ETH
Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Pipeline for automatic deep learning based detection of different cell types in 3D fluorescent microscopy images

Master Thesis

Ketan Gupta

August 31, 2024

Supervisor: Prof. Dr. B. Menze, Prof. Dr. E. Konukoglu

Advisor: P. Bueschl

Department of Information Technology and Electrical Engineering, ETH Zürich

Abstract

This thesis explores the application of advanced deep learning techniques to the detection and classification of hematopoietic stem cells (HSCs) and progenitor cells in 3D fluorescent microscopy images of murine bone marrow tissue. Four state-of-the-art models—NoduleNet, YOLOv5, Faster-RCNN, and RetinaNet—were implemented and evaluated to determine their effectiveness in accurately identifying and categorizing these cells. The results demonstrated varying levels of success, with RetinaNet showing the most promise in terms of recall and overall classification performance, while models like NoduleNet struggled due to architectural limitations and data constraints.

Despite these advancements, challenges such as overfitting, false positive rates, and difficulties in distinguishing closely related cell types persist. These findings underscore the need for further refinement of deep learning models tailored to the specific characteristics of 3D biological imaging. Future research should focus on expanding the dataset, optimizing model architectures, and incorporating biological context to enhance the accuracy and utility of these models in biomedical research.

This work contributes to the broader goal of automating and improving the accuracy of cell detection in complex biological images, thereby advancing our understanding of cellular behavior in the bone marrow microenvironment and its implications for aging, disease, and therapeutic interventions.

Acknowledgement

I would like to thank Prof. Dr. Menze for the opportunity to do this thesis at his group. His patience was a big encouragement and helped me tackle this thesis with a calmer mind. I would like to thank Prof. Dr. Konukoglu for his support I would also like to thank my advisor, Paul Bueschl, for taking the time to teach me about the dataset as well as for his support me me. I am deeply grateful for his extensive insight and help with this project. I could not have done it without his help. I would lastly like to thank the Menze Group for welcoming me with open arms. It was a very valuable period for me and I gained a lot of knowledge.

Contents

Chapter 1

Introduction

Hematopoietic stem cells (HSCs) play a critical role in the maintenance and regeneration of the blood system, residing predominantly in the bone marrow (BM) where they sustain hematopoiesis—the process of blood cell formation throughout an individual's life. HSCs possess the unique abilities of self-renewal and differentiation into various blood cell lineages, maintaining a delicate balance essential for blood homeostasis and immune response. This balance, however, is influenced by various intrinsic and extrinsic factors, including aging and pathological conditions such as cancer.

1.1 Background on Hematopoietic Stem Cells

HSCs are characterized by their rarity and high potency within the BM microenvironment. They exist at the apex of the hematopoietic hierarchy, giving rise to all blood cell types through a tightly regulated process of self-renewal and differentiation [1]. Under normal physiological conditions, HSCs maintain a quiescent state, minimizing cellular stress and potential DNA damage from active cycling [2]. This quiescence is critical for preserving the long-term regenerative capacity of HSCs.

As organisms age, the function and composition of HSCs undergo significant changes. Aging HSCs show an increased tendency toward myeloid lineage differentiation, often at the expense of lymphoid lineage production. This shift can lead to a higher incidence of myeloid leukemias and anemias in the elderly [3]. Additionally, aged HSCs exhibit altered self-renewal properties and a reduced overall regenerative potential, contributing to the decline in immune function and the increased prevalence of hematologic disorders in older adults [**?**].

1.2 The Role of Bone Marrow Microenvironment

The BM microenvironment, or niche, plays a pivotal role in regulating HSC behavior. It comprises various cell types, including stromal cells, endothelial cells, and niche-specific cells, all contributing to the maintenance and functional regulation of HSCs. The niche provides essential signals that influence HSC quiescence, activation, and differentiation [5]. Recent studies have highlighted the impact of the BM niche on HSC aging, suggesting that alterations in niche components, such as the loss of adrenergic innervation, can accelerate HSC aging and impair their regenerative capacity [4].

