false positives when encountering images with objects. Ideally, the model should be calibrated to recognize the presence of objects effectively, which is better achieved by focusing on positive samples.

Annotation of images

Once the videos have been transformed into images, it is crucial to annotate the data with objects. Annotation means drawing rectangles (bounding boxes) around the rectal nodules in order to precisely locate their position. Then these bounding boxes are saved in format that is suitable for object detection framework. For example, for YOLO framework the format is (class, x, y, w, h) for each image, where (x, y, w, h) are the coordinates of the center of the bounding box, its width, and height, normalized relative to the image size. All the annotations were meticulously completed by Emilio Picard and me, under the guidance of Dr. Valiere. We used an open source data labeling platform Label Studio to annotate the images, which contain endometriosis (Figure 3.2).

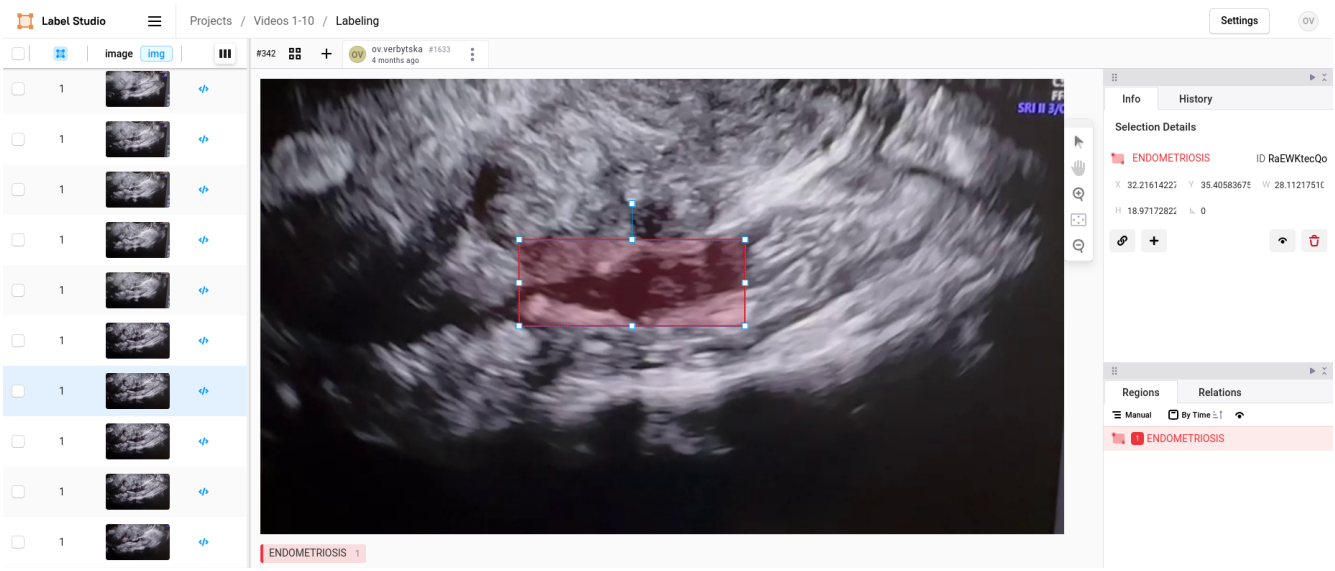


Figure 3.2: Data annotation in Label Studio.

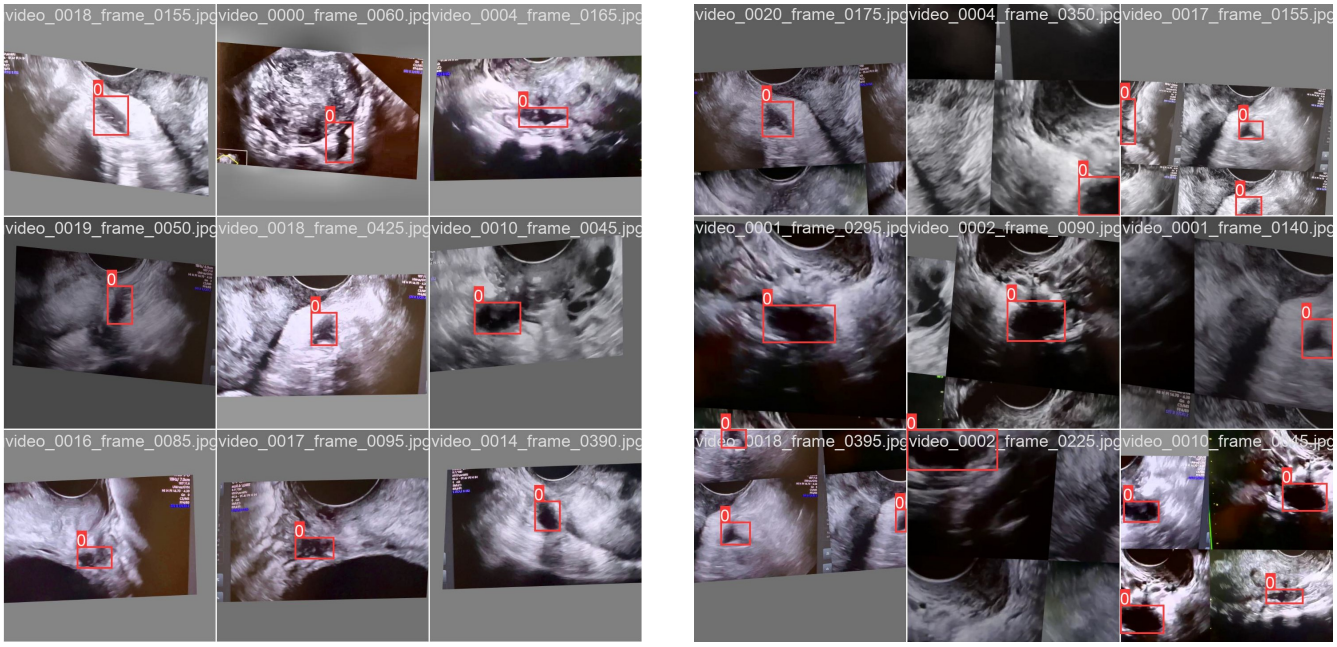
Data augmentation

Since we have little data, it is reasonable to use data augmentation techniques to increase variability of training data and achieve better results. Data augmentation prevents overfitting by training models on slightly modified copies of existing data during each epoch. In the following list we describe the techniques that may be relevant to ultrasound images from data scientist's point of view (Figure 3.3a).

