UNIVERSIDADE D
COIMBRA

Pedro José Carrinho Ribeiro

Exploiting Quantization in Convolutional Neural Networks for Polyp Detection on GPUs

Master thesis submitted to the Integrated Master in Electrical and Computer Engineering, specialization in computers, supervised by Professor Gabriel Falcão Paiva Fernandes, and presented to the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Coimbra.

October 2021

Exploiting Quantization in Convolutional Neural Networks for Polyp Detection on GPUs

Pedro José Carrinho Ribeiro

Dissertation submitted to the Department of Electrical and Computer Engineering of the Faculty of Science and Technology of the University of Coimbra to obtain the Master's Degree in

Electrical and Computer Engineering

Supervisor: Gabriel Falcão Paiva Fernandes, PhD

Jury

President:	João Pedro de Almeida Barreto, PhD
Supervisor:	Gabriel Falcão Paiva Fernandes, PhD
Member:	Fernando Manuel dos Santos Perdigão, PhD

October 2021

A bit beyond perception's reach
I sometimes believe I see
That life is two locked boxes, each
Containing the other's key.

- Piet Hien

Agradecimentos

Gostaria, em primeiro lugar, de agradecer ao meu orientador Doutor Gabriel Falcão Paiva Fernandes pela oportunidade que me foi dada de dar um contributo para a comunidade científica através da realização desta dissertação, e por todo o apoio, mentoria, tempo despendido e por ter mostrado uma constante disponibilidade, interesse e compreensão.

Em segundo lugar gostava de agradecer ao Instituto de Telecomunicações de Coimbra pelo material e espaço facultado durante o decorrer da dissertação.

Fico agradecido ao pessoal do Instituto de Telecomunicações de Coimbra, o Oscar Ferraz e em particular ao André Santos pelo tempo e apoio técnico prestado, que foi muito importante em algumas fases da dissertação.

Gostaria também de agradecer ao Professor Doutor Luís Alexandre do Departamento de Informática da Universidade da Beira interior pelo apoio inicial. E também à comunidade do Discord da Darknet pela ajuda prestada em alguns momentos no desenvolvimento da tese.

Finalmente gostaria de agradecer aos meus pais pelas oportunidades e educação que me deram ao longo da vida. E a todos os meus amigos e familiares pela amizade e apoio emocional que me deram ao longo do desenvolvimento da tese e ao longo da vida.

Obrigado a todos vós!

Abstract

Colorectal Cancer (CRC) is one of the most deadly cancers worldwide, with about 900 000 deaths in 2020, and with developed countries having a higher incidence of this type of cancer due to modern dietary habits. The gold standard for colorectal cancer screening is the colonoscopy, with studies concluding that colonoscopies significantly reduce mortality from CRC.

It has been shown in the literature that computer-aided detection (CAD) systems can improve adenoma detection. In particular, Deep Learning models have shown promising results helping physicians during real-time colonoscopies by reducing the number of missed lesions during the colonoscopy. Due to the real-time constraints of colonoscopies, the inference of the frames must happen in real-time. To keep up with the increase in resolution of the colonoscopies and to be able to perform inference in real-time in smaller devices, faster models (i.e., models that are able to process images at a higher frame rate) are required.

In this work we use the You Only Look Once (YOLO)v4 convolutional neural network (CNN) to perform polyp detection. Different methods of regularization, data pre-processing, and data augmentation were tested. To further increase the inference speed of the model to achieve real-time performance and make the model smaller, we deployed the model on NVIDIA TensorRT, which quantizes and optimizes the model. We used the publicly available datasets to train, test, and validate our model to facilitate comparison to other studies. To evaluate the inference speed of the model, a publicly available dataset containing videos was used.

We achieved 82.93% for median avarage precision (mAP), 81.44% for precision, 75.96% for recall, 78.61% for F1 score, 77.00% for F2 score in the publicly available dataset Etis-Larib, and 90.96% for mAP, 88.65% for precision, 87.62% for recall, 88.23% for F1, and 87.86% for F2 using CVC-ClinicDB dataset. An NVIDIA RTX 2080TI graphics processing unit (GPU) was used, and a speed of approximately 98 frames per second (FPS) was achieved on three videos from the Colonoscopic video dataset. For the FP16 version of the implementation, the inference speed was increased to approximately 163 FPS at the cost of a slight decrease in the accuracy metrics. Values of 79.15% in mAP, 75.49% in precision, 74.04% in recall, 74.76% in F1, and 74.72% in F2 was observed for Etis-Larib dataset. For the CVC-ClinicDB, 90.27% in mAP, 88.39% in precision, 86.07% in recall, 87.21% in F1, and 86.52% in F2 were registered. For the INT8 version of the model, the inference speed was further increased to approximately 172 FPS. Values of 80.07% in mAP, 78.68% in precision, 74.52% in recall, 76.54% in F1, and 75.32% in F2 was observed for Etis-Larib dataset. For the CVC-ClinicDB, 90.42% in mAP, 88.34% in precision, 85.60% in

recall, 86.95% in F1, and 86.14% in F2 were registered.

Keywords

Polyp Detection, Real-time, Medical Imaging, Colorectal Cancer, Convolutional Neural Network, CNN, Deep Learning, GPUs, Quantization, YOLO, YOLOv4, TensorRT

Resumo

O cancro colorretal CRC é um dos cancros mais mortíferos à escala mundial, com cerca de 900 000 óbitos registados em 2020. A incidência deste tipo de cancro é maior em países desenvolvidos devido aos hábitos alimentares modernos. A norma de referência para o rastreio do cancro colorretal é a colonoscopia, com estudos a concluir que colonoscopias reduzem significativamente a mortalidade por CRC.

Demonstrou-se na literatura que sistemas CAD podem melhorar a deteção de adenomas. Particularmente, modelos de Aprendizagem Profunda demonstram resultados promissores auxiliando a comunidade médica durante colonoscopias em tempo real, reduzindo o número de lesões não detetadas. Devido à natureza em tempo real das colonoscopias, a inferência das imagens adquiridas tem que ocorrer em tempo real. Para acompanhar o aumento da resolução das colonoscopias, e para possibilitar a inferência em tempo-real em dispositivos mais pequenos, modelos mais rápidos (i.e., modelos capazes de processar imagem com maior taxa de fotogramas) são necessários.

Neste trabalho a CNN YOLOv4 é utilizada para realizar deteção de pólipos. Diferentes métodos de regularização, pré-processamento de dados e *data augmentation* foram testados. De modo a aumentar a velocidade de inferência do modelo para atingir resultados em tempo-real, e para tornar o modelo mais pequeno, este foi lançado no framework NVIDIA TensoRT, que executa quantização e otimização do modelo. Para treino, teste e validação da rede foram usados datasets disponíveis para uso público, facilitando a comparação com outros estudos. Para avaliar a velocidade de inferência do modelo, foi utilizado um dataset público contendo vídeos.

Foram obtidos valores de 82.93% para mAP, 81.44% para precisão, 75.96% para recall, 78.61% para F1, e 77.00% para F2 no dataset de acesso público Etis-Larib, e valores de 90.96% para mAP, 88.65% para precisão, 87.62% para recall, 88.23% para F1, e 87.86% para F2 usando o CVC-ClinicDB dataset. A GPU usada foi a NVIDIA RTX 2080TI, e uma velocidade de aproximadamente 98 FPS foi atingida em três vídeos escolhidos do Colonoscopic dataset. Para a versão em FP16 da implementação, a velocidade de inferência aumentou para aproximadamente 163 FPS à custa de um pequeno decréscimo nas métricas de precisão. Valores de 79.15% em mAP, 75.49% em precisão, 74.04% em recall, 74.76% em F1, e 74.72% em F2 foram observados para o Etis-Larib dataset. Para o CVC-ClinicDB dataset, valores de 90.27% em mAP, 88.39% em precisão, 86.07% em recall, 87.21% em F1, e 86.52% em F2 foram registados. Para a versão INT8 do modelo, a velocidade de inferência aumentou subsequentemente para aproximadamente 172 FPS. Valores de 80.07% em mAP, 78.68% em precisão, 74.52% em recall, 76.54% em F1, e 75.32%

em F2 foram observados para o Etis-Larib dataset. Para o CVC-ClinicDB, valores de 90.42% em mAP, 88.34% em precisão, 85.60% em recall, 86.95% em F1, e 86.14% em F2 foram registados.

Palavras Chave

Deteção de pólipos, Tempo-real, Imagiologia médica, Cancro Colorretal, Convolutional Neural Network, CNN, Deep Learning, GPUs, Quantização, YOLO, YOLOv4, TensorRT

Contents

List of Figures

List of Tables

List of Acronyms

ADR adenoma detection rate

AI artificial intelligence

AUC area under the curve

AP average precision

CAD computer-aided detection

CUDA Compute Unified Device Architecture

MSCOCO Microsoft Common Objects in Context

CIOU complete intersection over union

CNN convolutional neural network

CRC Colorectal Cancer

CBN Cross-Iteration Batch Normalization

CPU central processing unit

CSP Cross stage partial connections

DPM deformable part-based model

DIOU-NMS distance-intersection over union non-maximum suppression

FN false negative

FP false positive

FP32 32-bit floating point

FP16 16-bit floating point

FPS frames per second

GUI graphical user interface

GPU graphics processing unit

GPGPU general purpose graphics processing unit

ILSVRC ImageNet Large Scale Visual Recognition Challenge

IOU intersection over union

INT8 8-bit integer

INT4 4-bit integer

MAC multiply-accumulate

mAP median avarage precision

MiWRC multi-input weighted residual connections

NBI narrow band imaging

NMS non-maximum suppression

NN neural network

R-CNN region based convolutional neural networks

ROC receiver operating characteristic

RPN region proposal network

F-CNN feed-forward Fully-Convolutional Neural Network

PAN Path Aggregation Network

PTQ Post-Training Quantization

QAT Quantization-Aware Training

SAT Self-Adversarial Training

SAM Spatial Attention Module

SPP Spatial Pyramid Pooling

SPPNet Spatial Pyramid Pooling Network

SSD Single Shot Multibox Detector

SVM support vector machines

TN true negative

TP true positive

VOC Visual Object Classes

WCE wireless capsule endoscopy

WL white light

YOLO You Only Look Once

ZSQ Zero-Shot Quantization

1

Introduction

Contents

1.1 Motivation

Colorectal Cancer (CRC) is one of the most deadly cancers worldwide, with about 900 000 deaths estimated in 2020 [9], and developed countries having a higher incidence of this type of cancer due to modern dietary habits [10]. The gold standard for colorectal cancer screening is the colonoscopy, with studies concluding that colonoscopies significantly reduce mortality from CRC [11]. According to the same authors, CRC screening methods should possess high sensitivity and specificity, characteristics that colonoscopies own.

Colonoscopies provide direct visualization of the colon and rectum, allowing a trained physician to look for cancer and detect and remove polyps. Polyps are caused by abnormal cell growth, and even though many of the polyps never become malignant, adenomas may potentially develop into cancer [12]. Usually, CRC takes within 10-15 years to develop [13]. An important metric to assess the quality of the colonoscopy is the adenoma detection rate (ADR), which is the rate at which a physician finds one or more precancerous polyp during a normal screening colonoscopy procedure. It has been shown that a 1% increase in ADR decreases the likelihood of interval CRC (CRC that is developed within 5 years after a colonoscopy procedure) by 3% [14]. The authors of [15] concluded that most of the interval CRCs were due to procedural factors during colonoscopies, especially missed lesions.

The physician's quality and several other factors contribute to achieving higher ADR. Some of these factors are techniques that require training physicians and might have a high learning curve of [15]. It has been shown that computer-aided detection (CAD) systems can improve the ADR metric and require no training whatsoever, with five recent trials published in 2020 supporting this claim [15]. In particular, artificial intelligence (AI) Deep Learning models have shown promising results helping physicians during real-time colonoscopies by reducing the number of missed lesions during the colonoscopy. Several studies support this claim, one of which showed that a deep learning model could detect 26.9% of the missed adenomas during real-time colonoscopies, thus lowering the ADR [16].

Due to the real-time nature of colonoscopies, the inference of the frames must happen in real-time. Edge inference is critical to achieving real-time results, and as in most cases, low-cost inference, low power, small size, and portability are essential features to take into account. To keep up with the increase in resolution of the colonoscopies [17] and to be able to perform inference in real-time in smaller devices, faster models (i.e., models that are able to process images at a higher frame rate) are required.

1.2 Objectives

In this thesis, a polyp detection convolutional neural network (CNN) will be trained and then quantized post-training. This document describes the process chosen for quantizing the network, and results will be evaluated in terms of speed (frames per second (FPS)) and precision. The main objectives of this thesis are:

- Investigate the state-of-the-art for detection problems based on the usage of deep CNNs, polyp detection, and quantization techniques.

- Acquire enough data to create training, testing, and validation datasets capable of assessing the network performance in terms of speed and precision.
- Develop a fully functional CNN capable of preforming polyp detection using a pre-existant CNN architecture;
- Evaluate the results in terms of inference speed and precision;
- Quantize the previous neural network in order to achieve real-time inference;
- Evaluate the results in therms of inference speed and precision;
- Compare the results between the pre-quantized and post-quantized CNN;
- Compare the results obtained with the results in the literature.

1.3 Main contributions

As previously stated, CAD systems can improve the ADR in colonoscopies. A particular advantage of these systems is that no additional training is required for endoscopists to use them, and it provides real-time results during the procedure.

This dissertation offers a CNN implementation of the You Only Look Once (YOLO)v4 algorithm, which to the best of our knowledge, the original ”off-the-shelf” version, has never been used for polyp detection until May of 2021. To train YOLOv4, Darknet was used, and TensorRT was used for quantization and optimization in 16-bit floating point (FP16) and 8-bit integer (INT8) precision. We achieved values of 82.93% for median avarage precision (mAP), 81.44% for precision, 75.96% for recall, 78.61% for F1, 77.00% for F2 in the publicly available dataset Etis-Larib, and values of 90.96% for mAP, 88.65% for precision, 87.62% for recall, 88.23% for F1, and 87.86% for F2 using CVC-ClinicDB dataset. An NVIDIA RTX 2080TI graphics processing unit (GPU) was used, and a speed of approximately 98 FPS was achieved on three videos from the Colonoscopic video dataset.

For the FP16 version of the implementation, the inference speed was increased to approximately 163 FPS at the cost of a slight decrease in the accuracy metrics. Values of 79.15% in mAP, 75.49% in precision, 74.04% in recall, 74.76% in F1, and 74.72% in F2 was observed for Etis-Larib dataset. For the CVC-ClinicDB, values of 90.27% in mAP, 88.39% in precision, 86.07% in recall, 87.21% in F1, and 86.52% in F2 were registered.

For the INT8 version of the model, the inference speed was further increased to approximately 172 FPS. Values of 80.07% in mAP, 78.68% in precision, 74.52% in recall, 76.54% in F1, and 75.32% in F2 was observed for Etis-Larib dataset. For the CVC-ClinicDB, values of 90.42% in mAP, 88.34% in precision, 85.60% in recall, 86.95% in F1, and 86.14% in F2 were registered.

1.4 Dissertation outline

This thesis is structured in 7 chapters. It starts with an introduction highlighting the importance of detecting polyps, followed by a careful state-of-the-art survey regarding polyp detection in chapter 2. Chapter 3 describes the algorithm used to solve the detection problem, the frameworks used to deploy the model, and the quantized version of the model. Next, in chapter 4, we go over the data acquisition process and address the generation of the training, testing, and validation datasets. The next chapter describes the experiences executed and compares the results to other studies. In chapter 6, we go over the process of deploying the model in a quantization framework. The final chapter draws a conclusion from the results and future work goals and possibilities are discussed.

2

State of the art

Contents

This chapter depicts an overview of the state of the art. Firstly, general detection problems are tackled, then detection problems applied to polyps, and finally, quantization techniques.

2.1 Detection problems in images

Because we want to perform polyp detection during colonoscopies, we can think of the video as a series of images put together. Therefore, it is essential to know what solutions exist for generic image detection problems and define the terminology used in this thesis since it may vary throughout the literature.

The main advantage of detection over binary classification is that a rectangular box is outputted around the polyp. This can be helpful in scenarios where the polyp is challenging to spot or if the algorithm outputs a false positive. In the second case, it is easier to know in real-time if the algorithm is wrong since the box will be in the wrong place compared to binary classification, where we may not be sure if it is a wrong classification or a difficult to spot polyp.

In comparison to object segmentation, object detection is not as computationally expensive, thus facilitating real-time performance, which is much needed during colonoscopies.

2.1.1 Object recognition

In this dissertation, object detection will be used to refer to a sub-problem of object recognition, which is the approach used in ImageNet Large Scale Visual Recognition Challenge (ILSVRC) [18].

According to [18], object recognition is used to encompass both image classification problems, where the goal is to identify the image or portion of the image as belonging to a class, and object localization, where the goal is to both classify and identify the location of the object in a given image. Localization problems can be detection problems where the object's location is given by a square bounding box with coordinates or segmentation where a pixel-wise mask gives the object's location. In the literature, the terms localization and detection are often used interchangeably. Figure 2.1 compiles the terminology used in this dissertation.

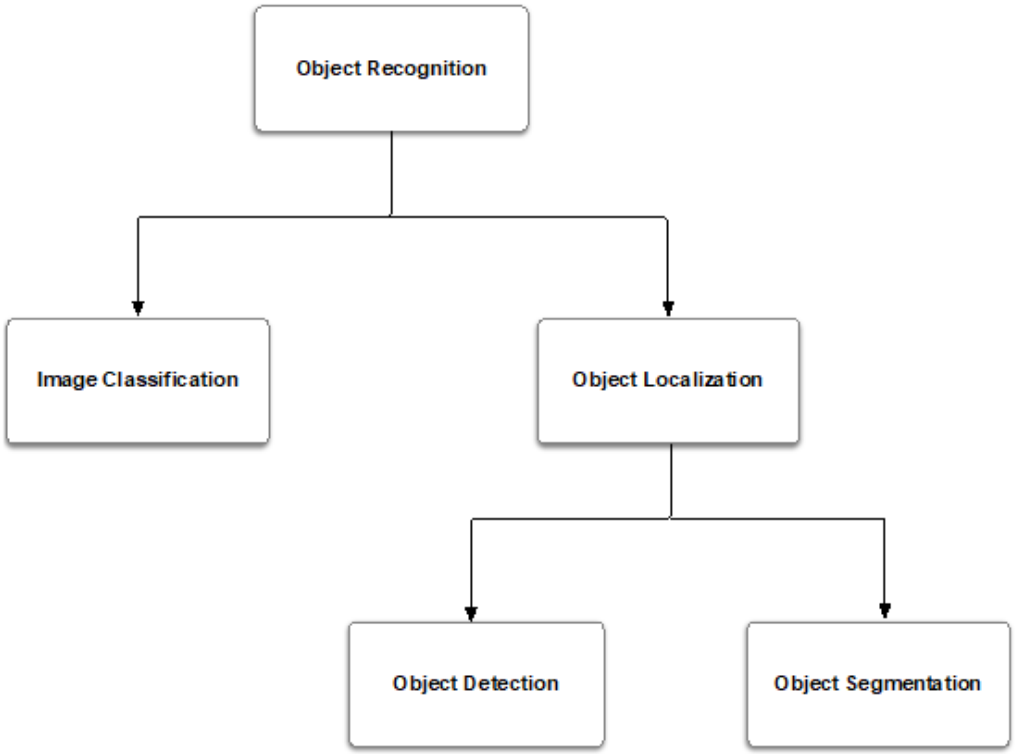

Figure 2.1: Object recognition hierarchy used in this dissertation. Object localization tasks encompass both finding the location of objects and classifying them.

2.1.2 Object detection

Different methods of object detection exist, those that involve a non-neural based approach through the usage of computer vision techniques, which in the literature are often referred to as traditional methods of object detection, and those that use neural network-based techniques where deep convolutional neural networks (CNNs) are employed to solve detection problems [19]. On the neural network-based side, two categories exist, single-stage detector and two-stage detectors. The figure 2.2 synthesizes the methods used to solve object detection problems, as well as giving some examples of current solutions.

Regarding the non-neural-based approaches, three different algorithms must work in conjunction to achieve object detection. First, there is the need for the region proposal algorithm, where the goal is to identify the regions of the image where a potential object could be. Then the feature descriptor is produced and finally, a classifier takes the features that describe the object and outputs a class to that object.