In pathological states, such as the presence of remote solid tumors, the BM niche undergoes dramatic remodeling. This remodeling can lead to the reprogramming of HSCs and progenitor cells, driving dysfunctional hematopoiesis distinct from the typical immune response

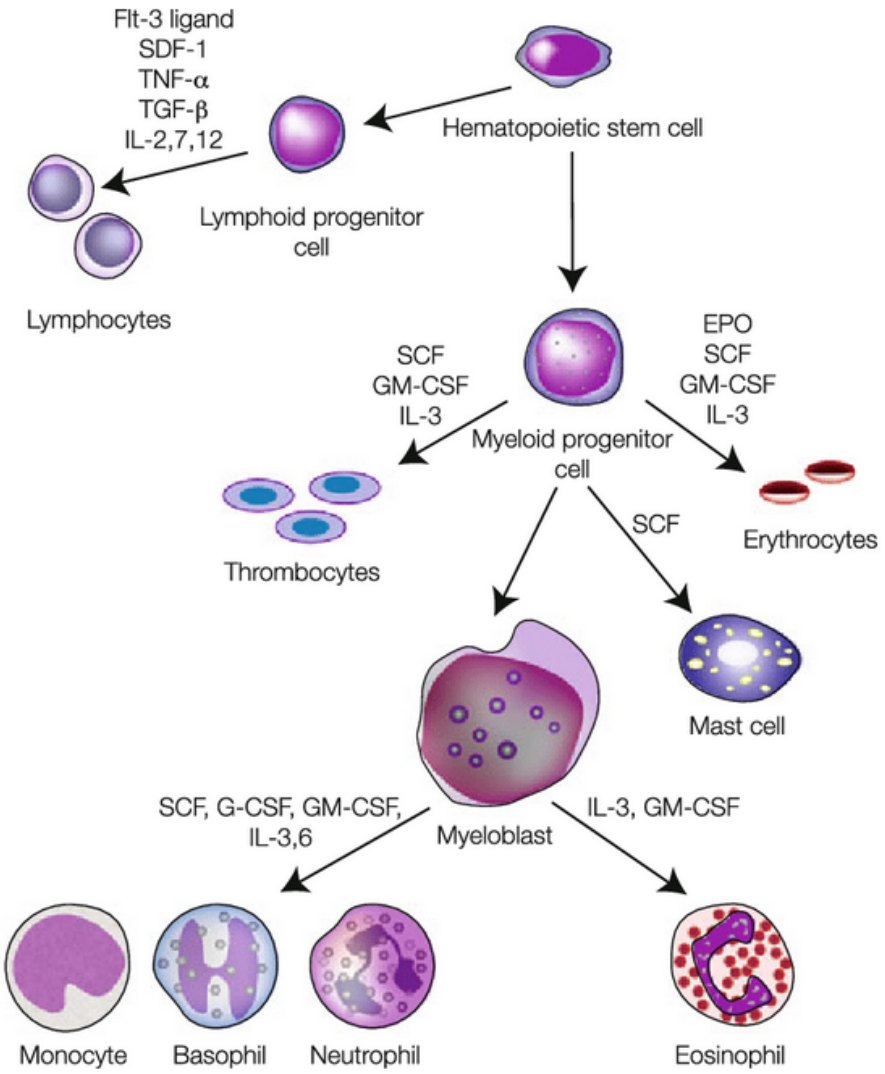

Figure 1.1: Schematic representation of hematopoiesis in which pluripotent stem and progenitor cells in the bone marrow divide and differentiate, passing through intermediate steps to form red blood cells, platelets, and white blood cells. The specific roles of hematopoietic cytokines known to stimulate transitions are indicated [4]

[5]. Understanding these interactions between HSCs and their niche is crucial for developing therapeutic strategies aimed at rejuvenating aged HSCs and improving outcomes in hematologic diseases. In order to facilitate this, microscopy data from bone marrow tissue must be analyzed so that cells may be located. This can be performed through object detection.

1.3 Object Detection

Object detection is a critical task in computer vision that involves identifying and locating objects within an image or a video. The primary goal of object detection is to draw bounding boxes around objects of interest and classify them into predefined categories [6]. This process enables machines to understand and interact with their environment by recognizing various objects, such as people, animals, vehicles, and everyday items. The desired output of object detection includes the coordinates of the bounding boxes surrounding each detected object and the corresponding class labels, which indicate the type of object detected.