- Adjust the **hue**, **saturation**, and **value** of the image by a fraction of the color wheel, introducing color variability. Helps the model generalize across different lighting conditions.
- **Translate** the image horizontally and vertically by a fraction of the image size, aiding in learning to detect partially visible objects.
- Randomly **erase** a portion of the image during classification training, encouraging the model to focus on less obvious features for recognition.
- **Rotate** the image randomly within the specified degree range, improving the model's ability to recognize objects at various orientations.
- **Shear** the image by a specified degree, mimicking the effect of objects being viewed from different angles.

In fact, the list can be longer considering that CNNs can find other techniques useful as well, which we will show later. Here are extra methods which did not look natural from the data scientist's points of view, but worked well combined with the previous ones (Figure 3.3b):

- **Flip** the image **left to right** with the specified probability.
- **Scale** the image by a gain factor, simulating objects at different distances.
- **Mosaic**: combine four training images into one, simulating different scene compositions and object interactions.



(a) (b)

Figure 3.3: Data augmentations applied to our data.

2 Evaluation

Precise evaluation metrics are essential for assessing the performance of models used for object detection. In this section, we will consider such important concepts as intersection over union (IoU), alongside discussions of precision, recall, and mean average precision (mAP) - metrics that together with techniques such as cross-validation define the framework for evaluating computer vision performance.

Metrics

For classification task we use such metrics as precision and recall. Precision is the proportion of true positives among all positive predictions. Recall is the proportion of true positives among all actual positives. In other words, precision measures the proportion of correctly identified cases among all diagnosed cases, while recall indicates the proportion of identified cases among all actual pathological cases.

When it comes to detection task, the metrics should take into account not only the class of object, but its position on the image. Therefore, it is important to compare the predicted bounding box with the ground truth bounding box. To perform this we define confidence score and intersection over union (IoU). Confidence score is the probability that an anchor box contains an object. Do not confuse it with confidence score used in YOLO, which takes into account both the result of classification and location of predicted bounding box.

$$\text{IoU} = \frac{\text{Area of intersection}}{\text{Area of union}} = \frac{\text{Area of intersection}}{\text{Ground truth area} + \text{Predicted box area} - \text{Area of intersection}}$$

A detection (class probabilities, bounding box coordinates) is considered a true positive (TP) only if it satisfies all of the following conditions:

1. Confidence score is greater than some threshold t_{conf}.
2. The predicted class is the same as the class of a ground truth.
3. The predicted bounding box has an IoU greater than a threshold (usually 0.5 or more) with the ground truth.

In case of false positives (FP) it is possible to get multiple FPs having one ground truth bounding box. We get a FP for each predicted bounding box, if one of the conditions 2 or 3 is not satisfied.
Every single not correctly predicted ground truth bounding box is a false negative (FN), *i.e.* there is no prediction for a ground truth bounding box, which satisfies conditions 1-3.
When the confidence score of a detection, that is not supposed to detect anything, *i.e.* detect background, is lower than the threshold, the detection counts as a true negative (TN). However, in object detection we usually do not care about these kind of detections.
Thus, we obtain precision and recall for object detection.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}, \quad \text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}.$$

It should be noted that the formulas for precision and recall are the same for classification and detection tasks, but the derivation process is different.
By changing the threshold for confidence score t_{conf} from 0 to 1, we get different pairs of precision and recall, which can be represented as a 2D plot with recall on the x-axis and precision on the y-axis. The resulting curve is called a precision-recall curve (or PR curve). We can notice, that recall increases monotonically, while precision can go up and down, but the general tendency is to decrease.
While the precision-recall curve is useful for evaluating a detector's performance, it becomes challenging to compare different detectors when their curves intersect on one plot. A numerical metric that allows to perform direct comparison is preferable, and this leads to definition of average precision (AP). AP is derived from the precision-recall curve and essentially represents the precision averaged across all unique recall levels.
To minimize the effect of fluctuations in the curve, precision is first interpolated at several recall levels before calculating AP. There are two approaches to selecting the recall levels at which precision is interpolated. The traditional method involves choosing 11 equally spaced recall levels (i.e., 0.0, 0.1, 0.2, ... 1.0). A newer standard selects all unique recall levels present in the data. It is considered more effective at enhancing precision and detecting differences between methods, especially when dealing with low AP. Figure 3.4 illustrates the interpolated precision-recall curve derived from the original curve using the newer standard. The interpolated precision p_{ip} at a certain level r is defined as the highest precision found for any recall level $r' \geq r$:

$$p_{ip}(r) = \max_{r' \geq r} p(r')$$

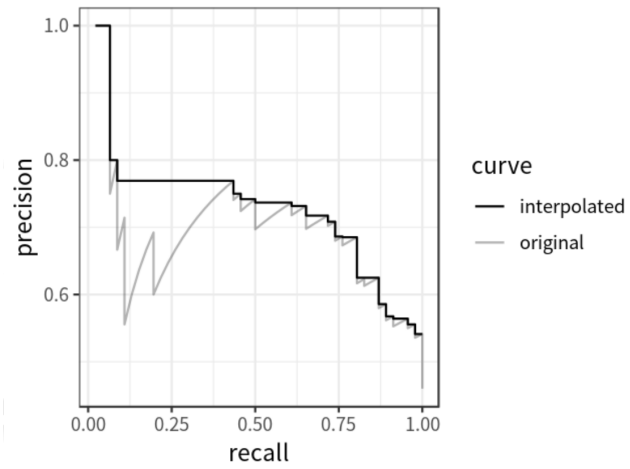


Figure 3.4: Original and interpolated precision-recall curve [29].

AP is defined as area under the interpolated precision-recall curve, which can be calculated using the following formula:

$$\text{AP} = \sum_{i=1}^{n-1} (r_{i+1} - r_i) p_{ip}(r_{i+1}),$$

where $r_1, r_2, ..., r_n$ are the recall levels in an ascending order.
Since AP only involves one class we would like to generalise this metric for $K > 1 classes$. Mean average precision (mAP) is defined as the mean of AP across all K classes:

$$\text{mAP} = \frac{\sum_{i=1}^{K} \text{AP}_i}{K}$$

There exist several mAP metrics using different thresholds:

- mAP50: computes the area under the precision-recall curve across multiple classes, calculated at an IoU threshold of 50%;
- mAP50-95: the average of the mean average precision calculated at varying IoU thresholds, ranging from 50% to 95%;
- mAP-small: mAP50-95 for small objects that covers area less than 32^2 pixels;
- mAP-medium: mAP50-95 for for medium objects that covers area greater than 32^2 but less than 96^2 pixels;
- mAP-large: mAP50-95 for large objects that covers area greater than 96^2 pixels;
- fitness (in YOLO): $0.1 \cdot \text{mAP50} + 0.9 \cdot \text{mAP50-95}$.