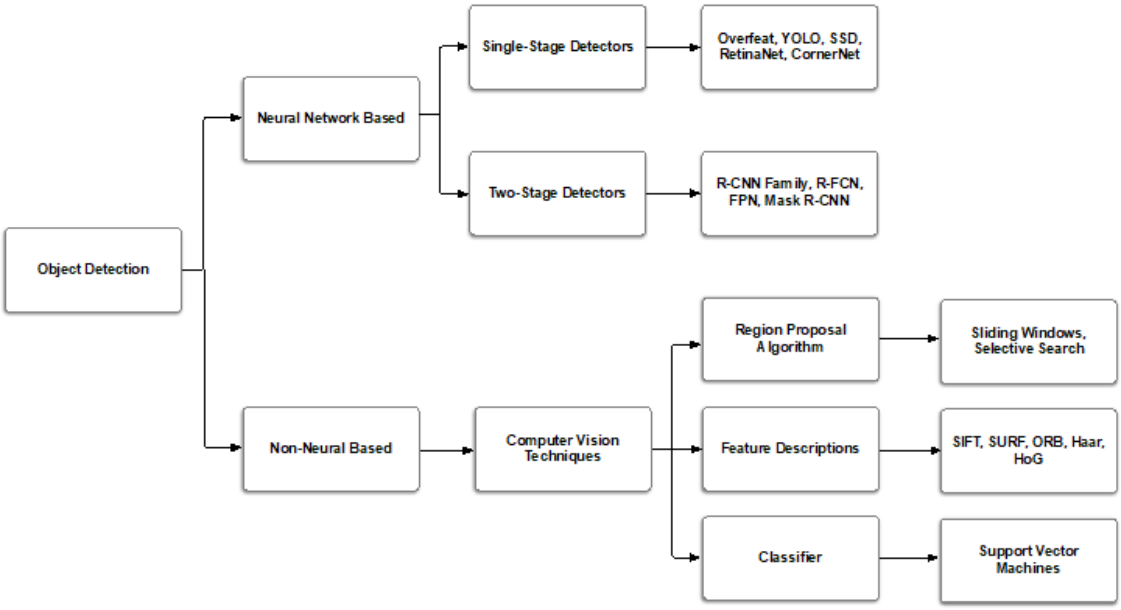

Figure 2.2: Object detection methods and examples.

A few examples for the region proposals are the sliding window where windows with different aspect ratios and sizes are slid across all the pixels of the image, and selective search, which successively aggregates segments of the image based on color, size, texture, and shape similarities until it forms an object [20]. Some object detection CNNs also use these two methods. For the classifier, support vector machines (SVM), AdaBoost and deformable part-based model (DPM) are commonly used [19].

The two-stage detectors work by having a first stage generating the region proposals and a second stage where the feature vectors for these proposals are extracted, and afterward, a class prediction is made.

The generation of region proposals can be achieved using a traditional method such as selective search, which was used in region based convolutional neural networks (R-CNN) and Fast R-CNN [21] or by using a region proposal network (RPN) which is a neural network based on the sliding window algorithm. However, instead of using every single scale and aspect ratio, only a few are used. These are called anchor boxes, and instead of sliding the window through the whole feature map, a downsample is calculated, and finally, a regressor calculates the offset and adjusts the Bounding Box into the correct position. The RPN is used by object detection networks such as Faster R-CNN [22], Mask R-CNN [23] and Spatial Pyramid Pooling Network (SPPNet) [24].

Regarding the single-stage detectors, these are usually faster, simpler, and less accurate than two-stage detectors. In [25], the authors compared the performance of the leading generic object detectors for small objects, which according to [19] is one of the weaknesses of You Only Look Once (YOLO)v1. However, in [25], YOLOv3 [26] was the second best at detecting small objects, in terms of accuracy, only outperformed by Faster R-CNN which is a two-stage detector. In the year of 2020 YOLOv4 [5] was released and it achieved state of the art results both for speed (frames per second (FPS)), and accuracy (average precision (AP)) for frame rates bigger than 65 FPS, out-performing YOLOv3 by 12% and 10% respectively. These

results were obtained using a NVIDIA Tesla V100 graphics processing unit (GPU) on the dataset Microsoft Common Objects in Context (MSCOCO).

2.2 Polyp detection using neural networks

This section aims to elucidate how well deep learning detection models can be applied to the particular field of polyp detection. This section will be heavily based on [7], which is a February 2020 review of the most relevant studies five years prior to the release of the paper. Other studies that were deemed relevant will also be used.

2.2.1 Overview of the most relevant polyp detection studies

In the review, [7], a total of 21 polyp localization and single-class (polyp or no polyp) classification studies using deep learning were analyzed, however in `https://github.com/sing-group/deep-learning-colonoscopy`, a table has been kept updated with the latest studies after the release of the review bringing the total to 29 polyp localization and single-class classification studies (checked in March of 2021), which include 2020 and 2021 studies. Out of these 29 studies, 16 are about polyp detection, with one of them using wireless capsule endoscopy (WCE). From these 16 studies, 7 achieved real-time detection. The table 2.1 was compiled for the studies which were able to perform polyp detection in real-time.

Table 2.1 Table for the most relevant real-time polyp detection studies using deep learning (courtesy of [7]). Regarding the endoscopy type it can either be conventional or WCE.

Study	Date	Endoscopy type	Multiple polyps	Real-time
Tajbakhsh et al. 2014, 2015, [27], [28]	Sept. 2014 Apr. 2015	Conventional	No	Yes
Zheng Y. et al. 2018, [29]	July 2018	Conventional	Yes	Yes
Urban et al. 2018, [30]	Sep. 2018	Conventional	No	Yes
Zhang X. et al. 2019, [31]	March 2019	Conventional	Yes	Yes
Ahmad et al. 2019, [8]	June 2019	Conventional	Yes	Yes
Young Lee J. et al. 2020, [32]	May 2020	Conventional	Yes	Yes
Qadir et al. 2021, [33]	Feb. 2021	Conventional	Yes	Yes

In WCE, also called colon capsule endoscopy, the patient ingests a capsule with a camera. According to [34], the average residence time of a capsule in the small and large intestine is approximately 8 hours, while frames up to 512x512 pixels or more, depending on the capsule model, are transmitted via wireless to the outside of the human body.

2.2.2 Datasets for polyp detection

Most of the studies in table 2.1 used a private dataset. Although multiple public datasets are available, there is still limited data regarding polyps because building medical datasets requires high costs in terms of time and expertise. Only [8] and [33] didn't use a private dataset. The remainder of the studies used a combination of private and public data for training and testing purposes, except for [30] and [31] that used

private datasets only. The table 2.2 was adapted from [7] listing all the public datasets, which studies used them, and some details about them.

Table 2.2 List of publicly available datasets for polyp detection adapted from [7]. CVC-PolypHD and CVC-ClinicVideoDB do not seem to be available anymore. The usage of white light (WL) or narrow band imaging (NBI) is given in the description of each dataset.

Dataset	Description	Format	Resolution	Used in
CVC-ClinicDB [35]	612 sequential WL images with polyps extracted from 31 sequences with 31 different polyps.	Image	388 × 288	[27], [28] [29] [32] [33]
CVC-ColonDB [36]	300 sequential WL images with polyps extracted from 15 videos.	Image	574 × 500	[28] [29] [33]
CVC-EndoSceneStill [37]	912 WL images with polyps extracted from 44 videos (CVC-ClinicDB + CVC-ColonDB).	Image	574 × 500, 388 × 288	-
CVC-PolypHD	56 WL images.	Image	1920 × 1080	-
ETIS-Larib [38]	196 WL images with polyps extracted from 34 sequences with 44 different polyps.	Image	1225 × 966	[29] [8] [33]
Kvasir-SEG [39]	1 000 WL unique images	Image	Various resolutions from 332 × 487 to 1920 × 1072	-
ASU-Mayo Clinic Colonoscopy Video [40]	38 small SD and HD video sequences: 20 training videos annotated with ground truth, and 18 testing videos without ground truth annotations. WL and NBI.	Video	N/A	[8]
CVC-ClinicVideoDB [41]	18 SD videos	Video	768 × 576	[33]
Colonoscopic Dataset [42]	76 short videos (both NBI and WL).	Video	768 × 576	-
PICCOLO [43]	3433 images (2131 WL and 1302 NBI) from 76 lesions from 40 patients.	Image	854 × 480, 1920 × 1080	-

As we can see from table 2.2 the image resolutions from the public datasets ranges from 388 × 288 to 1920 × 1080 pixels.

2.2.3 Methods in the literature where CNNs were adopted

The most important features in each of the seven studies deemed relevant will be highlighted in this subsection.

In Tajbakhsh et al. [28], traditional computer vision techniques were combined with deep learning. In the first stage, the prediction is made using the polyp detection computer vision-based method created in [27]. On a second stage, multi-scale patches around the predicted polyp are collected and fed to a third stage where a CNN corrects the classification of the polyp. Data augmentation techniques were adopted, and no transfer learning was used.

Zheng Y. et al. [29], used YOLO to perform polyp detection. Up to 2 objects are detected by each cell of a 7x7 grid. All images used were resized to 448x448. The initial learning rate was set to 0.0005 and scaled during training. Transfer learning was used with a set of pre-trained weights from PASCAL VOC 2007 and 2012. For training and testing, five trials were used combining multiple datasets, including a private dataset and datasets with data augmentation. The framework used was not mentioned.

Urban et al. [30], used different CNNs designed from scratch, based on pre-existing architectures such as VGG16, VGG19, and ResNet50. Transfer Learning was used in some networks with the initial weights previously trained on the ImageNet dataset. Dropout with a rate of 0.5 to the input of the first and second

fully connected layer was used. The actual values of other hyperparameters were not explicit. For training and testing, a total of 5 datasets were used, including one with colonoscopy videos, one with a combination of polyp images and frames extracted from the videos, and a unique dataset with 11 colonoscopy videos deemed as more challenging. The frameworks used were Keras and Tensorflow.

Zhang X. et al. [31] constructed a CNN from scratch based on Single Shot Multibox Detector (SSD). Data augmentation and transfer learning were adopted using pre-trained weights obtained by training on ILSVRC CLS-LOC. The Learning rate was set to 0.0005 using a multi-step decay policy after iteration 50 000 of 100 000. For training and testing, four trials were conducted, one for each of the CNNs experimented with (all of the SSD family), including the crafted CNN. The framework used was Caffe.

Ahmad et al. [8], used a CNN to perform polyp detection. The methods developed were not described in the paper.

Young Lee J. et al. [32], used YOLOv2 to perform polyp detection. Transfer learning was used with weights that were pre-trained on ImageNet. During the training, images were resized every ten batches to a random resolution. No information was given regarding other data augmentation techniques nor about the hyper-parameters. The framework used was not specified.

Qadir et al. [33], used a feed-forward Fully-Convolutional Neural Network (F-CNN) based model. The F-CNN based models are usually trained with binary masks. However, the model proposed was trained on 2D Gaussian masks to enable the model to perform polyp detection using the masks but outputting bounding boxes at inference time. Data augmentation was used.

These studies use data augmentation techniques to extend the number of images, increase robustness and prevent over-fitting. The following table (2.3) compiling the techniques used was adapted from [7]:

Table 2.3 Data augmentation techniques used in the seven selected papers, adapted from [7]. No data augmentation information was available in Ahmad et al. [8]

Study	Rotation	Flipping	Shearing	Translation (Shifting)	Zooming	Random brightness	Crop	Scale	Resize	Random contrast	Blurring	Color augmentations in HSV	Sharpening
Tajbakhsh et al. 2014, [27], Tajbakhsh et al. 2015, [28]	X			X			X	X	X				
Zheng Y. et al. 2018, [29]	X												
Urban et al. 2018, [30]	X	X	X										
Zhang X. et al. 2019, [31]	X												
Ahmad et al. 2019, [8]	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A
Young Lee J. et al. 2020, [32]						X				X	X		X
Qadir et al. 2021, [33]	X	X			X							X	

Table 2.4 presents extra details regarding the architectures, frameworks, transfer learning, and dropout techniques used in the seven studies deemed relevant.

Table 2.4 Extra details regarding the seven selected studies. Adapted from [7].

Study	Architecture	Framework	Transfer Learning	Dropout
Tajbakhsh et al. 2014, [27]Tajbakhsh et al. 2015, [28]	Costum architecture	-	ImageNet	-
Zheng Y. et al. 2018, [29]	YOLOv1	-	PASCAL VOC 2007 and 2012	-
Urban et al. 2018, [30]	ResNet-50, VGG16 and VGG19	Keras	ImageNet Also without any TL	Yes
Zhang X. et al. 2019, [31]	Costum architecture based on SSD	Caffe	ILSVRC CLS-LOC	No
Ahmad et al. 2019, [8]	-	-	-	-
Young Lee J. et al. 2020, [32]	YOLOv2	-	ImageNet	-
Qadir et al. 2021, [33]	Resnet34 and MDeNet(F-CNN based)	-	ImageNet (for the Resnet34 encoder)	-

2.2.4 Experiments and results in the literature

This subsection addresses the way experiments were conducted and the results obtained in the seven selected studies. The most important metrics to determine the success of the CNN in polyp detection context are precision and recall. The recall metric is very important since it is desired that the CNN miss very few polyps (low false negative count), even at the expense of having a higher value of the false positive count, thus decreasing precision. The precision-recall trade-off, as well as specificity, are discussed in chapter 5. Other metrics are also used across the different studies. The metrics used are further developed in chapter 5.

Table 2.5 contains the results obtained across the seven studies.

Table 2.5 Results obtained from the seven studies deemed as relevant for the dissertation. The values of the metrics for a specific dataset on a given study are shown one per line. If nothing is said after the value, it means the dataset is private. Public datasets have been identified after the value of each metric. The table is courtesy of [7].

Study	Recall (sensitivity)	Precision	Specificity	Other
Tajbakhsh et al. 2015, [28]	70%	63%	90%	F1: 0.66, F2: 0.68
Zheng Y. et al. 2018, [29]	74% on ETIS-Larib	77.4% on ETIS-Larib	N/A	F1: 0.757, F2: 0.747 on ETIS-Larib
Urban et al. 2018, [30]	93% 100% 93%	74% 35% 60%	93% (on the first dataset)	F1: 0.82, F2: 0.88 F1: 0.52, F2: 0.73 F1: 0.73, F2: 0.84
Zhang X. et al. 2019, [31]	76.37%	93.92%	N/A	F1: 0.84, F2: 0.79
Ahmad et al. 2019, [8]	91.6% on ETIS-Larib 84.5%	75.3% on ETIS-Larib	92.5%	F1: 0.83, F2: 0.88 on ETIS-Larib
Young Lee J. et al. 2020, [32]	96.7% 90.2% on CVC-ClinicDB	97.4% 98.2 on CVC-ClinicDB	N/A	F1: 0.97, F2: 0.97 F1: 0.94, F2: 0.96 on CVC-ClinicDB
Qadir et al. 2021, [33]	86.54% on ETIS-Larib 91% CVC-ColonDB	86.12% on ETIS-Larib 88.35% on CVC-ColonDB	N/A	F1: 0.863, F2: 0.864 on ETIS-Larib F1: 0.896, F2: 0.904 on CVC-ColonDB

The table 2.6 was compiled with information regarding the number of images on the dataset, the GPU used, and the results for inference speed.

Table 2.6 Summarization of the most critical aspects of the experiments performed in the seven studies, namely the number of images in each dataset and how the splits between training and testing were performed, the graphics card used, and the inference speed that the model was used able to achieve in the testing datasets.

Study	Number of frames datasets	GPU used	Inference speed	Comments
Tajbakhsh et al. 2015, [28]	7 000 frames with polyps, 28 000 without obtained from 40 videos. The frames were split into training and testing datasets.	-	-	A new performance curve to measure the latency between the appearance of the polyp in the video and its detection was proposed.
Zheng Y. et al. 2018, [29]	A total of 4766 images, including augmented images, was used for training split across 5 trials. A total of 196 images was used for testing.	-	16.67FPS	Trial 1: 612 images from CVC-ClinicDB Trial 2: 798 images from the private dataset,PWH-ColonDB. Trial 3: The images from CVC-ClinicDB were augmented to 2448 images. Trial 4: Trial 3 was combined with a data augmentation version of CVC-ColonDB, bringing the total to 3968 images. Trial 5: Trial 4 was combined with PWH-ColonDB, bringing the total to 4766 images. For testing, the dataset used in all of the trials was ETIS-Larib. It was observed that datasets that contained the most images led to better results, proving that the amount of data is the main bottleneck.
Urban et al. 2018, [30]	A private dataset with 8641 images, where 4 088 of those contained polyps. Two video datasets containing videos were created, one with 9 videos and another with 11 videos deemed as more challenging.	NVIDIA TITAN X	100FPS	Three main experiments were conducted: Cross-validation on the 8641 image dataset using the k-fold method with k = 7. Training/Testing using the 8641 images as training dataset and the 9+11 videos as testing dataset. Training/Testing using the 8641 images and the 9 videos as training dataset, and the 11 videos as testing dataset.
Zhang X. et al. 2019, [31]	A private training dataset of 354 polyp images was augmented to 708. A testing dataset with 50 polyp images. Another testing dataset with 171 polyp images.	NVIDIA TITAN V	SSD-GPNet: 50 FPS SSD: 62 FPS	Two models were compared SSD and SSD-GPNet. For training purposes, the PASCAL VOC2007 and VOC2012 datasets (general-purpose datasets), were extended by including "polyp" as class No. 21.
Ahmad et al. 2019, [8]	The 4 664 polyp image, MICCAI 2015 polyp challenge dataset, was used for training. A private testing dataset of 83 716 images with 14 634 with polyps and 69,082 without was used. The 196 polyp image, MICCAI 2015 polyp challenge test set, was used as testing dataset.	-	-	
Young Lee J. et al. 2020, [32]	Two training datasets were combined with a total of 8 075 + 420 polyp images. A private testing dataset with 1338 polyp images. CVC-ClinicDB with 612 polyp images was used as testing dataset. A dataset of 7 videos for testing. A dataset of 15 unaltered videos was used for real-world validation.	NVIDIA GeForce GTX 1080	67.16 FPS	The 7 video private dataset was used to perform sensitivity and specificity analysis. The true positive, false positive, true negative and false negative counts for the real-world validation dataset were obtained by having 3 expert endoscopists recheck every frame of the predictions made by the algorithm.
Qadir et al. 2021, [33]	CVC-ClinicDB with 612 polyp images was used as training dataset. ETIS-LARIB (196 polyp images) and CVC-ColonDB (300 polyp images) were used as testing datasets.	NVIDIA GeForce GTX 1080 Ti	EncDec: 35.71FPS MDeNetplus: 25.64FPS	A k-fold cross-validation method with 5 folds was used to train the model and choose the hyper-parameters. Two testing phases were conducted, one for each testing dataset.

2.2.5 A study conducted using YOLOv4

Recently as of May 2021, a study using YOLOv4 and YOLOv3 for polyp detection was released [44]. However, in this study, the authors used the "off-the-shelf" version of YOLOv4 and proposed different modifications to enhance the model. These modifications included changing the activation functions used, changes to the backbone CSPDarknet-53, and testing different loss functions.

This study used Etis-larib and CVC-ColonDB datasets for testing purposes, while CVC-ClinicDB was used for training. For hyper-parameter optimization, a genetic algorithm was used, customized anchor box priors were used, and a pre-processing step to the images was taken in which the frames that had small polyps were copy-pasted to balance out small and medium-large polyps, and the small polyps were zoomed in while the medium and small polyps were zoomed out. In regards to data augmentation, they used: mosaic, shear, scale, cutmix, and contrast to the dataset before training, and mosaic, cutmix, shear, scale, flip, mix-up, rotate, crop, saturation, hue, color value change, brightness, translation, exposure, blur, and random noise during training.

The authors also applied quantization to the different models tested, deploying them in TensorRT using 16-bit floating point (FP16) precision. However, 8-bit integer (INT8) precision wasn't used.

The GPU used was a single NVIDIA RTX-2080TI. One significant result is for the off-the-shelf version of YOLOv4. This model, on Etis-Larib, obtained results of 81.78% for precision, 79.25% for recall, and 80.50% for F1 score while using a value of 0.5 for intersection over union (IOU) threshold, and values ranging from 0.25 to 0.3 for confidence threshold.