In biological contexts, object detection extends to identifying cellular structures, detecting anomalies in medical scans, and analyzing complex tissues in microscopy images. Volumetric

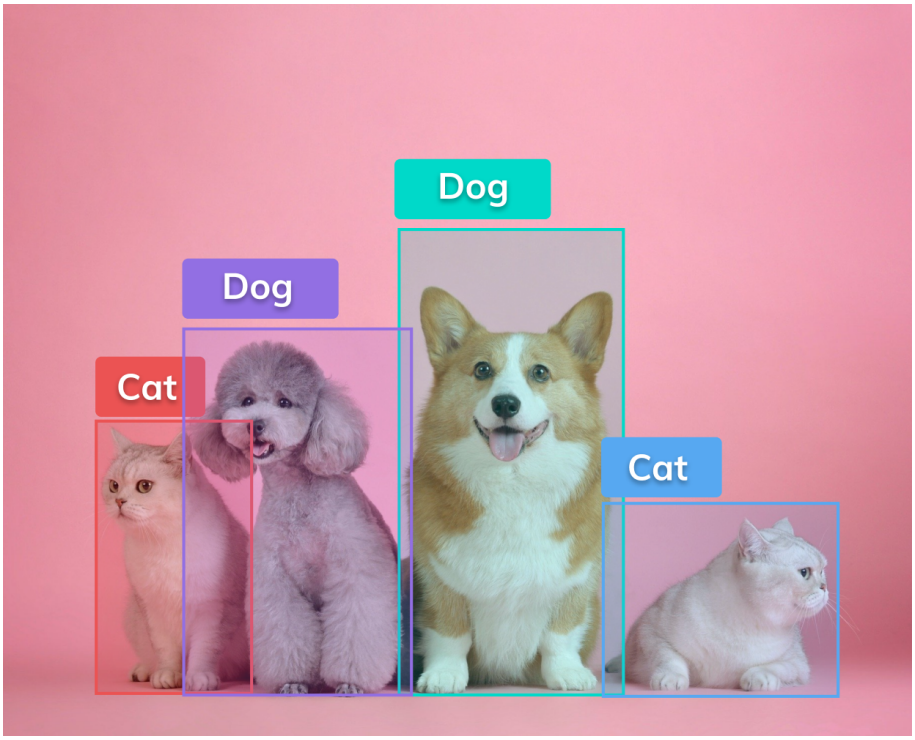

Figure 1.2: A simple example of object detection [6].

imaging, which provides three-dimensional representations of specimens, presents unique challenges and opportunities for object detection. The development of advanced algorithms and deep learning techniques has significantly enhanced the accuracy and efficiency of object detection in these complex datasets, making it an essential tool in modern biomedical research and diagnostics.

1.4 State-of-the-Art Object Detection Methods

Object detection in volumetric images, particularly within biological contexts, has advanced significantly with the advent of deep learning techniques. These methods enable the identification and localization of objects within three-dimensional data, providing critical insights into complex biological structures. This section provides an overview of current object detection techniques, key algorithms, recent advancements, and their application to microscopy images.

1.4.1 3D Convolutional Neural Networks

3D CNNs are a natural extension of traditional 2D CNNs, incorporating an additional dimension to process volumetric data. These networks apply 3D convolutional filters to capture spatial hierarchies in three dimensions, making them well-suited for detecting objects in volumetric images.

3D U-Net: An adaptation of the U-Net architecture for 3D data, 3D U-Net is extensively used for biomedical image segmentation. It features an encoder-decoder structure with skip connections, enabling precise localization of structures in volumetric images. This model has shown significant success in segmenting organs and detecting tumors in CT and MRI scans [7][8].

1.4.2 Region Proposal Networks (RPNs)

RPNs generate candidate regions (proposals) that are likely to contain objects, which are then classified and refined in subsequent stages.

Faster R-CNN: Faster R-CNN uses an RPN to quickly generate high-quality region proposals, which are then refined by a second network for object classification and bounding box regression. This architecture strikes a balance between speed and detection accuracy [9], making it suitable for tasks where both precision and performance are critical, such as identifying objects in complex scenes or detecting small objects in large images.

1.4.3 Single Stage Networks

The primary advantage of using a single-stage network is its speed [6]. Single-stage networks perform object detection in a single pass through the neural network, which means they directly predict class probabilities and bounding boxes from the input image without requiring an additional region proposal step. This makes them significantly faster and more suitable for real-time applications compared to two-stage networks, which first generate region proposals and then classify them. Additionally, single-stage networks often have a simpler architecture, which can result in lower computational requirements.