Cross-validation

Cross-validation is a technique used to assess the performance of a machine learning model by dividing the data into multiple subsets or "folds." The model is trained on some folds and tested on the remaining ones. This process is repeated multiple times, and the results are averaged to provide a more reliable estimate of the model's performance. It helps in avoiding overfitting and gives a better sense of how the model will generalize to unseen data, providing a more accurate measure of its real-world effectiveness. The most common type is k-fold cross-validation, where the data is split into k equal parts, with each part being used once as a test set and $k-1$ parts as the training set.
Since our images are not independent or simply repeat within one video, it is more reasonable to split images by the video id. For each video, we repeat the following steps:

1. Use all saved frames from the selected video as the test set.
2. Shuffle the frames from the remaining videos and split them into a training set (80%) and a validation set (20%).
3. Train and evaluate the model. Save the results.

Thus, we perform k-fold cross-validation with $k = 17$ unequal parts of data.

3 Training and results

This section presents the results obtained from applying YOLO and Faster R-CNN frameworks to our data for object detection. The primary aim was to assess the performance of these models in accurately classifying and detecting endometriosis lesions.

3.1 YOLO

Yolov8n is a version of the YOLO family of models, introduced by Ultralytics in early 2023 [28]. The "n" in *yolov8n* stands for "nano", indicating that it is the smallest and lightest version of the YOLOv8 architecture. It is designed to be highly efficient in terms of both computation and memory usage, making it suitable for the scenarios where processing power is limited with memory constraints. Due to its compact size, *yolov8n* is extremely fast, achieving real-time performance even on low-end hardware (e.g., mobile devices or embedded systems). It is less accurate than larger versions, but the lower computational requirements make it more practical in real-world applications. For training we used Ultralytics' *yolov8n* pretrained on the COCO dataset, which is a large-scale object detection dataset containing over 80 different classes (such as people, animals, vehicles, etc.). We then fine-tune it for our dataset. This helps in achieving better performance with less training data and computation.

Parameters

In order to have reproducible results we set the following parameters

- image size = (640, 640, 3);
- batch size = 16;
- optimizer AdamW with learning rate equal to 0.002, and momentum equal to 0.9;
- number of epochs = 100.

The speed of training *yolov8n* with these parameters for our dataset ranges from 2 to 5 seconds per epoch on NVIDIA A100 with 40 GB memory.

Data augmentation

The data augmentation applied in our process are designed to enhance the diversity and robustness of the training images. Let us denote the set of the following data augmentation methods as (A). The specific augmentations and their corresponding parameters are as follows:

- Color adjustments (hue, saturation, value): We adjust the hue by a factor of 0.015, meaning the overall color tone of the image is slightly shifted. Saturation is adjusted by 0.7, intensifying or dulling the colors as needed. Lastly, the value (brightness) is modified by 0.4, altering the brightness or darkness of the image.
- Translation: The image is translated (shifted) up to 10% of its size, meaning that the content can move horizontally or vertically by up to 10% of the total image dimensions. This introduces slight positional variations without distorting the main features.
- Erasing: There is a 40% probability of randomly erasing portions of the image. This technique, known as "random erasing", removes certain sections of the image to encourage the model to focus on more generalized patterns rather than specific areas.
- Rotation: The image can be rotated randomly within a range of -5 to 5 degrees. This helps account for minor rotations that could occur in real-world scenarios without dramatically changing the orientation.
- Shear transformation: We apply a shear factor of 5, meaning the angle of view is altered, distorting the object in the image as if viewed from a slanted perspective. This simulates variations in the camera angle or object alignment.
- Horizontal flip: There is a 50% chance of flipping the image horizontally (left to right), allowing the model to learn from both orientations of the objects.
- Scaling: The image is scaled down by up to 50%, meaning it can be reduced in size, helping the model become robust to variations in object size.
- Mosaic: A mosaic transformation is applied with a probability of 100%, meaning it is always applied. This technique merges several images into one, enhancing the dataset's variability by combining different objects and backgrounds into a single training image.

It should be noted that in last 10 epochs mosaic is not applied. In the final epochs the model needs to refine its decision-making based on real-world images rather than augmented or artificially complex ones, ensuring better precision.

Results

After performing cross-validation over 100 epochs we can compare the medians of the train (Table 3.1) and test (Table 3.2) metrics. Since there are 3 outlier values (Figure 3.5) for precision for test data and the total amount of values is 17, average values of metrics are not very informative. The testing data, which cause the occurrence of the minimal metrics' values, are from the video which contains large endometriosis lesion of unusual (for our model) form and colour, which is not seen in any other video (Figure 3.7a). The frames of this video bring more diversity in our dataset, and it is good for future training, when we have more data.

Train	Precision	Recall	mAP50	mAP50-95	fitness
Without data augmentations	0.94	0.91	0.96	0.66	0.68
Data augmentations (A)	0.96	0.97	0.98	0.68	0.71
(A) without flip, scale, and mosaic	0.96	0.94	0.97	0.66	0.69
(B): (A) without rotation and shear	0.97	0.97	0.98	0.7	0.73

Table 3.1: YOLO v8n train median metrics for 17-fold cross-validation.

Test	Precision	Recall	mAP50	mAP50-95	fitness
Without data augmentations	0.87	0.74	0.85	0.5	0.52
Data augmentations (A)	0.92	0.86	0.94	0.52	0.56
(A) without flip, scale, and mosaic	0.94	0.88	0.95	0.51	0.55
(B): (A) without rotation and shear	0.97	0.96	0.99	0.62	0.65

Table 3.2: YOLO v8n test median metrics for 17-fold cross-validation.