2.3 Quantization of CNNs

Real-time inference is a critical aspect to consider when solving a polyp detection problem since colonoscopies are real-time procedures. Usually, we do not have unlimited resources or a cluster of data centers GPUs at our disposal, and we have to resort to other options to increase the inference speed. These options result in a trade-off between accuracy and speed.

The efforts in making CNNs more efficient can be categorized. We followed the taxonomy proposed in [45] and [46], and the following categories are highlighted:

- Efficient CNN architecture design: using certain modules, kernel functions, or low-rank factorization that result in a more efficient CNN (e.g., using a certain kernel-type that makes convolution more efficient in a specific piece of hardware, layer fusion);
- Network architecture search: searches for a highly efficient structure from a large predefined space;
- Knowledge distillation: using a pre-trained large model as a teacher to train a smaller model.
- Co-Designing CNN architecture and hardware together: adapting the network for a specific hardware platform;
- Pruning: removing neurons, convolutional filter, or even layers from the network that minimally affects the model output;
- Quantization: lowering the precision of the weights and activations in order to increase the network's efficiency.

For CNN acceleration, a combination of the previous methods as well as hardware accelerators are used [45].

The current section (section 2.3) is heavily based on three surveys regarding quantization [45], [46], and [47].

2.3.1 What quantization of CNNs represents

Quantization is a method to map from a large set of input values to a smaller set of output values. Quantization is related to areas such as calculus, signal processing, and more recently neural networks (NNs). NNs are good candidates for quantization since inference and training are computationally intensive procedures and CNN models are usually over-parameterized [46]. Unlike in signal processing and calculus, we are not interested in minimizing the error between the input and output values of the quantizer. Instead, we want to optimize a forward error metric based on the quality of the task the model is trying to solve. We can have a quantized model whose values are much different from the values of the original model but with a similar forward error metric.

CNNs are usually trained using 32-bit floating point (FP32) weights and activations. The multiply-accumulate (MAC) operations and data transfers are the most prevalent operations in the processing unit. The goal of quantization is to reduce the precision of the weights and activations, reducing the amount of data transfer and the size and energy of the MAC operations [47]. Usually, FP16, INT8, 4-bit integer (INT4), or 1-bit precision are used to store the weights and activations in quantization. However, for precision lower than INT8, it usually requires retraining the network.

2.3.2 Uniform quantization

In uniform quantization, the resulting quantization levels are uniformly spaced, which is not the case in non-uniform quantization. Non-uniform quantization schemes are quite difficult to deploy on GPUs and central processing units (CPUs), which leads to uniform quantization being the most used method of quantization [46]. For this reason non-uniform quantization will not be discussed.

A simple way to define uniform quantization is using the following expression:

$$Q(x) = round\left(\frac{x}{S}\right) + Z \tag{2.1}$$

$Q(x)$ represents the quantized version of x, S represents the scale, and Z is the zero-point. Multiple expressions could be used to define $Q(x)$ depending on the method we might want to use.

The de-quantization function can be defined as:

$$\tilde{x} = S\,(Q(x) + Z) \tag{2.2}$$

2.3.3 Symmetric and asymmetric quantization

For asymmetric quantization, the zero-point Z is a value different from zero, and in the case of symmetric quantization, z is equal to zero. The scaling factor S can be defined as:

$$S = \frac{\beta - \alpha}{2^b - 1} \tag{2.3}$$

Where $[\alpha, \beta]$ is the clipping range (in symmetric quantization $-\alpha = \beta$) and b is the quantization bit width. The process of selecting the clipping range is also called calibration. Asymmetric quantization can be particularly important in cases where the values of the weights or activations are imbalanced, for example, quantizing an activation after a ReLU layer, which has non-symmetric values [46].

2.3.4 Selection of the calibration range

When choosing the calibration range, a few choices exist. A popular method is to use the minimum and maximum values of x, or in the case of symmetric quantization, the maximum of the modules between the minimum and the maximum values of x, percentiles can also be used, and KL divergence minimization can also be employed.

The algorithms for choosing the calibration range can be static or dynamic. The weights are frozen during inference so that the calibration range can be calculated statically. This is not the case for activations whose values change during each pass, and real-time computation of the statistics can be used to calculate the range (dynamically). In the case of the activations, the range can also be computed statically by running a series of calibration inputs, using other metrics such as cross-entropy, mean square error, using the parameters of batch normalization, or learning the best range during training [47].

When applying the previously mentioned methods for clipping the range, one must consider the granularity of the quantization. The quantization could be applied layerwise (the whole layer is clipped), groupwise (grouping channels inside a layer and clip them), channelwise (clipping channel by channel), and sub-channelwise (clipping a group of weights inside a layer).

2.3.5 Post-Training Quantization and Quantization-Aware Training

Quantization-Aware Training (QAT) can be used to eliminate the perturbation introduced by quantization by re-training the model. One approach is to do the forward pass and backpropagate on the quantized model (with the quantized weights and activations) but using FP32. Then re-quantize the model after each gradient update resulting in quantized parameters.

The gradient of the quantization function (equation 2.1), is zero almost everywhere. Therefore different approaches exist to approximate the gradient calculation. QAT will be no further discussed since it will not be used in this work.

Post-Training Quantization (PTQ) has a few advantages over QAT: the overhead introduced is very low, and it can be applied in situations where limited training data or lack of labeled data exist since it requires no re-training. On the other hand, the accuracy degradation is higher in PTQ than it is in QAT.

Several methods have been proposed to diminish the accuracy degradation. Studies observed bias in the mean and variance of the quantized weights and activations of a network, and other studies show that equalizing the weight ranges can reduce the bias [46]. To minimize the accuracy degradation several methods

exist:

- Calculating the optimal clipping range analytically [48];
- Optimizing the L2 distance between the quantized tensors and the corresponding FP32 tensors [49];
- Duplicating channels that contain outliers and then halving them [50];
- Using AdaRound, which allows for an adaptive rounding function [50];
- Using AdaQuant, which minimizes the error between the output of the quantized layer and the output of the full-precision layer [51].

Other methods exist, however, these were the ones highlighted in [46]. Although not as important to our work, Zero-Shot Quantization (ZSQ) [46] aims to perform quantization without any access to training and validation data, which is particularly important to Machine Learning as a Service due to privacy and security reasons or in scenarios where training data is scarce.

2.3.6 Quantization bellow 8 bits

For quantization below 8 bits, it is usually required to perform QAT [47] as well as to deploy advanced quantization techniques such as simulated quantization, mixed-precision quantization, hardware aware quantization, and distillation-assisted quantization [46].

Another line of work explores extremely low precision, such as binary quantization and ternary quantization [46] [45].

Usually, PTQ techniques are the first go-to tool as they are fast to implement [47]. In this thesis, quantization below 8-bits is not performed because in order to achieve real-time inference on general purpose graphics processing units (GPGPUs) going bellow 8-bits might not be necessary since in the work of [44] TensorRT with FP16 precision was used to solve a polyp detection problem achieving real-time results on an NVIDIA RTX 2080TI. In the work of [52], an analysis was conducted to the speedup using quantization, pruning, and tensor decomposition on GPUs and other devices, achieving a speedup of 3.4 to 7.2$\times$ on GPU when comparing different CNN models before and after optimization.

2.4 Summary

This chapter discusses, the state of the art. Firstly, detection problems in a general context were approached, then detection problems in the specific context of polyp detection were analyzed, particularly the most relevant studies using deep learning capable of reaching real-time inference. Details such as the datasets used, results, and how the experiments were conducted were included. Finally, the processes of quantizing a CNN was also addressed.

The work of Young Lee J. et al. 2020 [32], Zheng Y. et al. 2018 [53], and Pacal I. et al. [44] are particularly important since in this thesis YOLOv4 is going to be used, and their work used YOLOv2,

YOLOv1, and YOLOv4 respectively. Pacal et al. used several modified versions of YOLOv4. Next, in chapter 3, YOLOv4 [5], the framework Darknet [54], and TensorRT are going to be discussed.

3

Background on YOLO neural networks

Contents

This chapter depicts an overview of the algorithm and frameworks chosen to solve the problem of polyp detection. A comprehensive explanation of the architecture chosen and the reasons for such choice are given. A short overview of the implementation and frameworks used both for training/testing and quantization are also highlighted. In this chapter, metrics relative to the algorithms will be highlighted. Please refer to section 5.1 where we formally define these metrics.

3.1 Object detection Network: YOLOv4

As previously stated, using deep learning, we have three ways to solve the problem of polyp recognition: we can use a classifier to classify every single frame during the colonoscopy, use object detection, or use object segmentation. Due to the real-time nature of colonoscopies, using object segmentation would be computationally expensive. Object Detection outputs a rectangular box around the polyp, making it much easier for medical professionals to spot false positives than it would be with image classification. On top of that, there can be multiple polyps in the same frame, and outputting one box per polyp instead of simply stating if the fame contains polyps or not is superior since a single polyp miss by the medical professionals can lead to negative consequences to the patient.

On the You Only Look Once (YOLO)v4 paper [5] a comparison of frame rate and precision was made across the most relevant state-of-the-art detectors on Microsoft Common Objects in Context (MSCOCO) dataset [55], and YOLOv4 is superior to all of them both in precision and frame rate. The only exception being EfficientDet [56] which can obtain a better average precision (AP) [1] at 30 frames per second (FPS) (approx. 44.5% AP) while YOLOv4 achieves 43.5% AP but at more than the double of the frame rate. YOLOv4 seems to be much superior to the other state-of-the-art detectors for real-time inference, and for this reason, this was the architecture we decided to adopt. After the release of YOLOv4 other versions of YOLOv4 such as PP-YOLO [57] and Scaled-YOLOv4 [58] were released which led to even better results. However, we still decided to go for the vanilla version.

Currently, there are four main official versions of YOLO each with its paper. All of them are discussed in this subsection since the later versions are built on top of knowledge from previous versions.

3.1.1 Overview of YOLOv1

This subsection is based on YOLOv1 [1], May 2016. This model was the first iteration of the YOLO algorithm, and it is essential to understand the process behind the algorithm, the unique loss function, and the initial architecture where the next iterations were built on top of.

3.1.1.A Key features of YOLOv1

YOLOv1 is a single-stage detector that treats the problem as a single regression problem. It outputs rectangular Bounding Boxes surrounding the objects. YOLOv1 calculates 98 bounding boxes per image

[1] Please refer to section 5.1, where the evaluation metrics are formally defined

while being extremely fast (45 FPS on an NVIDIA Titan X GPU), thus being capable of running in real-time.

Despite being extremely fast compared to other methods at the date, such as the two-stage detector Fast R-CNN [21], it was not as accurate. However, YOLOv1 still made fewer background errors (classifying background as object) because it looks at the image as a whole instead of using a region-proposal-based technique.

3.1.1.B Architecture and main ideas

The detection network is composed of 24 convolutional layers followed by two fully connected layers. After every convolution, a leaky ReLU activation function is used, and following every convolutional layer, there is a Maxpool Layer except in the last convolutional layer following as it shows in figure 3.1.

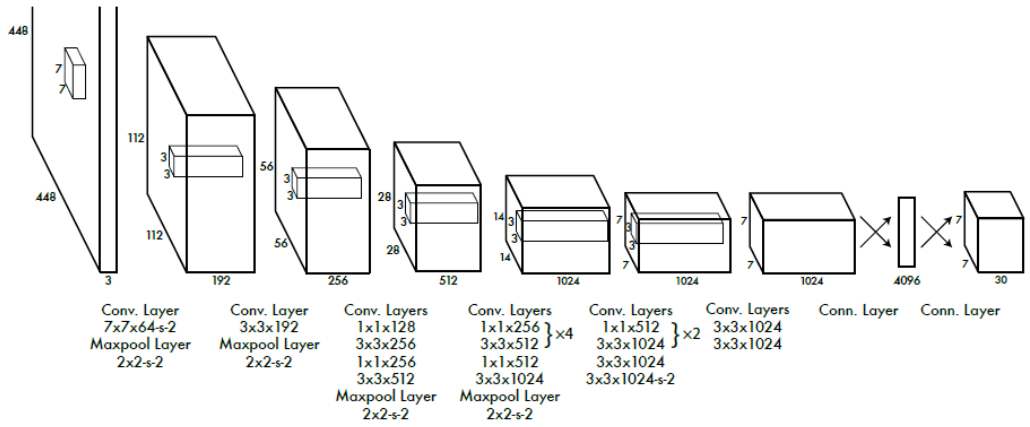

Figure 3.1: YOLOv1 architecture, courtesy of [1]

As we can see from the figure, the input layer is a 448×448 [2] resized version of the original image with three channels. The feature maps size gets resized to a lower dimension as we get deeper into the network, culminating in a $7 \times 7 \times 1024$ tensor fed to two fully connected layers after suffering a flattening operation. On the first fully connected layer, a linear transformation is applied to the $7 * 7 * 1024$ flattened layer connecting it to another layer with 4096 neurons, after, there is a dropout layer with a rate of 50%, followed by a Leaky ReLU activation function. Finally, another linear transformation is applied, generating the $7 \times 7 \times 30$ tensor at the output layer. Data augmentation such as random scaling, translation, random exposure, and saturation are used.

At the output layer we get a $S \times S \times (C + B * 5)$ tensor. This is because the network divides the input image into a $S \times S$ grid cell, where each cell is associated with five coordinates for B bounding boxes. Those coordinates are the center of the box (x,y), the width of the box (w), the height of the box (h), and the confidence score of the box (how likely is a box from a given cell to contain an object). Each cell also

[2] In this chapter, we follow the notation used in the YOLO papers, where $\times$ is used for multi-dimension, and $*$ is used for single-dimension

predicts C conditional probabilities for classification. During inference, one image is fed to the network at a time. However during training, the number of images fed simultaneously depends on the batch size. Figure 3.2 illustrates the output of the network.

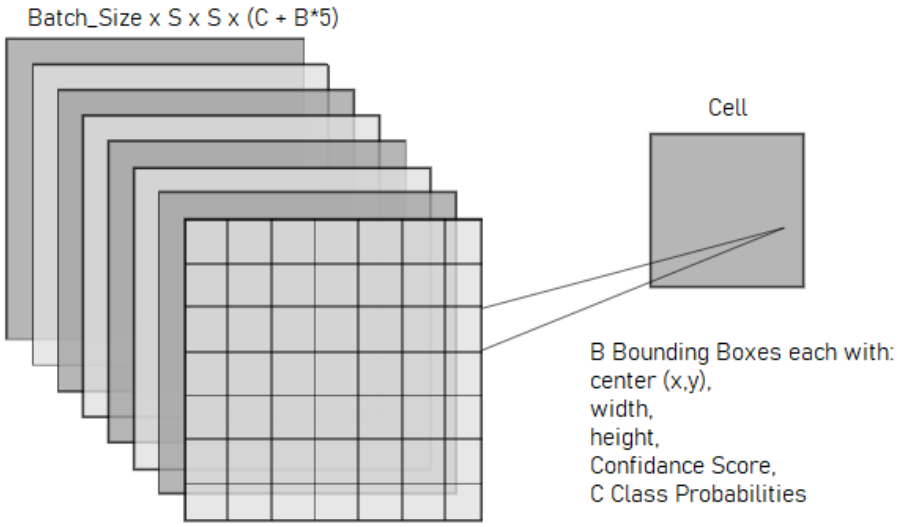

Figure 3.2: Output of the YOLO network

In the paper the authors used S = 7, B = 2, C = 20 (because PASCAL Visual Object Classes (VOC) has 20 different classes).

To calculate the confidence score, i.e., the probability of one of the B boxes in a given cell to contain an object, we used the following expression, where the probability of existing an object is multiplied by the intersection over union between the Ground Truth Box and the predicted Box:

$$C_i = P(Object) \times IOU_{prediction}^{GroundTruth} \tag{3.1}$$

The intersection over union (IOU) is calculated by dividing the area originated by the interception of two boxes by the area of the union of those same boxes:

$$IOU_A^B = \frac{Area(A \cap B)}{Area(A \cup B)} \tag{3.2}$$

To calculate the class probability for a specific class, we multiply the Confidence score by the conditional class probability:

$$P(Class_i|Object) \times P(Object) \times IOU_{prediction}^{GroundTruth} = P(Class_i) \times IOU_{prediction}^{GroundTruth} \tag{3.3}$$

3.1.1.C Loss function, training and inference of YOLOv1

During training, the network uses a loss function to compute and backpropagate the error using stochastic gradient descent. The labeled data comprises images and text files containing the class label and box coordinates for each object in the image. The YOLO loss function is the following:

$$\begin{aligned}
&\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i,j}^{obj}\left[(x_i-\hat{x}_i)^2+(y_i-\hat{y}_i)^2\right] \\
&\quad+\lambda_{coord}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i,j}^{obj}\left[\left(\sqrt{w_i}-\sqrt{\hat{w}_i}\right)^2+\left(\sqrt{h_i}-\sqrt{\hat{h}_i}\right)^2\right] \\
&\quad+\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i,j}^{obj}\left(C_i-\hat{C}_i\right)^2 \\
&\quad+\lambda_{noobj}\sum_{i=0}^{S^2}\sum_{j=0}^{B}\mathbb{1}_{i,j}^{noobj}\left(C_i-\hat{C}_i\right)^2 \\
&\quad+\sum_{i=0}^{S^2}\mathbb{1}_{i}^{obj}\sum_{c=1}^{classes}\left(p_i(c)-\hat{p}_i(c)\right)^2
\end{aligned} \tag{3.4}$$

One key aspect of the loss function is that we consider a cell responsible for the prediction if the center of the object is on that given cell. Therefore during training, we must differentiate somehow if the cell is supposed to contain an object or not. We do not want to backpropagate bounding box coordinate values associated with cells that are not predicting objects. The identity function accomplishes this idea:

$$\begin{aligned}
\mathbb{1}_{i,j}^{obj} &= \begin{cases} 1, \text{if the box j of cell i is responsible for the object} \\ 0, otherwise \end{cases} \\
\mathbb{1}_{i,j}^{noobj} &= \begin{cases} 1, \text{if the box j of cell i is not responsible for the object} \\ 0, otherwise \end{cases} \\
\mathbb{1}_{i}^{obj} &= \begin{cases} 1, \text{if the cell i is responsible for the object} \\ 0, otherwise \end{cases}
\end{aligned} \tag{3.5}$$

To find the results for the identity function during training, we must check the ground truth values to know which cells contain objects, and select the highest IOU between the B boxes on cell i and the target value.

The first two parcels of the loss function refer to the object loss. The parameter λ_{coord} increases the penalization of boxes containing objects since more cells containing background exist than cells containing foreground. We also take the square root of the width and height so that the model does not weigh large boxes and small boxes equally since a small error in a small box is not equal to the same error on a large box. The third and fourth parcels are responsible for calculating the square error of the confidence score (equation 3.1). Finally, the last parcel calculates the square error of the class probabilities.

During inference, 98 bounding boxes are predicted, and the best ones are selected based on non-maximum suppression. The authors pretrained the network on ImageNet [18], using the first 20 convolutional layers followed by an average-pooling layer and a fully connected layer, and using a resolution of 224×224 at the input layer. For detection training Pascal VOC 2007 [59] and 2012 [60] datasets were used.

3.1.2 Overview of YOLOv2

This subsection will be based on the YOLOv2 paper [2] of December 2016. This paper corresponds to the second iteration of the YOLO algorithm. The novels on this iteration are:

- Improvements made on the model using novel techniques and techniques from other past works.

- A dataset combination method which allows for classification data to be used to expand the scope of detection systems.

- A joint train method that allows training both in detection data and classification data.

The dataset combination and join training portions will not be discussed since the datasets used in this dissertation are detection-only datasets. However, some polyp classification datasets contain classes that differentiate between types of polyps, which YOLO could make use of if we wanted to.

3.1.2.A New techniques used on YOLOv2

YOLOv2 uses batch normalization, which improves the model convergence and eliminates the need for other regularization techniques, such as removing Dropout without causing overfit. Batch normalization applies normalization in the chosen layers, solving the problem of covariate shift and having a slight regularization, thus accelerating training [61].