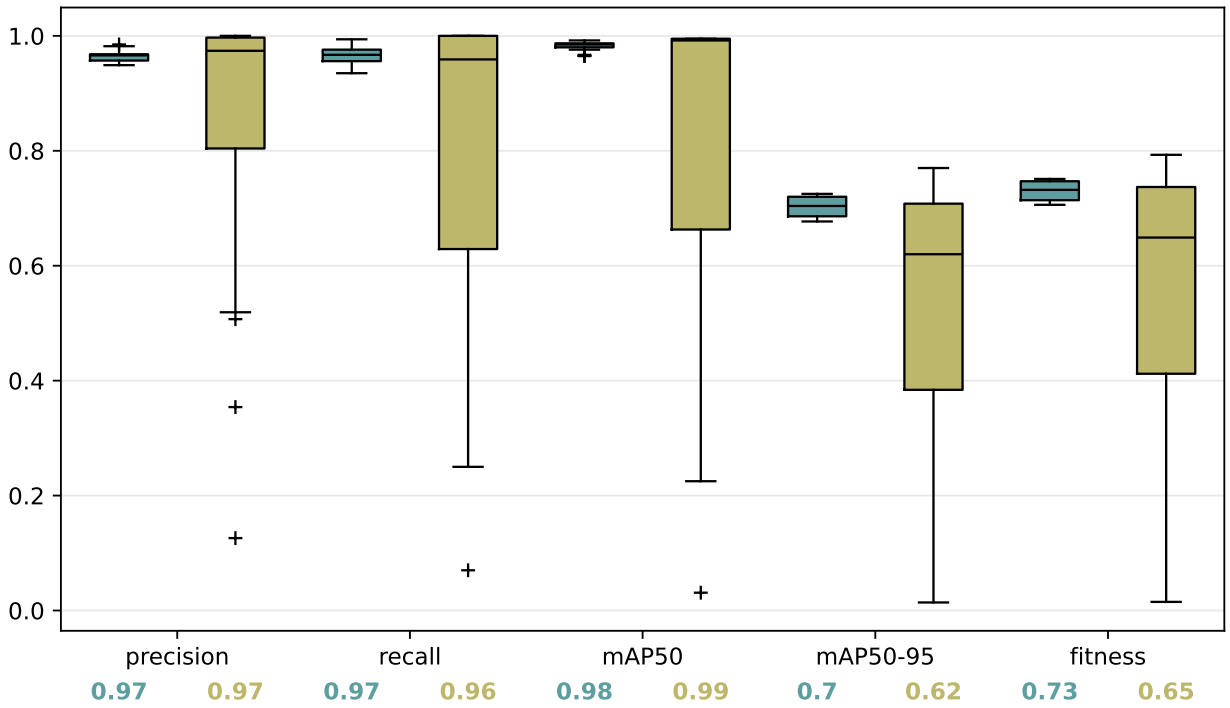


Figure 3.5: Boxplots of training (teal) and testing (yellow) metrics for *yolov8n* with data augmentation (B) during 100 epochs.

We denote better data augmentation as (B), in our case it is (A) without rotation and shear. Let us consider boxplots of training and testing metrics for *yolov8n* with data augmentation (B) (Figure 3.5). Training metrics (teal) exhibit consistently high values, generally hovering around 0.97 to 0.99. This indicates that the model performs very well on the training data, with high precision (minimizing false positives) and recall (minimizing false negatives). Testing metrics (yellow), however, show lower median values with greater variability. Precision and mAP are generally lower, especially for mAP50-95, which drops to around 0.65. This suggests that the model struggles to generalize to the test data, implying some level of overfitting despite data augmentation. However, for *yolov8n*, a good mAP50-95 score typically falls within the range of 0.35 to 0.5 for general object detection tasks, which means that obtained results with median mAP50-95 equal to 0.62 can be considered as good.

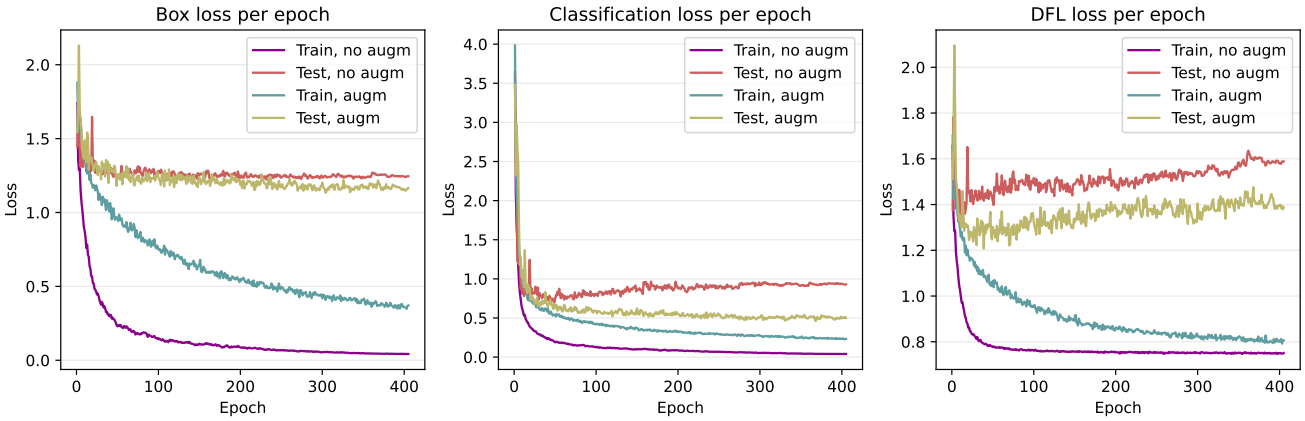
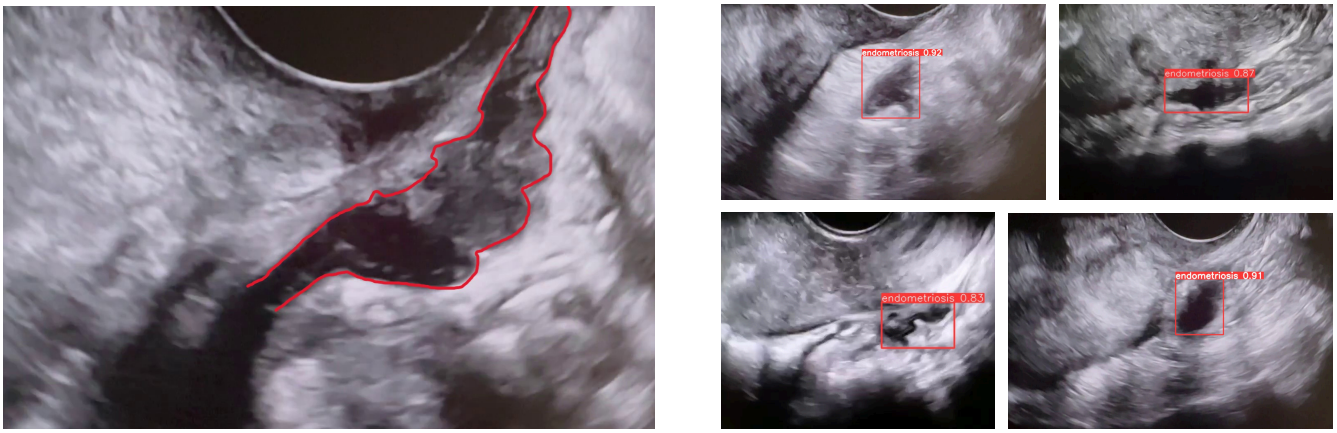


Figure 3.6: Train and test losses within epoch for dataset without data augmentation and with data augmentation (B).

Although the training process involves some randomness due to shuffling data to form mini-batches, we can still compare the trends in different types of losses for both the training and test data across two training processes (Figure 3.6): one using data augmentation (B) and another without any augmentations. In general, using data augmentation (B) results in lower test losses and improved generalization across all loss categories, while models trained without augmentation show signs of overfitting and higher test losses. However, the training loss is decreasing much faster in a model without data augmentation.



(a) A frame from the outlier video.

(b) Predictions for images with rectal nodule.

Figure 3.7: Examples of data and predictions.

As follows, for *yolov8n* with the data augmentations (B) we visualize our predicted detections along with the class name and confidence score, where the score is calculated as $\mathbb{P}(\text{Object}) \cdot \text{IOU}_{\text{pred}}^{\text{truth}}$. For images containing endometriosis rectal nodules, the detections show high confidence, ranging from 0.75 to 1.0 (Figure 3.7b). On the other hand, "empty" images can also receive predictions, though less frequently, with confidence scores ranging from 0.1 to 0.7 (Figure 3.8). To manage these false positive detections, we apply a confidence threshold of 0.75.

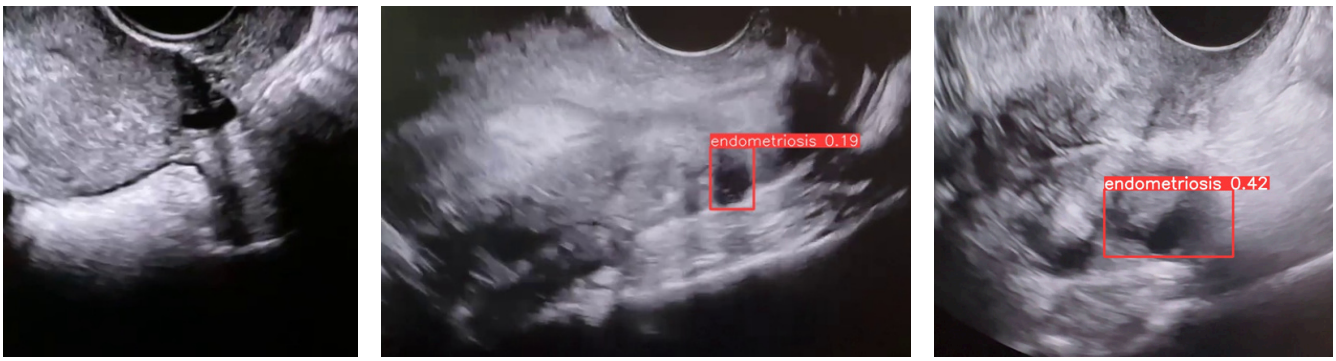


Figure 3.8: Predictions for images without rectal nodules.

Using our best model with a confidence threshold, we test an entire video, *i.e.* we perform detections on every frame and then reassemble the video. The results are the following. For the four videos that do not contain any pathologies, we do not see any detections at all. For a video with endometriosis we choose the model which did not see this video during training, and finally we obtain nice results even for the outlier video (Figure 3.7a), where we can see a few frames containing detections, despite poor metric values (the lowest values in Figure 3.5).

3.2 Faster R-CNN

We utilize the *faster_rcnn_R_50_C4_3x* model, which is a variant of the Faster R-CNN architecture for object detection implemented by Facebook AI Research in Detectron2 library [6].
The part of name *R_50* refers to the ResNet-50, which is a residual network that is 50 layers deep. It's used here for feature extraction, providing a good balance between computational efficiency and representation power. This indicates the backbone architecture configuration. The *C4* variant of ResNet50 uses the 4-th convolutional block of ResNet50 to output feature maps. In the original ResNet50 architecture (Figure 3.9), the output feature maps from different stages (C1, C2, C3, etc.) can be used, and C4 corresponds to a higher-level feature map from the 4-th block. *3x* is a training schedule, *i.e.* the model is trained for three times the number of iterations as the "default" or "1x" schedule. In Faster R-CNN models, the default schedule is typically around 90,000 iterations (when the model processes a single mini-batch of data). The *faster_rcnn_R_50_C4_3x* model in Detectron2 is pretrained on the ImageNet dataset with 3x schedule, *i.e.* 54 epochs with mini-batch size of 256 images.

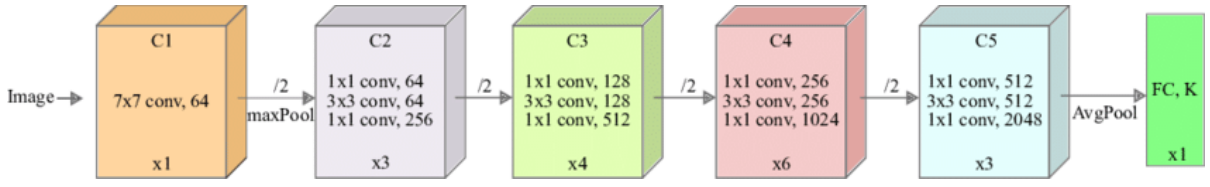


Figure 3.9: ResNet-50 simplified architecture. The bottom of each block denotes the repeated set of layers (Kernel size, operation, number of channels) in each convolution block (C). After each block (C1 to C5) the spatial dimension is reduced. The last layer is a fully connected layer with K units. Shortcut connections occur every two layers but are omitted for readability [25].

Parameters

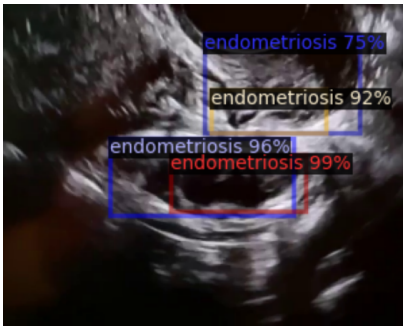
In order to have reproducible results we set the following parameters

- image size = (800, 800, 3);
- batch size = 16;
- optimizer SGD with learning rate equal to 0.002, and momentum equal to 0.9;
- number of epochs = 5000.

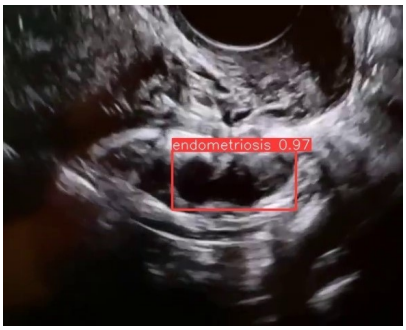
The speed of training *faster_rcnn_R_50_C4_3x* with these parameters for our dataset is up to 1 second per epoch on NVIDIA A100 with 40 GB memory.