The fully connected layers for bounding boxes predictions were removed and replaced by the usage of anchor boxes as priors and then calculating the offset of the coordinates as in Faster-region based convolutional neural networks (R-CNN) [22]. On YOLOv2, the class prediction was decoupled from the spatial location (cell) so that class prediction is made once per bounding box per cell instead of once per cell. We believe this is because now we have different priors, and we want to specialize each of these priors in detecting objects with specific characteristics. For example, we might have calculated some priors that are more well suited to detect horizontal objects or small objects depending on the shape and size of the prior we calculated.

To generate the priors (anchor boxes), k-mean clustering [62] was used on the training dataset. Using k = 5 led to good enough results [2]. The distance metric used was:

$$d(box, centroid) = 1 - IOU(box, centroid) \tag{3.6}$$

The model can now predict more than 1000 boxes, losing only 0.3% median avarage precision (mAP) and improving recall by 7% [2].

To avoid model instability, especially during early iteration YOLOv2, similarly to YOLOv1 predicts the bounding boxes at the cell level instead of using the whole image. The predictions made are $t_x, t_y t_w, t_h$ and t_0, respectively being the center (x,y) of the bounding box relative to the cell, the offsets for the width and height of the priors, and the confidence score. The following set of formulas are then applied using the

values predicted to obtain the coordinates of the bounding box (b_x, b_y, b_w, b_h):

$$\begin{aligned} b_x &= \sigma(t_x) + c_x \\ b_y &= \sigma(t_y) + c_y \\ b_w &= p_w e^{t_w} \\ b_h &= p_h e^{t_h} \\ P(Object) \times IOU_{prediction}^{GroundTruth} &= \sigma(t_0) \end{aligned} \tag{3.7}$$

Where $\sigma()$ is the sigmoid function, c_x and c_y represent the cell number in terms of rows and columns, and p_w and p_h are the width and height of the box priors. The figure 3.3 illustrates the previously stated:

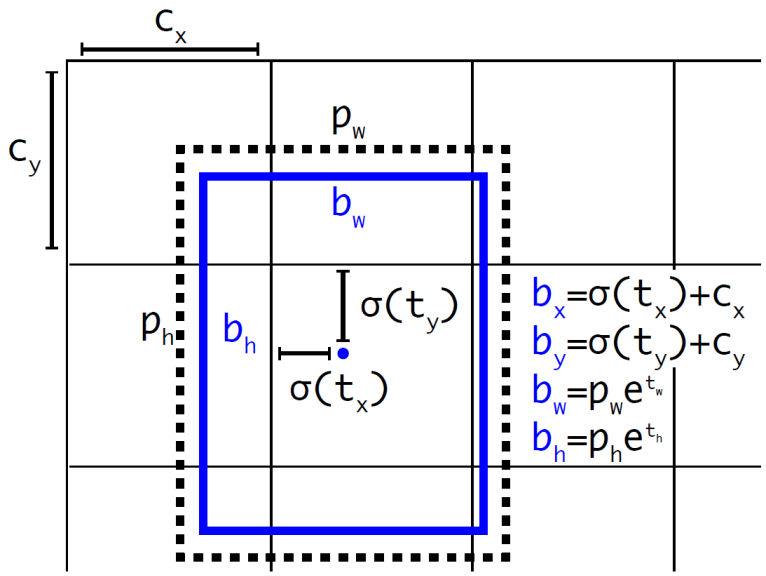

Figure 3.3: Bounding box prediction, courtesy of [2]. We predict the height and width as offsets to the priors and the center coordinates relative to the cell using a sigmoid function to set the value between 0 and 1.

Because the model no longer has fully connected layers, multi-scale training is possible, making the model more robust to different resolution images on inference, and since smaller images take less time to compute, a trade-off between accuracy and speed is now possible.

3.1.2.B Loss function, training and inference of YOLOv2

The model is pre-trained for classification on ImageNet with 1000 classes at 224×224 and then fine-tuned again on ImageNet at 448×448 for 10 epochs. This training phase lasts 160 epochs, and data augmentation is used. The backbone used, Darknet-19, is as follows on table 3.1:

Table 3.1 Darknet-19: The backbone for YOLOv2. The last three layers are added to train for classification on ImageNet. On the filter size column the letter "s" designates the stride of the the operation.

Type	Filters	Filter size	Output
Convolutional	32	3×3	224×224
Maxpool		2×2s2	112×112
Convolutional	64	3×3	56×56
Maxpool		2×2s2	56×56
Convolutional	128	3×3	56×56
Convolutional	64	1×1	56×56
Convolutional	128	3×3	28×28
Maxpool		2×2s2	28×28
Convolutional	256	3×3	28×28
Convolutional	128	1×1	28×28
Convolutional	256	3×3	14×14
Maxpool		2×2s2	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Convolutional	256	1×1	14×14
Convolutional	512	3×3	14×14
Maxpool		2×2s2	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	512	1×1	7×7
Convolutional	1024	3×3	7×7
Convolutional	1000	1×1	7×7
Avgpool		Global	1000
Softmax			

To train for detection model is modified. The input layer becomes $416 \times 416 \times 3$, a passthrough layer is added, the last convolutional layer is removed, three $3 \times 3 \times 1024$ convolutional layers are added, followed by a last convolutional layer, which is going to give us the final output layer of $13 \times 13 \times (5 * 25)$. The detection training phase also lasts 160 epochs, and data augmentation is used. The datasets used for detection training are the MSCOCO dataset and PASCAL VOC 2007 and 2012.

The passthrough layer added for detection purposes, allows to bring a higher resolution feature map that is then concatenated with the low-resolution feature map. According to the yolov2.cfg implemented on Darknet (`https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov2.cfg`) we can see that layer 16 is routed to the input of layer 26 and then reorganized, to be concatenated with layer 25, to be then routed to layer 27, thus implementing the passthrough layer.

It is worth noting that the loss function for YOLOv2 has been slightly changed. Now the class predictions are made at the box level instead of the cell level. Therefore, the third identity function in equation 3.5 no longer exists, and the math from equation 3.7 must be used before calculating the loss, thus changes

were made to the identity function:

$$\mathbb{1}_{i,j}^{obj} = \begin{cases} 1, \text{if the box j of cell i is responsible for the object} \\ 0, otherwise \end{cases}$$
$$\mathbb{1}_{i,j}^{noobj} = \begin{cases} 1, \text{if box j of cell i is not responsible for the object } \& \ IOU_{prediction}^{GroundTruth} < t \\ 0, otherwise \end{cases} \tag{3.8}$$

In other words, we ignore (do not backpropagate) boxes that do not have the biggest IOU and whose IOU is smaller than a given threshold t. Also, categorical cross-entropy is used for class loss, and for objectness loss, binary cross-entropy is used. The difference between categorical and binary cross entropy is that the class probability distribution in the categorical is the generalized Bernoulli's law for multi-class. Bernoulli's distribution is the particular case of categorical distribution where k is equal to two.

For inference the model uses the same architecture used for detection training.

3.1.3 Overview of YOLOv3

This subsection is based on YOLOv3 [26], April 2018. This model was the third iteration of the YOLO algorithm. Design changes were introduced, making the model bigger but more accurate.

3.1.3.A New techniques used in YOLOv3

In YOLOv3, the objectness loss is still calculated using binary cross-entropy instead of mean square error. For the class loss, binary cross-entropy is also used, allowing for multi-label classification.

The feature extractor was updated to Darknet-53, which can be checked in the YOLOv3 paper [26] and in the table 3.2. The reason why the network is better at feature extraction than Darknet-19 is because of the usage of residual blocks, introduced by ResNet [4], which allowed a much deeper network (53 instead of 19 convolutional layers). Darknet-53 is pre-trained on ImageNet, and then additional layers are added to train for detection according to figure 3.4. Leaky ReLU is still used as the activation function in all layers, including in the residual blocks.

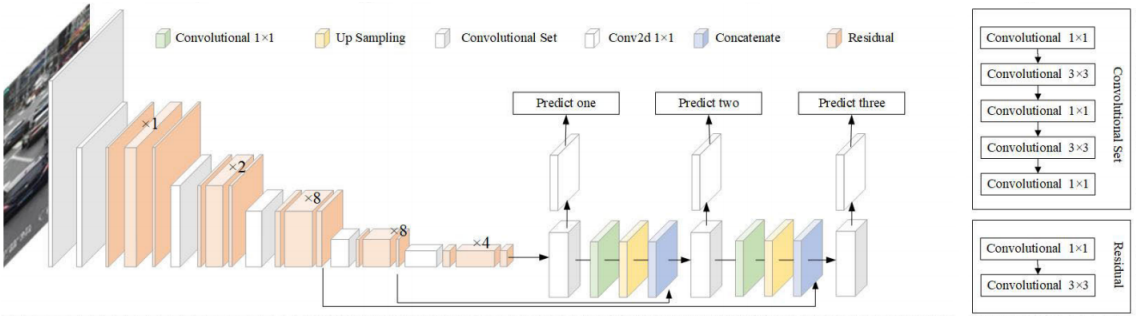

Figure 3.4: Detection architecture of YOLOv3, courtesy of [3]. Until the $\times 4$ residual block we have the backbone, Darknet-53 (for pre-training, Avgpool, Connected and Softmax layers are added at the end of the backbone). The $\times n$ represents the number of times the block is repeated.

Figure 3.5, taken from the ResNet paper [4], represents the residual block. Basically, the layers that output F(x) are added to x to calculate the residual of x, thus making it more accurate.

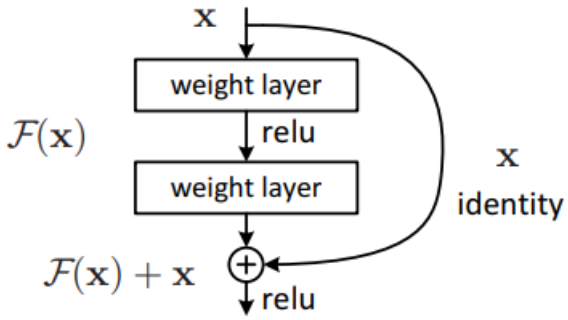

Figure 3.5: Residual block, courtesy of [4]. In this context a weight layer is a convolutional layer.

As we can see from figure 3.4, YOLOv3 produces 3 output predictions at different scales. Before prediction, 2 and 3 passthrough layers are used similarly to YOLOv2. YOLOv3 still uses k-mean clustering to produce 3 box priors per output, so at each output we get a $S \times S \times (3 * (4 + 1 + 80))$ tensor (the 80 classes are from MSCOCO). YOLOv3 is pre-trained in ImageNet for classification at 256×256 and then trained for detection in MSCOCO at 416×416.

Table 3.2 Darknet-53: The backbone for YOLOv3. For classification training, the last three layers are added to the backbone. The n× represents the number of consecutive residual blocks on the network. The residual blocks are highlighted with a darker color. On the filter size column, the letter "s" represents the stride of the convolution. It is worth noting that there are no pooling layers following the convolutions. Instead, a stride of two is used to reduce the feature space.

	Type	Number of filters	Filter size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3s2	128 × 128
1×	Convolutional Convolutional Residual	32 64	1 × 1 3 × 3	128 × 128
	Convolutional	128	3 × 3s2	64 × 64
2×	Convolutional Convolutional Residual	64 128	1 × 1 3 × 3	64 × 64
	Convolutional	256	3 × 3s2	32 × 32
8×	Convolutional Convolutional Residual	128 256	1 × 1 3 × 3	32 × 32
	Convolutional	512	3 × 3s2	16 × 16
8×	Convolutional Convolutional Residual	256 512	1 × 1 3 × 3	16 × 16
	Convolutional	1024	3 × 3s2	8 × 8
4×	Convolutional Convolutional Residual	512 1024	1 × 1 3 × 3	8 × 8
	Avgpool Connected Softmax		Global 1000	

3.1.4 Overview of YOLOv4

This subsection is based on YOLOv4 [5], of April 2020. This model was the fourth iteration of the YOLO algorithm. In this paper, several experiments are conducted to find the best techniques to apply to YOLO algorithm. A set of candidate techniques is chosen for each technique group, and then by either literature review or empirical experiments, the most optimal one is chosen. Some novelty techniques were introduced in this paper.

3.1.4.A New techniques introduced on YOLOv4

As in previous iterations of YOLO and as standard practice when using convolutional neural networks (CNNs) to solve object detection problems, a set of techniques and methods are used to improve the network's accuracy. In the paper, the authors begin by formally dividing the set of available techniques and methods into two large groups:

- Bag of freebies: Methods that improve the network's accuracy without impacting the inference cost but increase the training cost.

- Bag of specials: Methods and modules that improve the network's accuracy but increase the inference cost.

The bag of freebies and the bag of specials can be subsequently divided into different classes as it follows in figures 3.6 and 3.7.

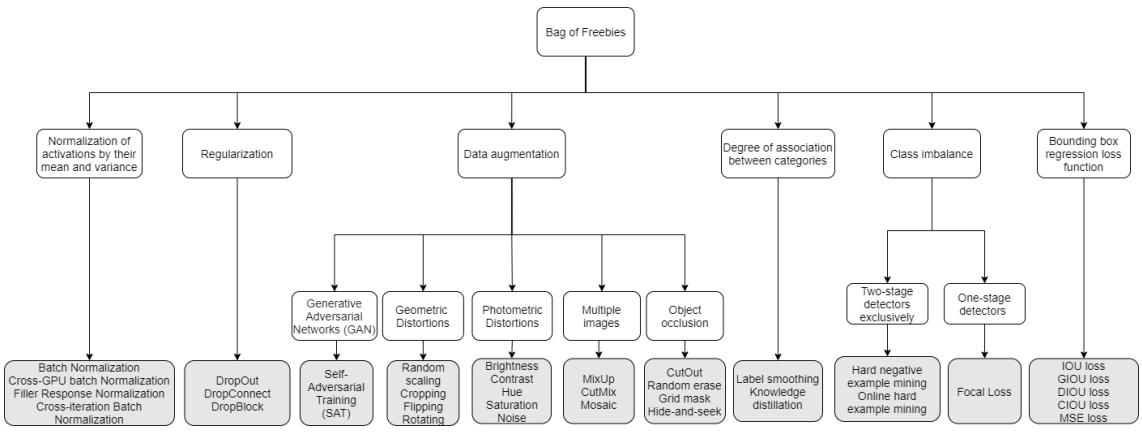

Figure 3.6: Bag of freebies taxonomy. The references for each technique are not provided. However YOLOv4 paper [5] contains all the references. The techniques highlighted with a darker background color are the candidates for each category of bag of freebies.

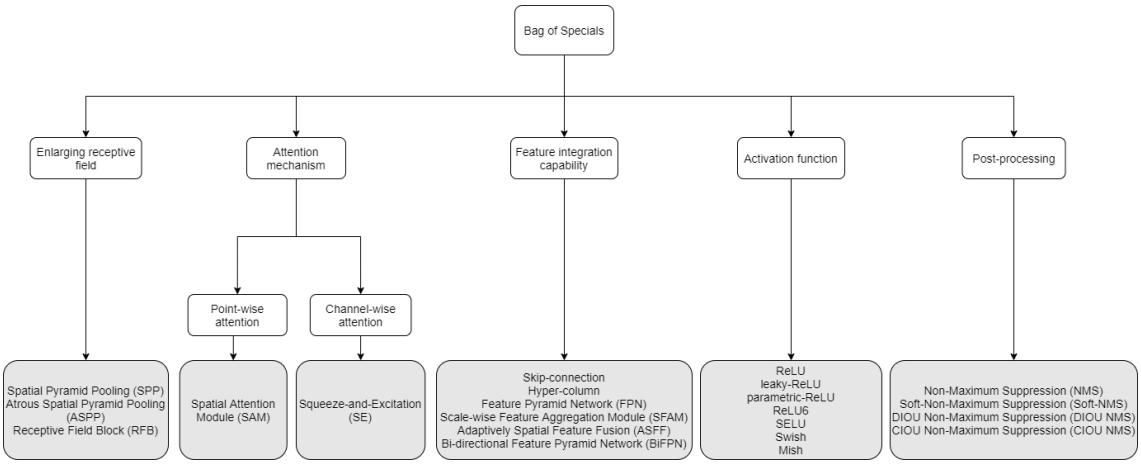

Figure 3.7: Bag of specials taxonomy. The references for each technique are not provided. However YOLOv4 paper [5] contains all the references. The techniques highlighted with a darker background color are the candidates for each category of bag of specials.

Some techniques and methods from previous YOLO iterations fit the taxonomy presented on figures 3.6 and 3.7. We did not previously categorize them not to present unnecessary concepts since some of these categories did not exist when the first three iterations were released. Regarding YOLOv4 we will only discuss the techniques that were chosen out of the candidates.

A modern one-stage object detector is comprised of 3 structures:

- Backbone: The part of the network which is responsible for the feature extraction and is usually pre-trained.
- Neck: The part of the network that collects feature maps from different stages of the network.
- Head: The part of the network responsible for the detection. It can be a one-stage or two-stage detector.

The techniques of the bag of freebies and bag of specials improve the overall network. Some of these techniques target the backbone portion of the network or the detector portion:

- Bag of freebies for backbone:
  - Data Augmentation: CutMix and Mosaic [5].
  - Regularization: DropBlock [63].
  - Degree of association between categories: Label smoothing [64].
- Bag of specials for backbone:
  - Activation function: Mish [65] (on previous YOLO iteration leaky-ReLU was used).
  - Increasing feature integration capability: Cross stage partial connections (CSP) [66] and multi-input weighted residual connections (MiWRC) (both are skip-connection methods).

- Bag of freebies for detector:
  - Data Augmentation: Mosaic and Self-Adversarial Training (SAT) [5].
  - Regularization: DropBlock [63].
  - Normalization of activations by their mean and variance: a variation of Cross-Iteration Batch Normalization (CBN) [67] is used. The difference is that the statistics are collected between mini-batches in a single batch.
  - Bounding box regression loss function: complete intersection over union (CIOU) [68] loss function.
  - Other: Grid sensitivity elimination, Cosine annealing scheduler [69], Optimal hyper-parameters (using a genetic algorithm), random training shapes and sizes (already used on previous YOLO iterations), and multiple anchor box priors for a single ground-truth (already used on previous YOLO iterations).
- Bag of specials for detector:
  - Activation function: Mish [65] and leaky ReLU.
  - Post-processing: distance-intersection over union non-maximum suppression (DIOU-NMS) [68] (previous YOLO iterations used regular non-maximum suppression (NMS)).
  - Enlarging receptive field: Spatial Pyramid Pooling (SPP) [70].
  - Attention mechanism: Spatial Attention Module (SAM) [71].
  - Path-aggregation block: Path Aggregation Network (PAN) [72].

Mosaic is a data augmentation technique that mixes four input images in the same image. These images can be cropped and resized versions of the originals. CutMix mixes only two images. These data augmentation techniques apply regularization to the network preventing overfitting. A novelty data augmentation technique introduced in this paper was SAT. SAT alters the input image instead of the weights (keeps the weights constant) during the backpropagation, generating a new altered version of the image that is then fed to the network. This is done with a probability of 50% every two mini-batches applied to all images of the mini-batch, and since SAT is changing the input image values, it has its optimizer. With SAT we can target through backpropagation the specific areas of the image that the network is using to detect the object. Contrary to what is expected, altering the input image degrades the network performance in comparison with the original [73].

DropBlock [63] is a regularization technique similar to Dropout. In Dropout, the weights of the CNN are randomly turned off during the backpropagation. One problem of Dropout is that the spatial information of a given convolutional layer can still flow through the network despite turning off some weights of the feature map (this problem does not apply to connected layers). Pixels that are close to each other share more spatial relations, and turning off random pixels could not be enough to make dropout work in convolutional

layers. DropBlock works by, in a given area of a feature map, for each pixel, drawing a result from a Bernoulli distribution. If the result is positive, a block of pixels surrounding the chosen pixel is turned off, thus solving Dropout's problem. Since we are using the "vanilla" version of YOLOv4, it will not contain any Dropout or DropBlock layer.

Label smoothing is a technique used to regularize and prevent over-confidant models. For a training example, an object contains a ground truth label with ones for the correct classification and zeros for the remaining classes. This poses a problem:

- The model can overfit because the maximum one is not achievable. The model is never truly correct [64].

- The model can become overconfident by assuming it is supposed to assign a probability of 100% to a specific class. If we have hard labels and then apply a softmax activation function, the resulting vector would be much larger than what would happen using soft labels, thus encouraging the previously stated [64].

Given the ground-truth hard label distribution $q(k|x) = \delta_{k,y}$, the following formula is used to smooth the labels:

$$q'(k|x) = (1-\varepsilon)\delta_{k,y} + \varepsilon u(k) \tag{3.9}$$

Where k is a label, x is an example, $\delta_{k,y}$ is the Dirac impulse for the ground-truth label y, ε is the smoothing parameter, and $u(k)$ is the uniform distribution.

One of the goals of YOLOv4 is to be trainable in a single graphics processing unit (GPU), thus requiring memory efficiency. One of the problems with batch normalization is that a small min-batch can lead to bad statistics causing noise because there could be not enough examples in the mini-batch to make a good representation of the dataset. CBN solves this problem by feeding the mean and variance of the previous iteration to the next iteration by summing them. Even though the mean and variance change between iterations, because the weights also change, we can approximate these changes using the Taylor series expansions of the mean and variance in order to the weights [67].

3.1.4.B Loss function, training and inference of YOLOv4

YOLOv4 loss function has been changed. Now it uses soft labels for the class predictions, and it uses CIOU for the bounding box regression loss. It still uses binary cross-entropy. Equation 3.10 is the YOLOv4

loss function:

$$1 - IOU + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha V$$
$$- \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{obj} \left[\hat{C}_i log(C_i) + (1 - \hat{C}_i log(1 - C_i)) \right]$$
$$- \lambda_{noobj} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{i,j}^{noobj} \left[\hat{C}_i log(C_i) + (1 - \hat{C}_i log(1 - C_i)) \right]$$
$$- \sum_{i=0}^{S^2} \mathbb{1}_{i,j}^{obj} \sum_{c=1}^{classes} [\hat{p}_i(c) log(p_i(c)) + (1 - \hat{p}_i(c)) log(1 - p_i(c))] \quad (3.10)$$

Regarding the CIOU portion of the network, ρ represents the euclidean distance, b and b^{gt} are the centers of the predicted box and ground truth box, c is the diagonal length of the box that results from overlapping both boxes, V implements the consistency of the aspect ratio:

$$V = \frac{4}{\pi^2} (arctan \frac{w^{gt}}{h^{gt}} - arctan \frac{w}{h})^2 \quad (3.11)$$

And α is the trade-off parameter:

$$\alpha = \begin{cases} 0, \text{if IOU} < 0.5, \\ \frac{V}{(1 - IOU) + V}, \text{if IOU} \geq 0.5 \end{cases} \quad (3.12)$$

During inference time for YOLOv4 uses DIOU-NMS, which can be formally defined as:

$$s_i = \begin{cases} s_i, \text{if } IOU - \frac{\rho^2(M, B_i)}{c^2} < \varepsilon, \\ 0, \text{if } IOU - \frac{\rho^2(M, B_i)}{c^2} \geq \varepsilon \end{cases} \quad (3.13)$$

Where M is the predicted box with the highest classification score, B_i is the box being compared, s_i is the classification score, and ε is the NMS threshold. In this type of NMS the IOU, and the distance between the central points of the two boxes is considered, which is essential because the same cell can predict multiple boxes, and those boxes could refer to different objects. Thus we should not eliminate boxes that have a large center distance. YOLOv4 is pre-trained in ImageNet for classification at 256×256 and then trained for detection in MSCOCO at 416×416. To choose the hyper-parameters for the network a genetic algorithm is used.

The figure 3.8 shows the YOLOv4 architecture.

SPP [70] is utilized in a slightly different way here than in the original paper. In the original paper, the purpose is to allow input size changes of a network without having to change the fully connected layer by using max-pooling to a set of feature maps with a fixed number of bins. In YOLOv4, SPP is used to originate feature maps that are then concatenate to the originals, which allows for a larger receptive field, that according to [5], benefits the detector. SPP, SAM [71] which integrates the attention mechanism, and PAN which introduces the short-circuit concept, constitute the neck of the network. However YOLOv4 uses modified versions of the SAM and PAN.

YOLOv4 also uses CSP connections for the backbone. CSP connections were introduced to solve the problem of duplicated gradient problem when using dense blocks [66]. Dense blocks use concatenation

type skip connections, allowing for a better gradient flow since all the layers are connected. However, this originates the problem of duplicated gradient.

The five downsample blocks constitute the Dense Block where CSP connections are applied. Inside of downsample blocks a residual block exists (figure 3.5), thus implementing an addition type skip-connection.

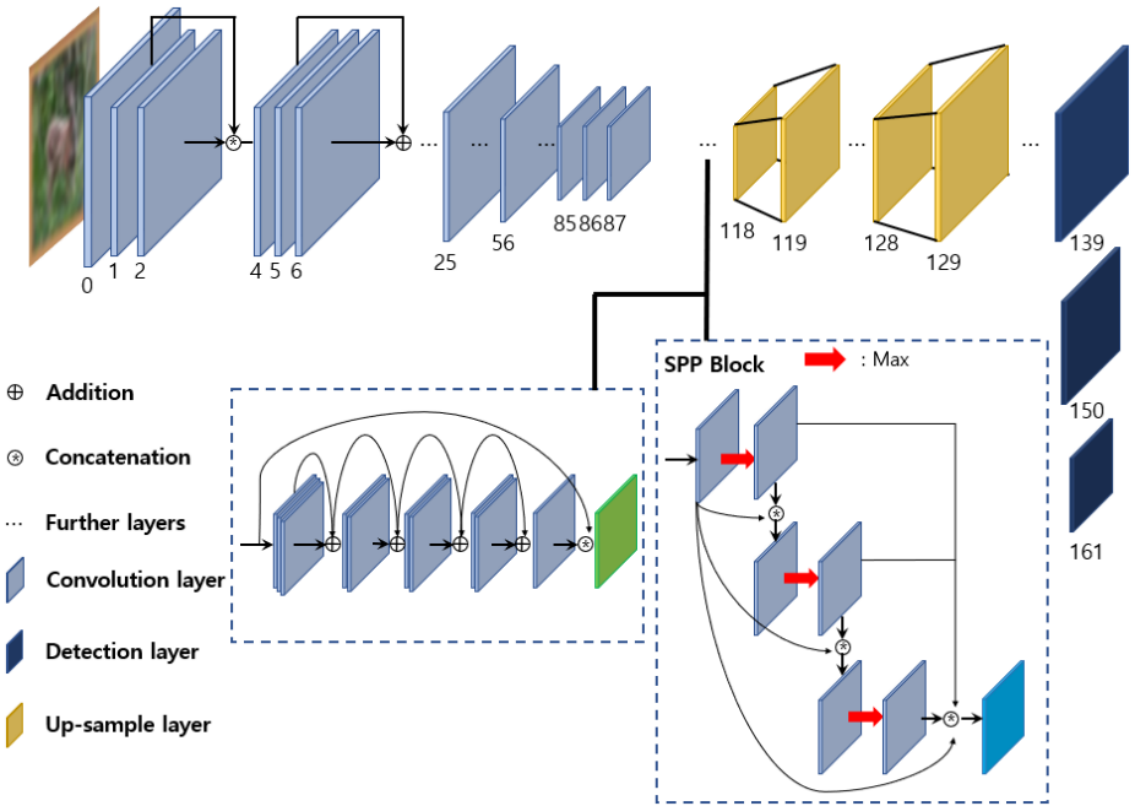

Figure 3.8: YOLOv4 architecture courtesy of [6]. The blue blocks on the top-left corner represent the 5 downsample blocks that constitute the backbone (these blocks are similar to Darknet-53 blocks, fig.3.2). The full architecture can be checked at `https://github.com/AlexeyAB/darknet/blob/master/cfg/yolov4.cfg`. The left side of the bifurcation represents the CSP connections that are applied to the downsample blocks originating the backbone, CSP-Darknet53. The right side of the bifurcation is the SPP which is part of the neck. The head of the detector in YOLOv4 is the same head used in YOLOv3.

3.2 Detection framework: Darknet

The framework we used to run the YOLOv4 algorithm is called Darknet. Darknet is the framework used by the authors of the original YOLO papers, including PP-YOLO and Scaled-YOLO. There are two forks of Darknet, the first one authored by J. Redmond [54] where the first three versions of YOLO and some other versions were implemented. However, J. Redmond abandoned the project, which A. Bochkovskiy now keeps, thus creating another version of Darknet [5] [58], available at `https://github.com/AlexeyAB/darknet`, where YOLOv4, PP-YOLO, Scaled-YOLO and others were implemented. It is also possible to run all the previous YOLO versions in A. Bochkovskiy's fork.

3.2.1 What is Darknet

Darknet is a framework written primarily in C/C++ and Compute Unified Device Architecture (CUDA). Darknet is used to train and use neural networks, allowing users to write .cfg files to configure the network. In this work, we are going to use the implementation of A. Bochkovskiy of YOLOv4.

3.2.2 Why use Darknet

Darknet is the framework where the YOLO versions were developed. Thus, we ensure we get the state-of-the-art results announced on the papers by using the original implementation. Lastly, training a network on ImageNet and MSCOCO can be a time-consuming task. By using pre-trained weights available on the Darknet repository, we can better manage the hardware resources.

3.3 Optimization framework: TensorRT

The framework we used to perform model optimization is called TensorRT [74]. TensorRT was developed by NVIDIA to target NVIDIA GPUs. TensorRT, among other optimization techniques, supports 16-bit floating point (FP16) and 8-bit integer (INT8) quantization. TensorRT receives the trained model, optimizes it, and then executes inference using the TensorRT Runtime engine.

3.3.1 What is TensorRT

TensorRT is a programmable inference accelerator created by NVIDIA to optimize neural network (NN) models allowing to extract more performance from NVIDIA GPUs. It performs layer and tensor fusion, either by fusing multiple layers that execute after another or by concatenating the inputs of different layers that are applying the same operation [75]. It also applies pruning, quantization, as previously stated, and automatically selects the most optimal kernel functions that take into account the hardware used [75].

3.3.2 Why use TensorRT

Performing quantization in Darknet is a challenge since most of it is written in C. Currently, Darknet does not support any Quantization libraries, which means we must somehow leave Darknet and deploy the model on another framework.

TensorRT is a framework designed explicitly for NVIDIA GPUs which would allow extracting the most performance with minimal accuracy degradation. It also supports ONNX [76], which is an open format build to represent machine learning models, thus allowing us to deploy our model in TensorRT even though Darknet does not support it.

3.4 Summary

In this chapter, we started by describing the YOLO family of models. Even though we are only going to use YOLOv4 to understand the nuances and the thought process that led to YOLOv4, the previous versions

were explored. We first start by giving the reasons for the model choice, followed by an explanation of how the YOLOv1 model works, where we describe the architecture, the key features, the loss function, the training, and inference process. For the more recent versions (YOLOv2, YOLOv3, and YOLOv4), we try to successively explain the novel features and updates that were built on top of previous versions.

Finally, we briefly discuss the frameworks used to train and optimize the model giving the reasons for the choices. In the next chapter, we discuss the process of creating a dataset for the experiments.

4

Construction of the dataset used for detection

Contents

As previously stated, detection datasets, especially polyp detection datasets, are not an abundant resource. This is due to polyps not being as common as everyday objects, and specialists in the medical field must perform the annotations. For this reason, we combined datasets from different sources to train/test the model, assess the inference speed, and validate our work in a way where we could compare it to the work of others.

This chapter addresses the construction of our dataset and the reasons for our choices.

In this chapter, we will refer to precision validation as the results of counting the samples deemed as true positive (TP), false positive (FP), true negative (TN), and false negative (FN) (confusion matrix).

4.1 Obtaining the data

From all the datasets available (table 2.2), we manage to obtain the following:

- Kvasir-SEG [39] (image dataset);
- CVC-ClinicDB [35] (image dataset);
- CVC-ColonDB [36] (image dataset);
- CVC-EndoSceneStill [37] (image dataset), which is composed of CVC-ClinicDB and CVC-ColonDB;
- ETIS-Larib [38] (image dataset);
- Colonoscopic Dataset [42] (video dataset).

The Colonoscopic Dataset contains the exact same videos in white light (WL) and narrow band imaging (NBI). Since all the other datasets are WL for the sake of consistency, we decided only to use the WL videos. However, this dataset is primarily used to assess the frame rate during inference, so using the NBI would not make a difference.

Even if we wanted to assess precision, we would not be able to because no ground truth boxes are provided with the dataset, and it is not in our capacities as Engineers to perform the work of a medical field specialist by placing the annotations by ourselves.

We made several attempts to obtain the remaining datasets from table 2.2, including filling forms, obtaining them in the way described in the supporting papers or websites, and contacting the authors by multiple means. However, all the attempts failed.

4.2 Converting the labeled data in Darknet format

Darknet reads the annotations in a specific format. Every image is associated with a .txt file with the same name where the object classes and respective bounding boxes coordinates are annotated. Each line in the .txt file contains 1 object in the following format: ”class center_x center_y width height”. All the values are float except the class, which is an integer:

- center_x = pixel_value_center_x/image_width
- center_y = pixel_value_center_y/image_height
- width = box_width/image_width
- height = box_height/image_height

Additionally, two .txt files must be created containing the paths to the training images and testing images, one .names file containing the different class labels (one per line), and a .data file containing the number of classes and the paths for the previously described files.

4.3 Training and testing dataset

For training, we randomly chose 80% (800/1000) of the images from Kvasir-SEG and manually chose 80% of the images (240/300) from CVC-ColonDB. The remaining images were used to test the model during training. The weights are evaluated on the testing dataset every few iterations during training to see how the model is evolving.

We manually chose the images from CVC-ColonDB because this dataset contains multiple images of identical polyps but from different angles. We wanted to make sure that, during the testing phase, we were testing the model in polyps it never saw. The dataset contained 15 polyps, each with 20 frames.

Kvasir-SEG annotations came in a single .json file. The information given was the class label, the image width and height, and the Bounding Box minimum and maximum for the x-axis and y-axis. For this reason, a Matlab script was created to convert the Bounding Boxes to the Darknet format. Additionally, the script allows us to manually check if all the boxes were correct. We noticed that the specialists made some accidental clicks when drawing the boxes which we chose to remove. Most of these boxes were too small to be visible; however, we removed all of the mistakes using a threshold of 12 pixels for height and 12 pixels for width.

For CVC-ColonDB, we used the binary masks mask images to generate the proposals for bounding boxes and then convert them to the Darknet format. We manually checked all of the bounding boxes using the same script. Finally, we converted the .bmp images to .jpg for the sake of consistency with the remaining dataset.

The figure 4.1 illustrates two examples of the generated bounding boxes:

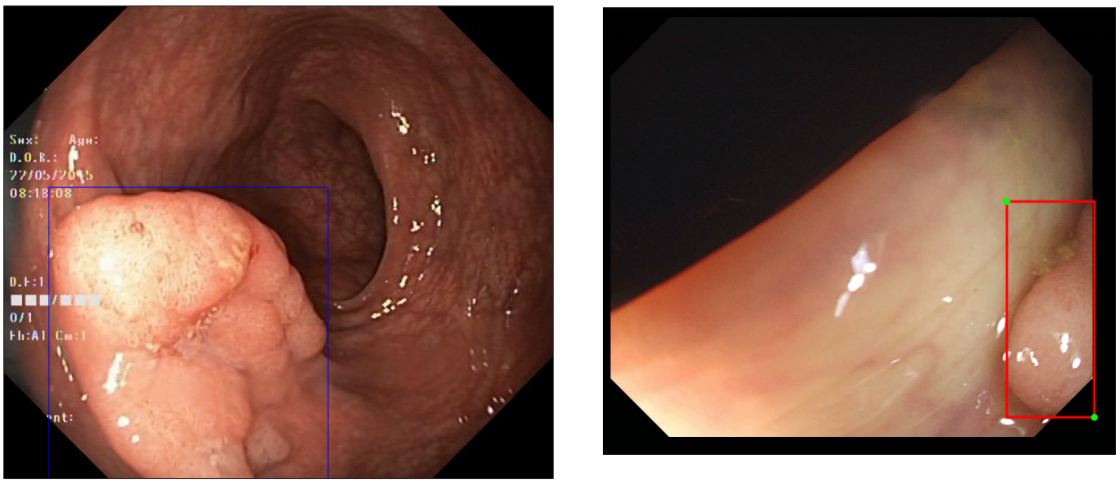

Figure 4.1: Two examples of the bounding boxes created using our MatLab scripts. The first image is from Kvasir-SEG and the second is from CVC-ColonDB.

Lastly, we used DarkMark [77] to validate all of the bounding boxes generated. DarkMark is a C++ graphical user interface (GUI) application that, among other use cases, allows the user to place bounding boxes or read the currently placed ones. The advantage of DarkMark is that it reads the .txt file in the Darknet format, allowing us to make sure there were no errors in the box conversions since we could not use the Darknet format to plot the boxes in MatLab.

4.4 Validation datasets

For validation purposes we used the following datasets:

- CVC-ClinicDB [35] (image dataset);
- ETIS-Larib [38] (image dataset);
- Colonoscopic Dataset [42] (video dataset);

4.4.1 Validating the model in precision

As discussed in chapter 2, most of the studies on polyp detection used CVC-ClinicDB and ETIS-Larib to evaluate their models (table 2.5). We decided to follow a similar approach, so that we could easily compare You Only Look Once (YOLO)v4 to the results from the studies. To generate the bounding boxes, we used the same process as described for CVC-ColonDB, including the verification using DarkMark. The metrics used to evaluate the model will also be the same used in all the studies deemed as relevant (table 2.5). Therefore precision, recall, F1, and F2 metrics are going to be used. On chapter 5 we explain the reasons for the metric choices in more detail.

4.4.2 Validating the model in speed

The only video dataset we managed to obtain was Colonoscopic Dataset. Therefore, we use these videos to compare the model's frame rate pre-quantization and post-quantization. From the 78 short videos, 3 were selected. We randomly chose a single video from each of the three types of polyps available in this dataset: adenoma, hyperplastic and serrated. The selected videos were the fifth video from the adenomas, the ninth video from the hyperplastic polyps, and the fourth video from the serrated polyps.

4.5 Summary

This chapter addresses the dataset creation in order to evaluate the model correctly. We started by discussing the publicly available datasets and the ones we were able to obtain.

Diferent datsets present the labeled data in diferent formats. Darknet has a specific format required to properly read the labels. Darknet format and the process of converting the data were discussed.

Finally, out of the available datasets, we explain the thought process for our choices in splitting those datasets into training/testing and validation.

For training/testing, we combined Kvasir-SEG and CVC-ColonDB. We separated speed validation using Colonoscopic Dataset and precision validation using ETIS-Larib and CVC-ClinicDB. In this context, precision means quantifying the results in a confusion matrix format and not the actual precision metric.

In the next chapter, we use the datasets to perform experiments to achieve good performance in terms of precision and set up the experiments to assess the trade-off between precision and speed pre-quantization and post-quantization.

5

Using Darknet to perform polyp detection

Contents

This chapter describes the experiments performed in order to increase the model's accuracy in the testing and validation datasets. Firstly we apply transfer learning. Then we conduct experiments to find the optimal width and height to train our network. After, a few pre-processing techniques are tested, and alternative optimizers are also tested. In the next set of experiments, we review and try different data augmentation strategies to further increase the model's performance. We then test regularization methods and other techniques. Finally, we set up the experiments to compare the performance of the model with the quantized versions.

5.1 Metrics used to evaluate the model

To evaluate the model, a state-of-the-art review was conducted in chapter 2 to determine which metrics are currently being used. In the following subsection, we analyze and select the best fitting metrics for our work and formally define the metrics used.

5.1.1 Choosing the metrics to evaluate performance

To assess the performance of the model in terms of precision, several metrics should be used. The most used metrics across the literature are shown in table 2.5. These metrics are precision, recall, F1 score, and F2 score. Very few studies use accuracy, specificity, and receiver operating characteristic (ROC) curves. This is because in the context of polyp detection is not important to evaluate the performance of the detector on negative frames (frames that do not contain any object), as we concluded in chapter 2. In our work, the datasets used for evaluation do not have negative frames, so using metrics that depend on the existence of negative frames is not possible.

There is a trade-off between precision and recall, whose values can be shifted by varying the intersection over union (IOU) threshold and the confidence score threshold. By fixing the IOU threshold, precision-recall curves can be drawn. The area under the curve (AUC) of the precision-recall curves is equal to the average precision (AP), and in our case, since it is a single class problem, the AP is also equal to the median avarage precision (mAP). The confidence score is the value predicted by the model that reflects how confident the model is in its prediction.