Comparison with YOLOv8

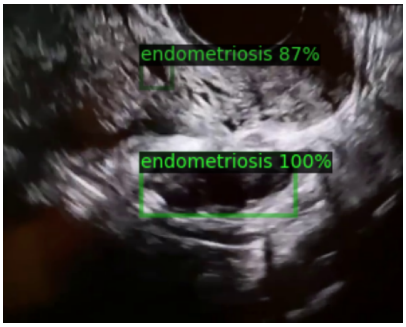
During *faster_rcnn_R_50_C4_3x* training we did not apply any data augmentations due to complexity of implementation. Therefore, we will discuss the detections and performances of *faster_rcnn_R_50_C4_3x* by comparing them with *yolov8n* model without data augmentation.
Both models use NMS with IoU equal to 0.5, but YOLOv8 makes better predictions for our dataset, even though they have lower confidence (Figure 3.10).



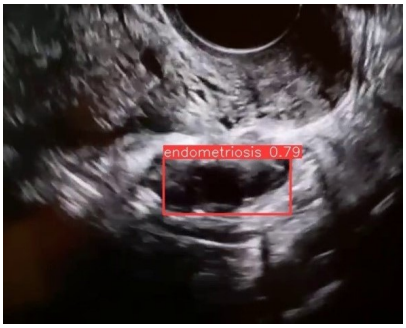
(a) Image 1, Faster R-CNN detection.



(b) Image 1, YOLOv8 detection.



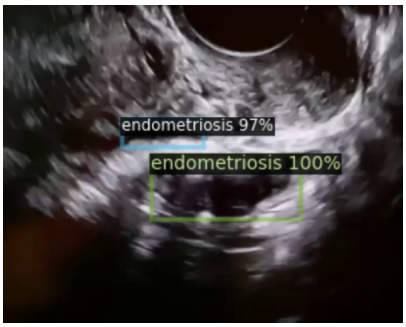
(c) Image 2, Faster R-CNN detection.



(d) Image 2, YOLOv8 detection.

Figure 3.10: Comparison of our models' detections.

Even if we set a confidence threshold equal to 0.95 for Faster R-CNN model, we still get a lot of false positives and false negatives (Figures 3.11a and 3.11b). Meanwhile, setting a threshold for YOLOv8 excluded false positives entirely.



(a) False positive (blue detection).

(b) False negative.

Figure 3.11: Detections of Faster R-CNN with confidence threshold equal to 0.95.

As we can see in Figure 3.12, Faster R-CNN model has better mAP50 for testing sets and higher median, and almost the same results for mAP50-95. It should be noted that both of the models overfit. If we look at the median values of mAP50, Faster R-CNN model seems to be less prone to overfitting, than the one of YOLOv8. However, the test metrics for both models have a lot of variability, therefore, we cannot be certain in our conclusions, unless we have more data.
Analysing the Figure 3.12, we would like to say that Faster R-CNN is better model for our dataset. However, as we have seen the examples of detections this model makes, which was very important to analyse, we can agree that YOLOv8's detections are better.

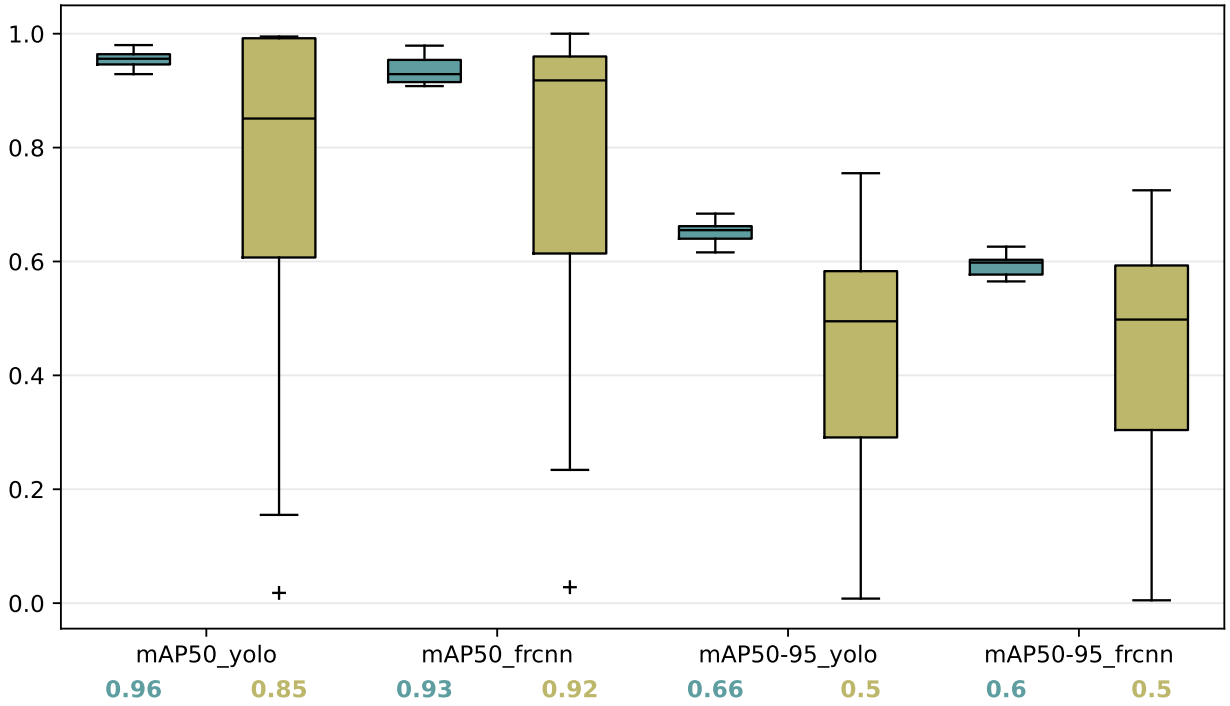


Figure 3.12: Boxplots of training (teal) and testing (yellow) metrics for *yolov8n* during 100 epochs and for *faster_rcnn_R_50_C4_3x* during 5000 epochs, both without utilisation of data augmentation.

Chapter 4

Development and feasibility study of the ECHOPICT tool: an AI-driven solution for endometriosis detection

Through the technical research carried out so far, we have conducted a study on the potential projection of an ECHOPICT tool to be offered to hospitals and gynecological practices to assist healthcare professionals in detecting rectal nodules. At the end of the purely technical work carried out using statistical tools, we aim to apply this research to entrepreneurship. We envisioned the creation of a computer tool, in the form of software, to be integrated into ultrasound machines. The following description is of an entrepreneurial project, undertaken during June and July, in collaboration with Emilio Picard. This work lasted six weeks, at a frequency of two hours per week. We were able to carry out this project through the Pepite initiative (student center for innovation, transfer, and entrepreneurship) of Sorbonne University, and we also received the status of student entrepreneurs along with Emilio. Pepite is a portal for entrepreneurship for all students of the Alliance Sorbonne University, from Bachelor's to Doctorate level. The mission of Pepite Sorbonne University is to raise awareness among students and young graduates about the spirit of entrepreneurship and, above all, to support them in the construction and realization of their entrepreneurial and intrapreneurial projects. Pepite directs all project leaders towards the actors and support systems best suited to their needs. In this context, and in collaboration with Pepite, we explored various aspects of the development of a new product, addressing questions concerning human and financial resources, applicants and needs, existing solutions, the balance of power between different organizations, and the viability of the product. We examined these issues through two main axes: the resources available and the feasibility of the tool. These two elements are crucial for the creation of an entrepreneurial project. Resources provide the necessary means to start and sustain the project, while feasibility ensures that the project is viable, realistic, and well-planned. Together, these elements increase the project's chances of success and sustainability. In the following, we assume that the ECHOPICT tool is ready, and that, in addition to the technical work, we have developed the software that we intend to propose at the conclusion of future work.