In the literature, it is not defined what the ideal trade-off between precision and recall in the problem of polyp detection should be. However, it is established that high precision and high recall are desirable, so we can use the mAP metric instead since a large AUC is indicative of high precision and high recall.

For the sake of comparison with other studies, F1 and F2 scores are used due to their ability to combine precision and recall in a single metric.

In summary, the metrics we chose to evaluate the model's performance are precision, recall, F1 score, F2 score, and mAP. To assess inference speed, we use frames per second (FPS). The metrics used serve the following purposes:

- Precision: Of all the frames the model detected as polyps, precision calculates the percentage that are actually polyps;

- Recall: Of all the polyps in the dataset, recall calculates the percentage that the model detected;
- F1 score: Calculates the harmonic mean of precision and recall;
- F2 score: Averages precision and recall but lowers the importance of precision and increases the importance of recall;
- mAP: Combines precision and recall at a given IOU threshold for different confidence values.

5.1.2 Implementation of the metrics adopted

We define true positive (TP) as a bounding box prediction that correctly predicts a polyp by overlapping with the ground-truth box. A false positive (FP) indicates that a bounding box is predicted that does not overlap with the ground-truth box. A false negative (FN) occurs when no bounding box is predicted, and a ground-truth box exists. A true negative (TN) occurs when no bounding box is predicted, and a ground-truth box does not exist. In our datasets, there are no images without objects. Also, the concept of a TN does not apply to You Only Look Once (YOLO). Therefore TN are not considered.

For the precision and recall metrics, we choose the values ranging from 0.25 to 0.35 for the confidence threshold and a value of 0.5 for the IOU threshold. The metrics used can be formulated as:

$$Precision = \frac{TP}{TP+FP} \tag{5.1}$$

$$Recall = \frac{TP}{TP+FN} \tag{5.2}$$

$$Fbeta = \frac{(1+beta^2) \times Precision \times Recall}{beta^2 \times Precision + Recall}, \text{ thus} \tag{5.3}$$

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall}, \text{ and} \tag{5.4}$$

$$F2 = \frac{5 \times Precision \times Recall}{4 \times Precision + Recall} \tag{5.5}$$

Since polyp detection is a single class problem, $Q = 1$ (total number of queries) and the AP is equal to the mAP:

$$\text{mAP} = \frac{\sum_{q=1}^{Q} AP(q)}{Q} \tag{5.6}$$

where AP is formally defined as:

$$AP = \int_0^1 p(r)dr \tag{5.7}$$

Where $p(r)$ is the precision as a function of recall (precision-recall curve). For the AP calculation, we followed the PASCAL VOC 2011 AP metric [78].

5.2 Experiments

In this section, we train the YOLOv4 model and test different methods of training, parameter selection, regularization, and data augmentation. Next, we conduct a set of experiments that allow us to obtain results for the model performance in terms of speed and precision to allow us to then compare these results with the results of quantized versions of the model.

In some of the experiences, we use the NVIDIA RTX 2080TI Tensor Cores to train the model, which uses quantization. Using Tensor Cores can speed up training and inference without injuring performance for FP16/32 mixed-precision, as shown in a GitHub issue published by the author of Darknet: `https://github.com/pjreddie/darknet/issues/704`. Tensor Cores are units that specialize in matrix calculation using mixed precision [79].

5.2.1 Input size selection

It is essential first to find the input size that we use to conduct all the other experiments. Even though Darknet resizes each image while feeding them to the network, pre-resizing every image to the selected size allows the acquisition of the statistics presented in section 5.2.2.

We trained the model using our training dataset while testing it during training using our testing dataset. This experiment is then repeated using Etis-Larib to test during training. After the end of each training session, we select the weight that leads to the highest mAP and use these weights to calculate precision, recall, F1, and F2 score.

In this first set of experiments, we apply transfer learning using weights from the repository (`https://github.com/AlexeyAB/darknet`) that were pre-trained on ImageNet and then trained on Microsoft Common Objects in Context (MSCOCO). The justification for this approach is that, in worst-case scenario, it cannot be worse than using randomly initialized weights, and from our initial experiments, we found that transfer learning helped the model to converge faster.

We used random, jitter, and ordinary non-maximum suppression (NMS), also called greedy-NMS. Random and jitter are parameters of the YOLO layer, which is a type of layer in our model that outputs the predictions. We decided to fix these parameters before initiating the first set of experiments, leaving them with the default values as we are trying to find the optimal size for the input layer, and these parameters change the image and network size. The random parameter changes the network's size by values ranging from $1/1.4$ to 1.4 (for a duration of 10 iterations). We used 0.3 for the jitter, which changes the aspect ratio and size of the images by randomly cropping the images. For greedy-NMS, we use the value of 0.6 for beta as suggested by the author of [80].

We also set the batch size to 64 while using 8 subdivisions. We adopted these values because it was the largest batch size and the smallest number of subdivisions the NVIDIA RTX 2080TI could handle, and our goal at this stage is to train as fast as possible in order to find the best input size. In our work (and also in Darknet), a batch size of 64 means that during an iteration, 64 examples are fed to the network, but only 64/8 examples are fed to the graphics processing unit (GPU) simultaneously.

Table 5.1 Hyper-parameters used to find the best input image size. The burn-in parameter is the number of iterations in which the learning rate is slowly increased from zero to the initial value. The step/scale means that a factor scales the learning rate in the given iteration. The momentum uses moving averages so that the steps taken in the previous gradient updates are used to calculate the current step, preventing big shifts in direction. The decay or weight decay is a small penalty added to the loss function for regularization purposes.

Learning rate	Burn-in	Max-batches	Step/scale	Momentum	Decay
0.0002	1000	20000	15000/0.1 18000/0.1	0.9	0.0005

For the optimizer, we use mini-batch stochastic gradient descent with the parameters shown in table 5.1.

First, we test the sizes ranging from 160×160 to 384×384 pixels jumping in intervals of 32 pixels for width and height simultaneously. The choice for this interval is because we empirically tested the sizes 128×128, 224×224, and 416×416 and we observed that 224×224 led to better results. Darknet only accepts input sizes multiples of 32 pixels.

Since 224×224 produced the best results, we now test different aspect ratios by varying the width while having the height fixed at 224 or vice versa. We tested values from 160 to 384 for both width and height, and we found that the optimal size was 320×224. Results obtained for the testing dataset and Etis-Larib (one of the validation datasets) are presented in table 5.2.

Table 5.2 Results obtained using the best weights during training and testing for the testing dataset and Etis-Larib validation dataset, using 320×224 for the input size, transfer learning, greedy-NMS, and using random and jitter. Values were obtained for a IOU threshold value of 0.5 and a confidence threshold value of 0.25.

Dataset	mAP	Precision	Recall	F1	F2
Testing	93.97%	93.98%	93.98%	93.98%	93.98%
Etis-Larib	70.14%	85.00%	65.38%	73.91%	68.55%

Figure 5.1 shows the training charts we obtained for the 320×224 training results. As observed in the right-hand side chart, the best results for the Etis-Larib validation dataset were obtained after iteration 14000. This is the reason why we use such a large number of iterations.

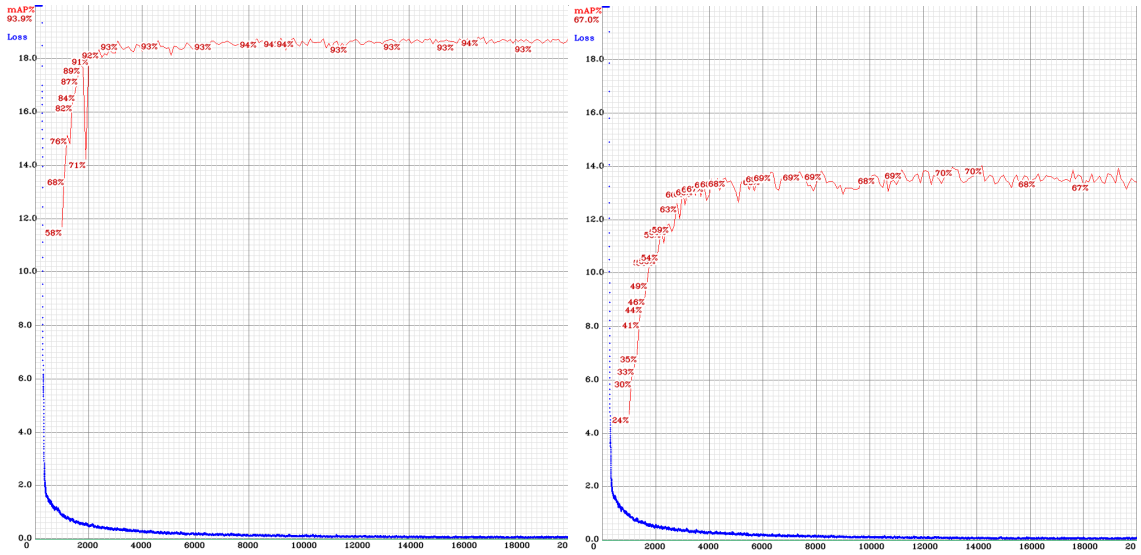

Figure 5.1: Training charts obtained for the testing dataset and Etis-Larib dataset, respectively, using 320×224 for the input size, transfer learning, greedy-NMS, and using random and jitter. The blue line represents the loss, and the red line the mAP.

5.2.2 Image pre-processing and post-processing

We start by first resizing the images from the training dataset to 320×224. Every 100 iterations, the weights are saved, and a script was created to find the best weights for a given dataset. We obtained the results across the testing dataset, Etis-Larib validation dataset, and CVC-ClinicDB validation dataset in table 5.3.

Table 5.3 Results obtained for the testing dataset, Etis-Larib validation dataset, and CVC-ClinicDB validation dataset after training the network with the re-sized dataset. Transfer learning, greedy-NMS, random, and jitter data augmentation were used. Values were obtained for a IOU threshold value of 0.5 and a confidence threshold value of 0.25.

Dataset	mAP	Precision	Recall	F1	F2
Testing	94.47%	93.70%	95.11%	94.40%	94.83%
Etis-Larib	70.54%	85.80%	66.83%	75.14%	69.92%
CVC-ClinicDB	86.86%	81.87%	83.90%	82.87%	83.49%

We also collected statistics regarding the datasets displayed in table 5.4. We did not collect the statistics for the testing dataset because the model was already performing extremely well, as table 5.3 shows. As we mentioned in chapter 3, YOLO might have some trouble detecting small objects. We believe this is because if an object is small, to begin with, when we resize the image to a smaller size, the object might become too small to be detectable. Under this assumption, we executed a few tests to analyze if our model had trouble detecting small polyps.

We can see in table 5.3 that the model generalizes relatively well except for the recall metric in Etis-Larib. This tells us that the model is having trouble detecting some of the polyps in this particular dataset. We can play with the bias-variance trade-off in order to lower the variance, which can be done by Regularization techniques or by introducing more data in the dataset through data augmentation techniques. An-

other way to increase recall is to play with the precision-recall trade-off by varying the confidence threshold.

Table 5.4 Statistics gathered for the training dataset, Etis-Larib validation dataset, and CVC-ClinicDB validation dataset.

Dataset	Objects	Images	Image size	Min object size	Object avg. size	Max object size	Width SD	Height SD
Training	1098	1040	320 × 224	18 × 13	114.45 × 95.26	320 × 224	59.87	47.10
Etis-larib	208	196	1225 × 966	39 × 43	229.21 × 233.47	776 × 623	158.34	147.17
CVC-ClinicDB	646	612	768 × 576	17 × 17	115.76 × 108.54	306 × 262	64.67	58.26

In order to verify the impact of small polyps, we used DarkMark to generate tiled versions of the images and zoomed-cropped versions of the images for the training dataset. Tiling splits a large image into smaller images that we feed one at a time to train the network, while zoom-crop randomly zooms and crops the images. We used these two new training datasets to train the network. We removed from the datasets the frames with no bounding boxes that were originated. However, these two experiments actually hurt the performance of our model. We believe this was because the smaller polyps were still large enough for our model to learn, and the process of tiling and cropping splits the larger bounding boxes across multiple images, which could have damaged the performance.

We manually evaluated the model's results without tiling and zooming, and the model was able to detect small polyps on the testing dataset. However, in Etis-Larib, some of the polyps are considerably smaller when resized, as we can see in table 5.4. In this dataset, we identified a few frames that could be potentially troublesome and in which the model was unable to detect the polyps correctly. We constructed zoomed versions of the images and tested them with the model, which still performed poorly, leading us to believe that the size of the polyps is not a problem for our model.

We also found other wrong detections that the detector inadvertently performs: mistaking fecal matter by polyps and mistaking light reflection by polyps.

Upon analyzing the predictions made by the model on CVC-ClinicDB, we did not find any relevant information regarding the reasoning behind the wrong detections the model performed.

Finally, for post-processing, we replaced greedy-NMS with distance-intersection over union non-maximum suppression (DIOU-NMS) using the recommended value of 0.6 for beta, but it led to worse performance, so we kept using greedy-NMS.

Figure 5.2: Examples of images where the model performed poorly on the Etis-Larib dataset. The polyp is missed in the left-hand side image, and two false positives occur, one where the fecal matter is mistaken by polyp. On the center image, the reflection is mistaken by a polyp, and a polyp is missed. On the right-hand side image, the polyp is missed.

5.2.3 Adam optimizer and cosine annealing scheduler

We experimented with the optimizer policy, changing it from steps to cosine annealing scheduler [69]. Several methods of annealing exist. However, we experimented with cosine because it allows the training to start with a high learning rate that slowly reduces as we approach the global minimum, then gradually decreases and has it gets closer to the global minimum, taking very small decrements in the last iterations. Using this technique, we can set multiple restarts allowing us to repeat the learning rate cycles at different weights.

Using cosine annealing scheduler led to worse performance results in the testing and Etis-Larib datasets and better performance in CVC-ClinicDB dataset, but it did not make the model converge faster. Since the results for Etis-Larib were significantly worse, we did not use cosine annealing scheduler and kept using mini-batch stochastic gradient descent without cosine annealing scheduler.

We also experimented with Adam optimizer [81]. The Adam method combines AdaGrad and RMSProp, and it is a commonly used optimizer. We wanted to verify if Adam optimizer can improve our model performance or, as reported in [81] if it leads to faster convergence, thus speeding up the training. Using the Adam optimizer, compared with stochastic gradient descent, we achieved better results in mAP across the testing, Etis-Larib, and CVC-ClinicDB datasets. However, the results for F1 and F2 scores were superior on the CVC-ClinicDB dataset but worse on the testing and Etis-Larib datasets. The results are shown in table 5.5. Using Adam optimizer showed promising results, but it might require adjusting the confidence threshold. We still choose to keep using stochastic gradient descent because we experiment with Cross-Iteration Batch Normalization (CBN) in section 5.2.5, and Adam optimizer does not work well with CBN. However, in section 5.2.5 we compare the results using regular batch normalization and Adam optimizer vs. CBN and stochastic gradient descent.

Table 5.5 Results obtained using Adam optimizer with a learning rate of 0.0001 and a weight decay value of 0.0005. Values were obtained for a IOU threshold value of 0.5 and a confidence threshold value of 0.25 for the testing dataset, Etis-Larib validation dataset, and CVC-ClinicDB validation dataset.

Dataset	mAP	Precision	Recall	F1	F2
Testing	94.60%	92.28%	94.36%	93.31%	93.94%
Etis-Larib	71.89%	75.00%	69.23%	72.00%	70.31%
CVC-ClinicDB	86.97%	89.62%	84.21%	86.83%	85.24%

5.2.4 Data augmentation

As previously stated, for polyp detection, a limited number of training examples is available in public datasets. Data augmentation can serve as a solution to prevent overfitting and enhance the size and quality of the training dataset [82]. In this set of experiments, classic data augmentation techniques, following the guidelines of [82], and other novel techniques introduced by [5] were used. Some of the techniques we experimented with were deployed on the fly during training, and others were applied beforehand by creating new images. The work of [82] provided a taxonomy of the different methods of data augmentation as well as detailed explanations on each of the techniques.

Data augmentation and adding more frames to the dataset are two common tactics deployed to increase performance metrics. In this subsection, data augmentation will be used to improve performance, particularly in the Etis-Larib dataset.

Table 5.6 compiles the different techniques of data augmentation and the respective results for the validation datasets. We first tried using mosaic, which is described in section 3.1.4.A. Mosaic is implemented in Darknet, and this technique is applied on the fly during training. Following mosaic, we applied blur, which blurs the image's background, leaving the object intact, and is applied on the fly during training using a chance of 50% each time a training sample is fed to the network. After testing blur, we tried different combinations of hue, saturation, and exposure. We included the values that led to the best results, from the values we tested, in table 5.6. Darknet allows the values of saturation and exposure to vary from 0.0 to 10.0 and hue to vary from 0.0 to 1.0. The value 0.0 refers to the raw image, and Darknet applies these augmentation techniques by randomly varying the values from zero to the set value. We also changed the confidence threshold from 0.25 to 0.35 as also stated in table 5.6. It is worth noting that this change only affects the precision, recall, F1, and F2 scores since the mAP metric is independent of the confidence threshold. As shown in table 5.6 blur was injuring the performance of the model when used in combination with hue, saturation, and exposure. For this reason, we stopped using blur.

After this point in the experiments, we started to use Tensor Cores to speed up the model's training due to time constraints. The usage of Tensor Cores allowed training the model faster without injuring the model's performance. For testing purposes, we did not use the Tensor Cores. We tried to use Flip and Rotation in separate experiments and together. We added horizontally and vertically flipped versions of the images to the dataset and then trained the model. We rotated the original images by 90º, 180º, and 270º and added them to the original training dataset, and trained the model. However, the best results were achieved when we used vertical flip and rotation together by flipping the images vertically and then applying rotation by 90º, 180º, and 270º to both the original and flipped images. We only flipped vertically because some of the new images would be duplicated if we flipped vertically and horizontally and then applied rotations. We then tried to add adding Gaussian noise, which is performed on the fly during training. This data augmentation type works by injecting into the image a matrix of random values drawn from a Gaussian distribution. Using Gaussian noise decreased the performance of the model.

We conclude the data augmentation portion of the experiments by using Self-Adversarial Training (SAT) with different adversarial learning rates. The best results were achieved when using an adversarial learning rate value of 1.0. However, SAT did not improve the performance of the model. The adversarial training is performed on the fly during training and is described in 3.1.4.A. However, the usage of SAT led to worse results, and thus we removed it before conducting the next set of experiments. Figure 5.3 exhibits examples of the data augmentation methods used to train the algorithm.

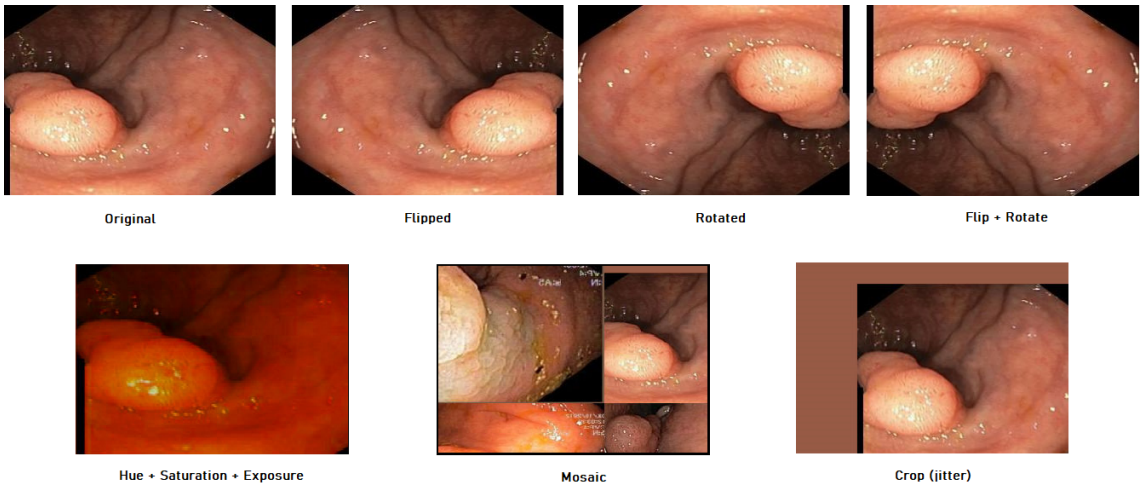

Figure 5.3: Examples of the data augmentation methods used to train the model. Multiple data augmentation methods can be combined during training. The crop (jitter) method was previously defined in section 5.2.1.

Table 5.6 All the experiments conducted using different types of data augmentation and respective results. F/R stands for Flip/Rotation, Gn stands for Gaussian noise, and SAT stands for Self-Adversarial Training. We highlighted the results we considered the best, which will be used in the subsequent experiments. The results were obtained for the two validation datasets, Etis-Larib and CVC-ClinicDB.

Data Augmentation Types								Datasets										Confidence threshold
								Etis-Larib					CVC-ColonDB					
Mosaic	Blur	Hue	Exposure	Saturation	F/R	Gn	SAT	mAP	Precision	Recall	F1	F2	mAP	Precision	Recall	F1	F2	
								70.54%	85.80%	66.83%	75.14%	69.92%	86.86%	81.87%	83.90%	82.87%	83.49%	0.25
✓								73.68%	77.20%	71.63%	74.31	72.68%	88.58%	82.96%	85.91%	84.41%	85.31%	0.25
✓	✓							75.67%	66.81%	75.48%	70.88%	73.57%	88.87%	85.09%	87.46%	86.26%	86.98%	0.25
✓	✓	✓	✓	✓				76.07%	63.28%	77.88%	69.83%	74.45%	90.36%	82.33%	88.70%	85.39%	87.35%	0.25
✓		✓	✓	✓				78.92%	79.49%	74.52%	76.92%	75.46%	90.87	84.98%	87.62%	86.28%	87.08%	0.35
✓		✓	✓	✓	✓			**81.04%**	**80.53%**	**73.56%**	**76.88%**	**74.85%**	**91.43%**	**87.46%**	**87.46%**	**87.46%**	**87.46%**	**0.35**
✓		✓	✓	✓	✓	✓		80.67%	79.06%	72.60%	75.69%	73.80%	90.65%	89.60%	86.69%	88.12%	87.25%	0.35
✓		✓	✓	✓	✓		✓	79.94%	81.15%	74.52%	77.69%	75.66%	90.58%	88.73%	85.29%	86.98%	85.96%	0.25

5.2.5 Regularization and other methods

This subsection describes three experiments conducted. Firstly we use k-mean clustering with 9 clusters to calculate anchor box priors using our training dataset similarly to what the authors of YOLO did, and as described in the chapter 3.1.3.A. To calculate k-mean clustering, DarkMark [77] was used. Since this led to worse results, we removed the custom anchor boxes, and we replaced the regular batch normalization in every layer that used it by CBN, which improved the model's performance. Finally we re-tested the Adam optimizer using regular batch normalization, which after adjusting the confidence threshold achieved even better results than those obtained using CBN with stochastic gradient descent. The results are shown in table 5.7.

Table 5.7 All the experiments conducted using CBN, custom anchor boxes, and Adam optimizer. The results that performed best are highlighted and used in the next set of experiments. Using Adam optimizer together with CBN is not possible.

Techniques			Datasets										Confidence threshold
			Etis-Larib					CVC-ColonDB					
Custom Anchor boxes	CBN	Adam optimizer	mAP	Precision	Recall	F1	F2	mAP	Precision	Recall	F1	F2	
✓			78.65%	81.46%	67.44%	73.79%	69.85%	91.07%	88.89%	86.69%	87.77%	87.12%	0.35
	✓		81.71%	82.35%	74.04%	77.97%	75.56%	91.72%	88.52%	85.91%	87.20%	86.42%	0.35
		✓	**82.93%**	**81.44%**	**75.96%**	**78.61%**	**77.00%**	**90.96%**	**88.65%**	**87.62%**	**88.23%**	**87.86%**	**0.30**

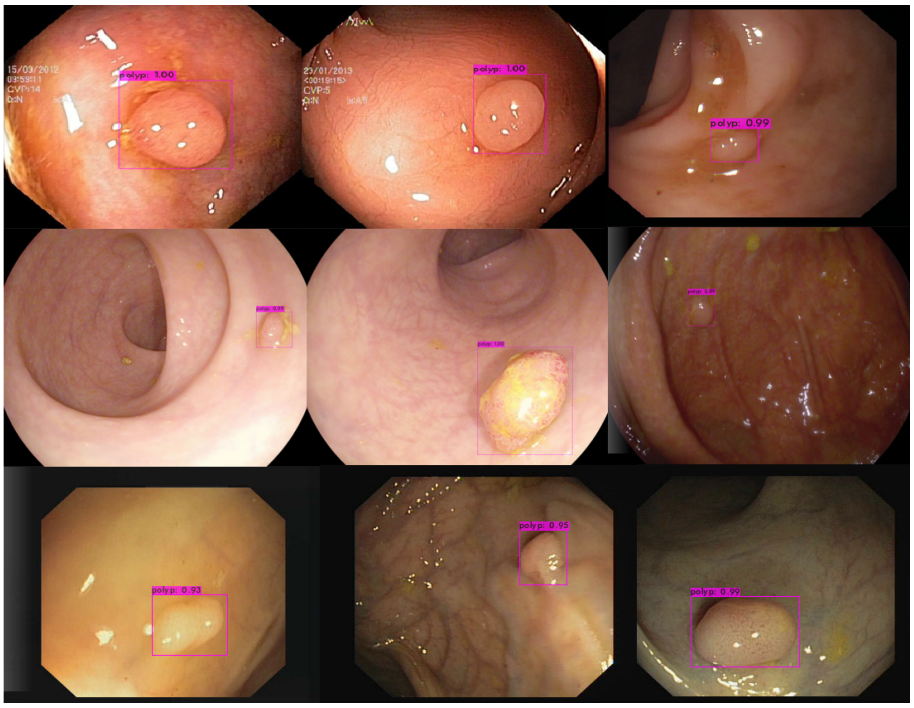

Figure 5.4: Examples of inferences performed by the YOLOv4 model using Darknet. The top row consists of frames that belong to the testing dataset. The middle row frames belong to the Etis-Larib validation dataset, and the bottom row frames are from the CVC-ClinicDB validation dataset.

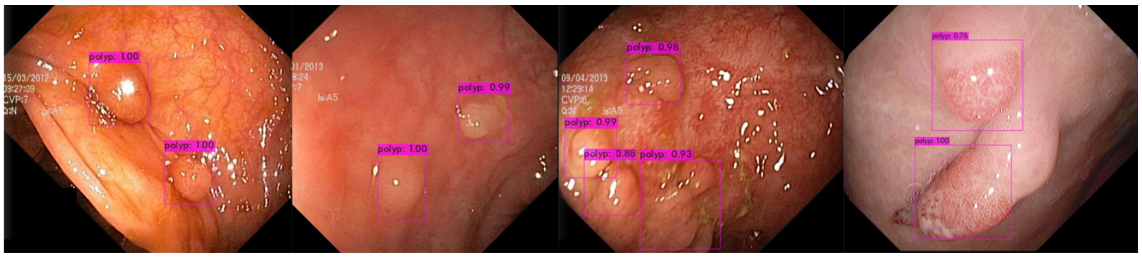

Figure 5.5: Examples of inferences performed on the training dataset by the YOLOv4 model using Darknet. These frames contain multiple polyps.

Figure 5.4 exhibits examples of some inferences performed on the testing dataset and on the two validation dataset, Etis-Larib, and CVC-ClinicDB. Figure 5.5 exhibits examples with multiple polyps of some inferences performed on the training dataset.

5.2.6 Final experiments to assess inference speed

The final results for this portion of the experiments in terms of accuracy are again shown in table 5.8.

Table 5.8 Best results achieved by the YOLOv4 model in the validation datasets, Etis-Larib and CVC-ClinicDB.

Datasets										Confidence threshold
Etis-Larib					CVC-ColonDB					
mAP	Precision	Recall	F1	F2	mAP	Precision	Recall	F1	F2	
82.93%	81.44%	75.96%	78.61%	77.00%	90.96%	88.65%	87.62%	88.23%	87.86%	0.30

We can refer to table 2.5 to compare our accuracy metrics with the literature. For the 2021 study of Pacal et al. (section 2.2.5) the results obtained for Etis-Larib using the ”off-the-shelf” YOLOv4 model were: 81.78% for precision, 79.25% for recall and 80.50% for F1 score.

To perform inference on videos, Darknet allows the user to perform inference using the video in a benchmark mode, which does not account for the overhead of drawing boxes, applying NMS, and displaying the video on the screen. Using the benchmark mode on the 3 videos we achieved an average of 131 FPS. However, as it was intended to test the video in a scenario closer to the real world, the inference was run while capturing the video, drawing the boxes and, applying NMS. These results are shown in table 5.9. We present the obtained results using the speed validation dataset constructed as described in section 4.4.2.

Table 5.9 Average frame rate obtained for each of the three selected videos from Colonoscopic Dataset. We ran each video 5 times and took the average of the average frame rate in each of the three videos. The GPU used was an NVIDIA RTX 2080TI.

Video name	Average frame rate in FPS
Adenoma.mp4	98.64
Hyperplastic.mp4	97.10
Serrated.mp4	97.14

We can refer to table 2.6 to compare the inference speed results from table 5.9 with the literature. For the 2021 study of Pacal et al. (section 2.2.5) the results obtained for the ”off-the-shelf” YOLOv4 model were: 77 FPS on an NVIDIA RTX 2080TI. It is worth noting that the number of FPS achieved depend on the size of the input images, the overhead introduced by the re-sizing of the images, the overhead of drawing boxes, applying NMS and displaying the video on the screen.

5.3 Summary

This chapter describes the experiments conducted to obtain good results in terms of precision. We begin by choosing the metrics to evaluate the model’s performance. We conduct experiments to assess the values for the hyper-parameters, image input size, pre-processing step, experiment with other optimizers. Also in the experiments we test different data augmentation methods, try different regularization methods and other techniques, and finally conduct the final experiments to set up a proper comparison between the pre-quantized and post-quantized model. In the next chapter, we conduct the experiments to quantize the model and evaluate it.

6

Deploying the model in TensorRT

Contents

This chapter describes the experiments performed in order to obtain a quantized version of the proposed neural network model. The framework used for this task is NVIDIA TensorRT, which allows for quantization and optimization of the model. For quantization, 16-bit floating point (FP16) precision and 8-bit integer (INT8) precision are adopted. We identify the process used to translate the model from the initial Darknet format to the final TensorRT format. We then evaluate the quantized version of the model in the validation datasets and compare results against the pre-quantized version.

6.1 ONNX and TensorRT translation

In order to deploy the model in TensorRT, the Github repository [83] was used. In order to deploy the Darknet model into TensorRT, the model must be translated to the ONNX format. ONNX is an open format that permits the representation of deep learning models, allowing the developers to use the models in ONNX format in different frameworks, tools, and compilers such as TensorRT.

We use the repository [83] first to translate the model to ONNX and then convert the ONNX model to TensorRT. Using the TensorRT model, we can perform inference on the chosen videos and images to then obtain the necessary evaluation metrics for the model.

TensorRT runs a set of optimizations as described in section 3.3. We opted to use FP16 and INT8 precision for quantization as we observed an increase in speed performance with significant degradation of detection performance. In the next section, the results between the Darknet model and the TensorRT model are compared.

For INT8 quantization, a calibration dataset must be provided. The goal of the calibration step, as described in section 2.3.4 is to acquire statistics to find the optimal scale for each activation. TensorRT uses symmetric quantization.

We chose to use 600 images from our training dataset as it led to better results than 800 images or the whole training dataset (1040 images). We also tested using CVC-ClinicDB as the calibration dataset, but this led to worse results than using the 600 images from the training dataset.

6.2 Final results

Similar to what we did using Darknet, we evaluate the model's performance in the validation datasets Etis-Larib, CVC-ClinicDB for accuracy assessment, and three videos from Colonoscopic Database for inference speed evaluation. The results for accuracy evaluation follow in table 6.1.

For some reason we are unaware of, it was not possible to properly apply the calibration step when using INT8 quantization on the model that obtained the best results, which uses the Adam optimizer. Using an appropriate number of images for the calibration dataset would cause the program to fail during the calibration step, and not using an appropriate number of images for calibration can severely impact the results. For this reason, it was decided to use the next best model i.e., up to the point where Cross-Iteration Batch Normalization (CBN) with stochastic gradient descent was used. The results for accuracy evaluation

follow in table 6.1.

As we can observe in table 6.1, for the FP16 model a decrease of 2.56% in median avarage precision (mAP), 6.86% in precision, 3.21% in F1 and 0.84% in F2 was observed while recall stayed the same for the Etis-Larib dataset. For CVC-ClinicDB, a decrease of 1.45% in mAP, 0.13% in precision, was registered while recall improved by 0.16%, F1 by 0.01% and F2 continued the same. The improvements registered in CVC-ClinicDB could be due to the precision-recall trade-off as F1 and F2 remained the same if we attribute the 0.01% increase in F1 to rounding error.

Making use of the table 6.1 we can compare the FP16 model with the INT8 model, which for the Etis-Larib dataset we observed an increase of 0.92% in mAP, 3.19% in precision, 0.48% in recall, 1.78% in F1 and 0.6% in F2. For the CVC-ClinicDB we register an increase of 0.15% in mAP, and a decrease of 0.05% in precision 0.47% in recall, 0.26% in F1, and 0.38% in F2. The increases registered in all metrics for the Etis-Larib dataset and in mAP for the CVC-ClinicDB dataset can be attributed to a regularization effect that occurs when casting from FP16 to INT8, which prevents overfitting of the model. This phenomenon is observed in the literature and is documented and mathematically formulated in [84].

Table 6.1 Comparison of the results obtained with the TensorRT model (FP16 and INT8 precision) and the Darknet model (32-bit floating point (FP32) precision). The model deployed in both frameworks uses CBN with stochastic gradient descent optimizer. The results were obtained for the Etis-Larib validation dataset and the CVC-ClinicDB validation dataset.

Bit-precision	Datasets										Confidence threshold
	Etis-Larib					CVC-ClinicDB					
	mAP	Precision	Recall	F1	F2	mAP	Precision	Recall	F1	F2	
FP32	81.71%	82.35%	74.04%	77.97%	75.56%	91.72%	88.52%	85.91%	87.20%	86.42%	0.35
FP16	79.15%	75.49%	74.04%	74.76%	74.72%	90.27%	88.39%	86.07%	87.21%	86.52%	0.35
INT8	80.07%	78.68%	74.52%	76.54%	75.32%	90.42%	88.34%	85.60%	86.95%	86.14%	0.35

Figure 6.1 exhibits four examples of inferences performed using the quantized TensorRT models at FP16 and INT8 precision. For these two cases and many other we observed very few or none visual differences between the FP16 and the INT8 models.

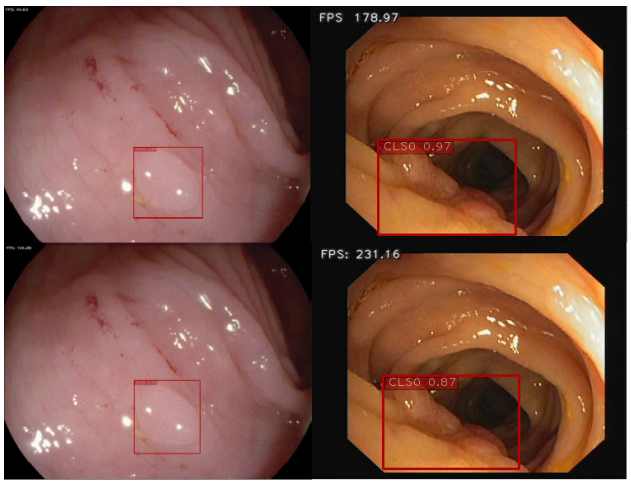

Figure 6.1: Examples of inferences performed on Etis-Larib (left-hand side images) and CVC-ClinicDB (right-hand side images) using the TensorRT FP16 model (top row images) and the TensorRT INT8 model (bottom row images).

Figure 6.2 presents the precision-recall curves for the Darknet FP32 model, TensorRT FP16 model, and TensorRT INT8 model on the Etis-Larib validation dataset and CVC-ClinicDB validation dataset. The area under each of the curves represents the mAP metric for each of the models, that as we can see presents a slight decrease for INT8 and FP16 when compared with FP32 for both datasets.

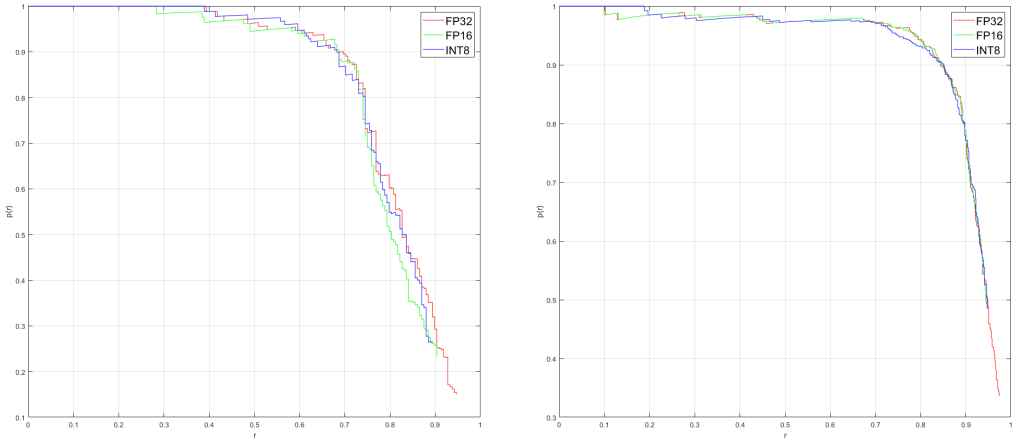

Figure 6.2: Precision-recall curves for the Darknet FP32 model, TensorRT FP16 model, and TensorRT INT8 model on the Etis-Larib validation dataset (left-hand side) and CVC-ClinicDB validation dataset (right-hand side). The letter r stands for recall and $p(r)$ is precision in function of the recall.

Finally, the inference speed results are displayed in table 6.2, for the Darknet FP32 model, the TensorRT FP16 model, and the TensorRT INT8 model. As observed in this table, the number of processed frames per second increases significantly as we move from FP32 down to FP16 quantization levels. For the case of the 'Hyperplastic.mp4' colonoscopy video, decreasing the precision from 32 to 16-bit FP increases throughput performance from 97.10 to 172.18 frames per second (FPS). Moving down from FP16 to INT8 precision, we see a slight increase in the frame rate, with the highest increase being of 13.02 FPS for the 'Adenoma.mp4' colonoscopy video.

Table 6.2 Average frame rate obtained for each of the three selected videos from Colonoscopic Dataset both in the Darknet FP32 model and in the TensorRT model (FP16 and INT8 precision).

	Average frame rate in FPS		
Video name	FP32	FP16	INT8
Adenoma.mp4	98.64	157.39	170.41
Hyperplastic.mp4	97.10	172.18	176.22
Serrated.mp4	97.14	159.85	167.81

6.3 Summary

This chapter describes the experiments conducted to obtain a quantized FP16 and INT8 version of the "original" model. We begin by explaining how the model was deployed in another framework named TensorRT. For this, the model must be translated to ONNX format and then converted to TensorRT. The TensorRT models use FP16 and INT8 precision.

7

Conclusions

Contents

The use of single-stage object detectors such as You Only Look Once (YOLO)v4 allow a solution to perform polyp detection in real-time using general purpose graphics processing units (GPGPUs). Even though the initial accuracy might be lower, different techniques of bag of freebies and bag of specials can be applied to the model to increase accuracy.

Applying optimization techniques, particularly quantization techniques, helped speed up the model's inference with minimal loss in accuracy metrics. To further increase the inference speed, and to make a convolutional neural network (CNN) model smaller, thus being able to fit the model in a smaller and less power-hungry device, more optimization techniques and lower precision quantization can be deployed at the cost of a slight loss in precision, such as using quantization bellow 8-bit integer (INT8). A barrier exists at 4-bit integer (INT4) quantization, which requires re-training the model at a lower precision in the current state of the art, which cannot be optimal when using more complex detectors.