Means Available

The development of an endometriosis detection tool using artificial intelligence (AI) relies on human expertise in deep learning and medical image processing. This expertise is essential for designing models capable of accurately analyzing medical images and identifying signs of the disease. Additionally, close collaboration with medical experts is crucial to validate the models' predictions, ensure the clinical relevance of the results, and guarantee that the tool meets the real needs of patients and healthcare professionals. The main contributors to the project include Dr. Valière for her medical expertise, and Baptiste Gregorutti, Emilio Picard, and myself for the technical development of the tool.

Feasibility of the Tool

Hospitals, clinics, and private gynecological practices are key partners for the development and testing of the ECHOPICT tool. They provide the medical data necessary for training the algorithms and participate in clinical trials to validate the tool's performance. Companies specializing in IT can also contribute to the technical

development of the tool by providing expertise in image processing, software development, and data security. Collaborations with research organizations and universities help integrate the latest scientific and technological advancements, which are essential for innovation and continuous improvement of the tool. Through these partners, an entrepreneurial project requires other human and financial resources. The development team should comprise data scientists, engineers, software developers, and medical image processing specialists. A multidisciplinary team is essential to cover all technical aspects of the project. The development and launch of the tool require significant financial investment. Funding sources may include research grants, private investments, partnerships with technology companies, and public funds. In the future, we could seek financial support for product development by responding to AI and medicine-oriented research calls for projects. The primary users of the ECHOPICT program are patients with endometriosis and gynecology specialists. An accurate and rapid diagnosis can reduce treatment times and improve patients' quality of life. Hospitals and clinics require reliable diagnostic tools to optimize resource management and improve the quality of care. ECHOPICT can address these needs by increasing diagnostic accuracy and reducing the costs associated with diagnostic errors. ECHOPICT would also meet essential needs such as treating a serious illness, identifying the cause of pain, providing rapid diagnosis, and offering training for diagnostic purposes. The tool would offer valuable training for detecting rectal nodules, but most importantly, it would contribute to understanding endometriosis.

Feasibility Assessment

To properly assess the usefulness and feasibility of the tool, it is necessary to explore existing solutions to the problem we aim to address. There are already solutions for diagnosing endometriosis, but they are often limited by their accuracy and cost. Additionally, existing AI-based tools focus on MRI images, while coelioscopy, a highly accurate diagnostic method, involves complex and invasive procedures. Patients are seeking alternative treatments that offer the same precision but are less invasive. As stated, patients require reliable information from ultrasound scans. ECHOPICT distinguishes itself by its accuracy, adaptability to different machines, and the healthcare professionals involved. It provides real-time analysis of rectal nodules during ultrasound consultations. The technical feasibility of the tool depends on the quality of the available data, the performance of the algorithms, and the compatibility with ultrasound machines. Clinical trials and technical validations are necessary to ensure the tool's reliability. This project requires close collaboration between different organizations, and it could potentially lead to competitive tensions. We evaluate two types of stakeholders in the ecosystem: actors and influencers. Hospitals, specialist doctors, patients, and midwives would test and use the product, acting as key stakeholders. Influencers, such as the World Health Organization (WHO), universities, APHP (Assistance Publique – Hôpitaux de Paris), or the media, could help in deploying the ECHOPICT tool. Market analysis indicates a growing demand for accurate and rapid diagnostic tools for endometriosis. The potential market includes hospitals, clinics, research centers, and specialized medical practices. The ECHOPICT project's business model could include user licenses, subscriptions, maintenance services, and training programs. Endometriosis detection via ultrasound is of great interest to many and addresses critical needs. The technical work we've completed would allow us to engage in discussions with doctors and raise awareness of our proposal. For specialized doctors training students (medical interns), the learning program would allow them to study different types of endometriosis in a fun and intuitive way, helping them recognize the condition. For specialists struggling to detect rectal nodules, ECHOPICT's AI functionality would enable efficient and timely detection of rectal nodules, allowing treatment to begin before the disease progresses too far. Pepite, as a portal for entrepreneurship, played a crucial role in fostering our entrepreneurial spirit and supporting us throughout our project. By examining human and financial resources, applicant needs, existing solutions, and product viability, we have made significant progress in public health, opening the door to future collaborations to improve the diagnosis and treatment of endometriosis.

Chapter 5

Conclusions

During this internship we explored the application of deep learning algorithms for the detection of endometriosis in ultrasound videos. Specifically, we focused on evaluating the performance of YOLO and Faster R-CNN frameworks for the task of detecting rectal nodules. While both models demonstrated potential, YOLOv8n, particularly with data augmentation, achieved better overall results, showing high accuracy and low error rates across multiple videos. Moreover, cross-validation testing has shown that our YOLOv8 model did not detect false positives on our dataset. However, we also observed challenges related to variability in the ultrasound images, where outlier videos posed difficulties due to unique patterns not present in the training set. This highlights the need for more diverse datasets to improve the model's generalization capabilities.
The development of the ECHOPICT tool as an AI-driven solution for endometriosis detection shows promise in improving diagnostic accuracy and reducing reliance on clinical expertise for ultrasound interpretation. Future research should focus on expanding the dataset, addition of precise annotations, refining the model, and further validating its use in clinical settings. Additionally, there is potential for detecting other types of endometriosis (*e.g.* ovarian). With continued improvements, AI-based diagnostics tool like ECHOPICT could significantly reduce diagnostic delays and improve the quality of life for those affected by endometriosis.