We observed that using INT8 quantization can sometimes offer better results than 16-bit floating point (FP16) quantization, so it should always be explored if enough data is available to construct a calibration dataset. As stated in the literature, this increase in performance happens due to a regularization effect from lowering the bit width precision.

Data augmentation tactics are crucial for solving problems where labeled data is a scarce recourse, as it happens in the case of polyp detection problems, due to privacy reasons or lack of qualified human resources to perform the labeling. Selecting the correct data augmentation tactics requires analysis and interpretation of the results observed.

7.1 Future work

The results obtained are positive and very encouraging. However, polyp detection still poses a few problems to be tackled in the future:

- Explore more techniques for data augmentation, regularization, and the usage of genetic algorithms to optimize the parameters and hyper-parameters fine-tuning;
- Explore lower-levels of quantization to facilitate real-time inference on less power-hungry devices;
- Explore lower-levels of quantization to facilitate the deployment of larger models in smaller and more constrained graphics processing units (GPUs);
- The lack of training data is still a problem that remains unsolved.

Bibliography

[1] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You only look once: Unified, real-time object detection," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 779–788. [Online]. Available: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html

[2] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2017, pp. 7263–7271.

[3] Q.-C. Mao, H.-M. Sun, Y.-B. Liu, and R.-S. Jia, "Mini-yolov3: real-time object detector for embedded applications," *IEEE Access*, vol. 7, pp. 133 529–133 538, 2019.

[4] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[5] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934

[6] S. Kim and H. Kim, "Zero-centered fixed-point quantization with iterative retraining for deep convolutional neural network-based object detectors," *IEEE Access*, vol. 9, pp. 20 828–20 839, 2021.

[7] A. Nogueira-Rodríguez, R. Dominguez-Carbajales, H. López-Fernández, A. Iglesias, J. Cubiella, F. Fdez-Riverola, M. Reboiro-Jato, and D. Glez-Pena, "Deep neural networks approaches for detecting and classifying colorectal polyps," *Neurocomputing*, vol. 423, pp. 721–734, 2021. [Online]. Available: https://doi.org/10.1016/j.neucom.2020.02.123

[8] O. F. Ahmad, P. Brandao, S. S. Sami, E. Mazomenos, A. Rau, R. Haidry, R. Vega, E. Seward, T. K. Vercauteren, D. Stoyanov *et al.*, "Tu1991 artificial intelligence for real-time polyp localisation in colonoscopy withdrawal videos," *Gastrointestinal Endoscopy*, vol. 89, no. 6, p. AB647, 2019. [Online]. Available: https://doi.org/10.1016/j.gie.2019.03.1135

[9] Y. Xi and P. Xu, "Global colorectal cancer burden in 2020 and projections to 2040," *Translational Oncology*, vol. 14, no. 10, p. 101174, 2021.

[10] P. Rawla, T. Sunkara, and A. Barsouk, "Epidemiology of colorectal cancer: incidence, mortality, survival, and risk factors," *Przeglad gastroenterologiczny*, vol. 14, no. 2, pp. 89–103., 2019. [Online]. Available: https://doi.org/10.5114/pg.2018.81072

[11] I. A. Issa and M. Noureddine, "Colorectal cancer screening: An updated review of the available options," *World journal of gastroenterology*, vol. 23, no. 28, p. 5086–5096., 2017. [Online]. Available: https://doi.org/10.3748/wjg.v23.i28.5086

[12] B. Hurt and C. C. Barnett, "Chapter 55 - colorectal polyps," in *Abernathy's Surgical Secrets (Seventh Edition)*, seventh edition ed., A. H. Harken and E. E. Moore, Eds. Elsevier, 2018, pp. 240–243. [Online]. Available: https://doi.org/10.1016/B978-0-323-47873-1.00055-3

[13] H. Singh, D. Turner, L. Xue, L. E. Targownik, and C. N. Bernstein, "Risk of developing colorectal cancer following a negative colonoscopy examination: evidence for a 10-year interval between colonoscopies," *Jama*, vol. 295, no. 20, pp. 2366–2373, 2006. [Online]. Available: https://doi.org/10.1001/jama.295.20.2366

[14] D. A. Corley, C. D. Jensen, W. K. Marks, A. R. ad Zhao, J. K. Lee, C. A. Doubeni, A. G. Zauber, J. de Boer, B. H. Fireman, J. E. Schottinger, V. P. Quinn, N. R. Ghai, T. R. Levin, and C. P. Quesenberry, "Adenoma detection rate and risk of colorectal cancer and death," *The New England journal of medicine*, vol. 370, no. 14, p. 1298–1306., 2014. [Online]. Available: https://doi.org/10.1056/NEJMoa1309086

[15] C. M. C. le Clercq, M. W. E. Bouwens, E. J. A. Rondagh, C. M. Bakker, E. T. P. Keulen, R. J. de Ridder, B. Winkens, A. A. M. Masclee, and S. Sanduleanu, "Postcolonoscopy colorectal cancers are preventable: a population-based study," *Gut*, vol. 63, no. 6, pp. 957–963, 2014. [Online]. Available: doi.org/10.1136/gutjnl-2013-304880

[16] T. Lui and W. K. Leung, "Is artificial intelligence the final answer to missed polyps in colonoscopy?" *World journal of gastroenterology*, vol. 26, no. 35, p. 95248–5255, 2020. [Online]. Available: https://doi.org/10.3748/wjg.v26.i35.5248

[17] G. Tziatzios, P. Gkolfakis, L. D. Lazaridis, A. Facciorusso, G. Antonelli, C. Hassan, A. Repici, P. Sharma, D. K. Rex, and K. Triantafyllou, "High-definition colonoscopy for improving adenoma detection: a systematic review and meta-analysis of randomized controlled studies," *Gastrointestinal endoscopy*, vol. 91, no. 5, pp. 1027–1036, 2020.

[18] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein *et al.*, "Imagenet large scale visual recognition challenge," *International journal of computer vision*, vol. 115, no. 3, pp. 211–252, 2015. [Online]. Available: http://arxiv.org/abs/1409.0575

[19] V. Sharma and R. N. Mir, "A comprehensive and systematic look up into deep learning based object detection techniques: A review," *Computer Science Review*, vol. 38, p. 100301, 2020. [Online]. Available: https://doi.org/10.1016/j.cosrev.2020.100301

[20] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014, pp. 580–587. [Online]. Available: https://arxiv.org/pdf/1311.2524.pdf

[21] R. Girshick, "Fast r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 1440–1448. [Online]. Available: https://arxiv.org/abs/1504.08083

[22] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," *arXiv preprint arXiv:1506.01497*, 2015. [Online]. Available: https://arxiv.org/abs/1506.01497

[23] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, 2017, pp. 2961–2969. [Online]. Available: https://arxiv.org/abs/1703.06870

[24] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015. [Online]. Available: https://doi.org/10.1109/TPAMI.2015.2389824

[25] Y. Liu, P. Sun, N. Wergeles, and Y. Shang, "A survey and performance evaluation of deep learning methods for small object detection," *Expert Systems with Applications*, p. 114602, 2021. [Online]. Available: https://doi.org/10.1016/j.eswa.2021.114602

[26] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement," *arXiv preprint arXiv:1804.02767*, 2018. [Online]. Available: https://arxiv.org/abs/1804.02767v1

[27] N. Tajbakhsh, S. R. Gurudu, and J. Liang, "Automatic polyp detection using global geometric constraints and local intensity variation patterns," in *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer, 2014, pp. 179–187. [Online]. Available: https://doi.org/10.1007/978-3-319-10470-6_23

[28] ——, "Automatic polyp detection in colonoscopy videos using an ensemble of convolutional neural networks," in *2015 IEEE 12th International Symposium on Biomedical Imaging (ISBI)*. IEEE, 2015, pp. 79–83. [Online]. Available: https://doi.org/10.1109/ISBI.2015.7163821

[29] Y. Zheng, R. Zhang, R. Yu, Y. Jiang, T. W. Mak, S. H. Wong, J. Y. Lau, and C. C. Poon, "Localisation of colorectal polyps by convolutional neural network features learnt from white light and narrow band endoscopic images of multiple databases," in *2018 40th annual international conference of the IEEE engineering in medicine and biology society (EMBC)*. IEEE, 2018, pp. 4142–4145. [Online]. Available: https://doi.org/10.1109/EMBC.2018.8513337

[30] G. Urban, P. Tripathi, T. Alkayali, M. Mittal, F. Jalali, W. Karnes, and P. Baldi, "Deep learning localizes and identifies polyps in real time with 96% accuracy in screening colonoscopy," *Gastroenterology*, vol. 155, no. 4, pp. 1069–1078, 2018. [Online]. Available: https://doi.org/10.1053/j.gastro.2018.06.037

[31] X. Zhang, F. Chen, T. Yu, J. An, Z. Huang, J. Liu, W. Hu, L. Wang, H. Duan, and J. Si, "Real-time gastric polyp detection using convolutional neural networks," *PloS one*, vol. 14, no. 3, p. e0214133, 2019. [Online]. Available: https://doi.org/10.1371/journal.pone.0214133

[32] J. Y. Lee, J. Jeong, E. M. Song, C. Ha, H. J. Lee, J. E. Koo, D.-H. Yang, N. Kim, and J.-S. Byeon, "Real-time detection of colon polyps during colonoscopy using deep learning: systematic validation with four independent datasets," *Scientific reports*, vol. 10, no. 1, pp. 1–9, 2020. [Online]. Available: https://doi.org/10.1038/s41598-020-65387-1

[33] H. A. Qadir, Y. Shin, J. Solhusvik, J. Bergsland, L. Aabakken, and I. Balasingham, "Toward real-time polyp detection using fully cnns for 2d gaussian shapes prediction," *Medical Image Analysis*, vol. 68, p. 101897, 2021. [Online]. Available: https://doi.org/10.1016/j.media.2020.101897

[34] R. Koprowski, "Overview of technical solutions and assessment of clinical usefulness of capsule endoscopy," *Biomedical engineering online*, vol. 14, no. 1, pp. 1–23, 2015. [Online]. Available: https://doi.org/10.1186/s12938-015-0108-3

[35] J. Bernal, F. J. Sánchez, G. Fernández-Esparrach, D. Gil, C. Rodríguez, and F. Vilariño, "Wm-dova maps for accurate polyp highlighting in colonoscopy: Validation vs. saliency maps from physicians," *Computerized Medical Imaging and Graphics*, vol. 43, pp. 99–111, 2015. [Online]. Available: https://doi.org/10.1016/j.compmedimag.2015.02.007

[36] J. Bernal, J. Sánchez, and F. Vilarino, "Towards automatic polyp detection with a polyp appearance model," *Pattern Recognition*, vol. 45, no. 9, pp. 3166–3182, 2012. [Online]. Available: https://doi.org/10.1016/j.patcog.2012.03.002

[37] D. Vázquez, J. Bernal, F. J. Sánchez, G. Fernández-Esparrach, A. M. López, A. Romero, M. Drozdzal, and A. Courville, "A benchmark for endoluminal scene segmentation of colonoscopy images," *Journal of healthcare engineering*, vol. 2017, 2017. [Online]. Available: https://doi.org/10.1155/2017/4037190

[38] J. Silva, A. Histace, O. Romain, X. Dray, and B. Granado, "Toward embedded detection of polyps in wce images for early diagnosis of colorectal cancer," *International journal of computer assisted radiology and surgery*, vol. 9, no. 2, pp. 283–293, 2014. [Online]. Available: https://doi.org/10.1007/s11548-013-0926-3

[39] K. Pogorelov, K. R. Randel, C. Griwodz, S. L. Eskeland, T. de Lange, D. Johansen, C. Spampinato, D.-T. Dang-Nguyen, M. Lux, P. T. Schmidt *et al.*, "Kvasir: A multi-class image dataset for computer

aided gastrointestinal disease detection," in *Proceedings of the 8th ACM on Multimedia Systems Conference*, 2017, pp. 164–169. [Online]. Available: https://doi.org/10.1145/3193289

[40] N. Tajbakhsh, S. R. Gurudu, and J. Liang, "Automated polyp detection in colonoscopy videos using shape and context information," *IEEE transactions on medical imaging*, vol. 35, no. 2, pp. 630–644, 2015. [Online]. Available: https://doi.org/10.1109/TMI.2015.2487997

[41] Q. Angermann, J. Bernal, C. Sánchez-Montes, M. Hammami, G. Fernández-Esparrach, X. Dray, O. Romain, F. J. Sánchez, and A. Histace, "Towards real-time polyp detection in colonoscopy videos: Adapting still frame-based methodologies for video sequences analysis," in *Computer Assisted and Robotic Endoscopy and Clinical Image-Based Procedures*. Springer, 2017, pp. 29–41. [Online]. Available: https://doi.org/10.1007/978-3-319-67543-5_3

[42] P. Mesejo, D. Pizarro, A. Abergel, O. Rouquette, S. Beorchia, L. Poincloux, and A. Bartoli, "Computer-aided classification of gastrointestinal lesions in regular colonoscopy," *IEEE transactions on medical imaging*, vol. 35, no. 9, pp. 2051–2063, 2016. [Online]. Available: https://doi.org/10.1109/TMI.2016.2547947

[43] L. F. Sánchez-Peralta, J. B. Pagador, A. Picón, Á. J. Calderón, F. Polo, N. Andraka, R. Bilbao, B. Glover, C. L. Saratxaga, and F. M. Sánchez-Margallo, "Piccolo white-light and narrow-band imaging colonoscopic dataset: A performance comparative of models and datasets," *Applied Sciences*, vol. 10, no. 23, p. 8501, 2020. [Online]. Available: https://doi.org/10.3390/app10238501

[44] I. Pacal and D. Karaboga, "A robust real-time deep learning based automatic polyp detection system," *Computers in Biology and Medicine*, p. 104519, 2021.

[45] T. Liang, J. Glossner, L. Wang, S. Shi, and X. Zhang, "Pruning and quantization for deep neural network acceleration: A survey," *Neurocomputing*, vol. 461, pp. 370–403, 2021.

[46] A. Gholami, S. Kim, Z. Dong, Z. Yao, M. W. Mahoney, and K. Keutzer, "A survey of quantization methods for efficient neural network inference," *arXiv preprint arXiv:2103.13630*, 2021.

[47] H. V. Habi, R. Peretz, E. Cohen, L. Dikstein, O. Dror, I. Diamant, R. H. Jennings, and A. Netzer, "Hptq: Hardware-friendly post training quantization," *arXiv preprint arXiv:2109.09113*, 2021.

[48] R. Banner, Y. Nahshan, E. Hoffer, and D. Soudry, "Post-training 4-bit quantization of convolution networks for rapid-deployment," *arXiv preprint arXiv:1810.05723*, 2018.

[49] Y. Choukroun, E. Kravchik, F. Yang, and P. Kisilev, "Low-bit quantization of neural networks for efficient inference." in *ICCV Workshops*, 2019, pp. 3009–3018.

[50] M. Nagel, R. A. Amjad, M. Van Baalen, C. Louizos, and T. Blankevoort, "Up or down? adaptive rounding for post-training quantization," in *International Conference on Machine Learning*. PMLR, 2020, pp. 7197–7206.

[51] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, and D. Soudry, "Improving post training neural quantization: Layer-wise calibration and integer programming," *arXiv preprint arXiv:2006.10518*, 2020.

[52] R. Ramakrishnan, A. K. Dev, A. Darshik, R. Chinchwadkar, and M. Purnaprajna, "Demystifying compression techniques in cnns: Cpu, gpu and fpga cross-platform analysis," in *2021 34th International Conference on VLSI Design and 2021 20th International Conference on Embedded Systems (VLSID)*. IEEE, 2021, pp. 240–245.

[53] Z.-Q. Zhao, P. Zheng, S.-t. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE transactions on neural networks and learning systems*, vol. 30, no. 11, pp. 3212–3232, 2019. [Online]. Available: https://doi.org/10.1109/TNNLS.2018.2876865

[54] J. Redmon, "Darknet: Open source neural networks in c," http://pjreddie.com/darknet/, 2013–2016.

[55] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *European conference on computer vision*. Springer, 2014, pp. 740–755.

[56] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 10 781–10 790.

[57] X. Long, K. Deng, G. Wang, Y. Zhang, Q. Dang, Y. Gao, H. Shen, J. Ren, S. Han, E. Ding *et al.*, "Pp-yolo: An effective and efficient implementation of object detector," *arXiv preprint arXiv:2007.12099*, 2020.

[58] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao, "Scaled-yolov4: Scaling cross stage partial network," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 13 029–13 038.

[59] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The PASCAL Visual Object Classes Challenge 2007 (VOC2007) Results," http://www.pascal-network.org/challenges/VOC/voc2007/workshop/index.html.

[60] ——, "The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results," http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html.

[61] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[62] P. Vora, B. Oza *et al.*, "A survey on k-mean clustering and particle swarm optimization," *International Journal of Science and Modern Engineering*, vol. 1, no. 3, pp. 24–26, 2013.

[63] G. Ghiasi, T.-Y. Lin, and Q. V. Le, "Dropblock: A regularization method for convolutional networks," *arXiv preprint arXiv:1810.12890*, 2018.

[64] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2818–2826.

[65] D. Misra, "Mish: A self regularized non-monotonic neural activation function," *arXiv preprint arXiv:1908.08681*, vol. 4, p. 2, 2019.

[66] C.-Y. Wang, H.-Y. M. Liao, Y.-H. Wu, P.-Y. Chen, J.-W. Hsieh, and I.-H. Yeh, "Cspnet: A new backbone that can enhance learning capability of cnn," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, 2020, pp. 390–391.

[67] Z. Yao, Y. Cao, S. Zheng, G. Huang, and S. Lin, "Cross-iteration batch normalization," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2021, pp. 12 331–12 340.

[68] Z. Zheng, P. Wang, W. Liu, J. Li, R. Ye, and D. Ren, "Distance-iou loss: Faster and better learning for bounding box regression," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 07, 2020, pp. 12 993–13 000.

[69] I. Loshchilov and F. Hutter, "Sgdr: Stochastic gradient descent with warm restarts," *arXiv preprint arXiv:1608.03983*, 2016.

[70] K. He, X. Zhang, S. Ren, and J. Sun, "Spatial pyramid pooling in deep convolutional networks for visual recognition," *IEEE transactions on pattern analysis and machine intelligence*, vol. 37, no. 9, pp. 1904–1916, 2015.

[71] S. Woo, J. Park, J.-Y. Lee, and I. S. Kweon, "Cbam: Convolutional block attention module," in *Proceedings of the European conference on computer vision (ECCV)*, 2018, pp. 3–19.

[72] S. Liu, L. Qi, H. Qin, J. Shi, and J. Jia, "Path aggregation network for instance segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[73] P. Mahto, P. Garg, P. Seth, and J. Panda, "Refining yolov4 for vehicle detection," *International Journal of Advanced Research in Engineering and Technology (IJARET)*, vol. 11, no. 5, 2020.

[74] "NVIDIA TensorRT," https://developer.nvidia.com/tensorrt, Accessed 30 September 2021.

[75] D. Korobchenko, "How to accelerate your neural net inference with TensorRT," https://www.youtube.com/watch?v=6My-daDk4zE, July 2018.

[76] J. Bai, F. Lu, K. Zhang *et al.*, "Onnx: Open neural network exchange," https://github.com/onnx/onnx, 2019.

[77] S. Charette, "Darkmark: Image markup for darknet machine learning." https://github.com/stephanecharette/DarkMark, 2019–2021.

[78] M. Everingham, L. Van Gool, C. K. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge," *International journal of computer vision*, vol. 88, no. 2, pp. 303–338, 2010.

[79] D. Stosic, "Training Neural Networks with Tensor Cores," https://www.youtube.com/watch?v=jF4-_ZK_tyc, August 2020.

[80] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Diou-darknet: Distance-iou loss into yolo v3," https://github.com/Zzh-tju/DIoU-darknet, 2019.

[81] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[82] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, pp. 1–48, 2019.

[83] J. Jung, "tensorrt_demos," https://github.com/jkjung-avt/tensorrt_demos, 2019.

[84] K. Zhao, S. Huang, P. Pan, Y. Li, Y. Zhang, Z. Gu, and Y. Xu, "Distribution adaptive int8 quantization for training cnns," in *Proceedings of the Thirty-Fifth AAAI Conference on Artificial Intelligence*, 2021.