Further discussion

In August and September, 2024, there were published few articles, which may be interesting for further development of the project.
Y. Li et al. [17] worked on classification and segmentation of ovarian endometriomas on ultrasond images. The results of segmentation are fused with the classification features with the assistance of the attention mechanism. The proposed model achieved a classification accuracy of 91.36%.
W. Figueredo et al. [7] contributed to endometriosis rectal and sigmoid colon nodules segmentation on MRI scans. Using an ensemble of networks, classification of images and patients, with or without endometriosis, they achieved accuracies of 87.46% and 96.67%, respectively.
A. Podda et al. [19] worked on endometriosis segmentation on ultrasound transvaginal scans. They proposed a method which allows to significantly improve the performance of the individual neural networks in the presence of a limited training set.
Since all of these articles focus on ultrasound images or limited datasets, we could draw inspiration from some of the proposed techniques to enhance the performance of our models. Furthermore, advancing to a segmentation task could be a valuable next step in the ECHOPICT project.

Bibliography

[1] Agarwal, Sanjay K. et al. "Clinical diagnosis of endometriosis: a call to action". In: *American Journal of Obstetrics and Gynecology* 220.4 (2019), 354.e1–354.e12. ISSN: 0002-9378. DOI: https://doi.org/10.1016/j.ajog.2018.12.039.

[2] Ananth, Shilpa. *Fast R-CNN for Object detection*. Medium. Aug. 30, 2019. URL: https://towardsdatascience.com/fast-r-cnn-for-object-detection-a-technical-summary-a0ff94faa022.

[3] Berker, Bulent and Seval, Murat. "Problems with the Diagnosis of Endometriosis". In: *Women's Health* 11.5 (Sept. 1, 2015), pp. 597–601. ISSN: 1745-5057. DOI: 10.2217/whe.15.44.

[4] *Brief summary of YOLOv8 model structure · Issue #189 · ultralytics/ultralytics*. GitHub. URL: https://github.com/ultralytics/ultralytics/issues/189.

[5] Bulletti, Carlo et al. "Endometriosis and infertility". In: *Journal of Assisted Reproduction and Genetics* 27.8 (Aug. 2010), pp. 441–447. ISSN: 1058-0468. DOI: 10.1007/s10815-010-9436-1.

[6] *facebookresearch/detectron2*. URL: https://github.com/facebookresearch/detectron2.

[7] Figueredo, Weslley Kelson Ribeiro et al. "Automatic segmentation of deep endometriosis in the rectosigmoid using deep learning". In: *Image and Vision Computing* 151 (2024), p. 105261. ISSN: 0262-8856. DOI: https://doi.org/10.1016/j.imavis.2024.105261.

[8] Fukushima, Kunihiko and Miyake, Sei. "Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position". In: *Pattern Recognition* 15.6 (1982), pp. 455–469. ISSN: 0031-3203. DOI: https://doi.org/10.1016/0031-3203(82)90024-3.

[9] Girshick, Ross. *Fast R-CNN*. 2015. arXiv: 1504.08083 [cs.CV]. URL: https://arxiv.org/abs/1504.08083.

[10] Girshick, Ross et al. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. 2014, pp. 580–587. DOI: 10.1109/CVPR.2014.81.

[11] Hassan, Muneeb ul. *VGG16 - Convolutional Network for Classification and Detection*. Nov. 20, 2018. URL: https://neurohive.io/en/popular-networks/vgg16/.

[12] *ImageNet Large Scale Visual Recognition Competition 2011 (ILSVRC2011)*. URL: https://www.image-net.org/challenges/LSVRC/2011/results.php.

[13] Ioffe, Sergey and Szegedy, Christian. *Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift*. 2015. arXiv: 1502.03167 [cs.LG]. URL: https://arxiv.org/abs/1502.03167.

[14] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. "ImageNet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems*. Ed. by F. Pereira et al. Vol. 25. Curran Associates, Inc., 2012. URL: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[15] LeCun, Y. et al. "Gradient-based learning applied to document recognition". In: *Proceedings of the IEEE* 86.11 (1998), pp. 2278–2324. DOI: 10.1109/5.726791.

[16] Li, Xiang et al. *Generalized Focal Loss: Learning Qualified and Distributed Bounding Boxes for Dense Object Detection*. 2020. arXiv: 2006.04388 [cs.CV]. URL: https://arxiv.org/abs/2006.04388.

[17] Li, Yishuo et al. "Multi-purposed diagnostic system for ovarian endometrioma using CNN and transformer networks in ultrasound". In: *Biomedical Signal Processing and Control* 91 (2024), p. 105923. ISSN: 1746-8094. DOI: https://doi.org/10.1016/j.bspc.2023.105923.

[18] Liu, Wei et al. "SSD: Single Shot MultiBox Detector". In: *Computer Vision – ECCV 2016*. Springer International Publishing, 2016, pp. 21–37. ISBN: 9783319464480. DOI: 10.1007/978-3-319-46448-0_2. URL: http://dx.doi.org/10.1007/978-3-319-46448-0_2.

[19] Podda, Alessandro Sebastian et al. "Multi-scale deep learning ensemble for segmentation of endometriotic lesions". In: *Neural Computing and Applications* 36.24 (Aug. 1, 2024), pp. 14895–14908. ISSN: 1433-3058. DOI: 10.1007/s00521-024-09828-2.

[20] Prince, Simon J.D. *Understanding Deep Learning*. The MIT Press, 2023. URL: http://udlbook.com.

[21] Rahman, Md. Raqibur et al. "CNN-based Deep Learning Approach for Micro-crack Detection of Solar Panels". In: Dec. 18, 2021, pp. 1–6. DOI: 10.1109/STI53101.2021.9732592.

[22] Redmon, Joseph and Farhadi, Ali. *YOLO9000: Better, Faster, Stronger*. 2016. arXiv: 1612.08242 [cs.CV]. URL: https://arxiv.org/abs/1612.08242.

[23] Redmon, Joseph et al. *You Only Look Once: Unified, Real-Time Object Detection*. Version Number: 5. 2015. DOI: 10.48550/ARXIV.1506.02640.

[24] Ren, Shaoqing et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. 2016. arXiv: 1506.01497 [cs.CV]. URL: https://arxiv.org/abs/1506.01497.

[25] Rodrigues, Dinis et al. *Automated Detection of Coronary Artery Stenosis in X-ray Angiography using Deep Neural Networks*. Mar. 4, 2021. DOI: 10.48550/arXiv.2103.02969.

[26] Simonyan, Karen and Zisserman, Andrew. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV]. URL: https://arxiv.org/abs/1409.1556.

[27] Uijlings, J. R. R. et al. "Selective Search for Object Recognition". In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171. DOI: https://doi.org/10.1007/s11263-013-0620-5.

[28] Ultralytics. *Ultralytics YOLO Docs*. URL: https://docs.ultralytics.com/.

[29] Zenggyu. *An Introduction to Evaluation Metrics for Object Detectio*. URL: https://blog.zenggyu.com/posts/en/2018-12-16-an-introduction-to-evaluation-metrics-for-object-detection/#variations-among-the-metrics.