RetinaNet with Focal Loss: RetinaNet improves detection accuracy by focusing training on hard-to-classify examples [10]. It addresses the challenge of class imbalance in object detection by introducing the Focal Loss function [10]. This innovative loss function focuses training on hard-to-classify examples, improving detection performance, especially for small or less frequent objects.

YOLOv5: YOLOv5 is a highly efficient single-stage object detection model known for its speed and accuracy. It builds on the success of previous YOLO models by introducing enhancements in architecture [11] and training techniques, making it one of the most widely used models for real-time object detection in various applications. YOLOv5 is particularly effective in scenarios requiring quick detection with minimal computational resources [12], such as autonomous driving and video surveillance.

1.5 Aim of Thesis

Through this thesis in computational science, deep learning pipelines were implemented to detect progenitor and long-term HSCs in 3-dimensional light sheet fluorescence microscopic images of murine BM tissue. This pipeline outputs bounding boxes of the HSCs, facilitating the identification process for biologists without a computational background and eliminating the need for manual annotation. The training images, in the IMARIS (.ims) file format based on the hierarchical data format (HDF), were used to train and validate the deep learning models.

The César Nombela Arrieta lab acquired light sheet fluorescence microscopy images of murine BM tissues aged 2, 4, 6, and 8 weeks, targeting α-catulin and hepatic leukemia factor (HLF) proteins with GFP and tdTomato dyes, respectively. These dyes identify progenitor and long-term HSCs, with current annotations performed via the 'Spot' function on Imaris and manual curations. Automating this process using deep learning object detection techniques enhanced reliability, accuracy, and scalability, allowing for faster and more comprehensive analysis of larger datasets.

By advancing the methodology for detecting and mapping HSCs, this thesis contributes to the broader goal of understanding the cellular and molecular adaptations within the BM microenvironment that regulate HSC behavior. Insights gained from this research could

inform strategies for manipulating HSC activity in therapeutic contexts, ultimately improving the management of hematologic disorders and enhancing regenerative medicine approaches.

Chapter 2

Methods and Materials

This section outlines the methodologies used to detect hematopoietic stem cells in volumetric bone marrow tissue images from mice. The data was acquired using lightsheet fluorescence microscopy. The primary goal of this project was to apply advanced computer vision techniques for object detection.

2.1 Materials

2.1.1 Image Acquisition

Volumetric images of murine bone marrow tissues were provided by the César Nombela Arrieta (CNA) lab. These images were captured using lightsheet fluorescence microscopy. Fluorescence microscopy exploits the phenomenon fluorescence, whereby a fluorophore is excited from its ground state to an excited state when it absorbs electromagnetic radiation of a certain wavelength, loses a portion of its energy to the environment, and then emits electromagnetic radiation in order to return to the ground state. This fluorescence light can be acquired by a camera or observed by the human eye. The emitted light is of a lower energy than the excitation light, therefore it has a different wavelength according to Planck's Law. This process can be visualised using a Jablonski diagram (Figure 2.1). In biology, fluorescent dyes are often used as fluorophores as they bind to a ligand in the specimen. This ligand could be a particular cell type or protein among other things. Such dyes require a specific excitation wavelength and emit a specific fluorescent wavelength. When they are bound to a desired ligand in the specimen and fluorescence imaging is conducted, the fluorescent dyes enable this ligand to be observed through a microscope. The CNA lab targeted $/alpha$-catulin and hepatic leukemia factor (HLF) proteins with GFP and tdTomato dyes, respectively. These dyes identify progenitor and long-term HSCs. The images were acquired using a total of 6 fluorescent dyes, of which 3 are relevant: DAPI, GFP and Hlf-tdTomato.

Lightsheet fluorescence microscopy is a specific modality of fluorescence microscopy that enables optical sectioning and viewing of tissues with subcellular (of the order of micrometers) resolution [14]. This is accomplished by illuminating the specimen with a thin plane of light [14]. Because of this type of illumination, photobleaching and phototoxicity are minimized as compared to other fluorescence modalities including wide-field, confocal or multiphoton microscopy [14]. The light sheet is formed by a laser (solid state or gas) and is collimated and expanded with a beam expander. A cylindrical lens forms the light sheet (green beam), and it is projected through an illuminating objective. The focal point or the thinnest portion of the light sheet is positioned usually within the middle of the specimen chamber. The thin plane of excitation light is perpendicular to the z-axis, which is the axis that runs along the detecting objective. An optical filter is used so that only the respective fluorescent wavelength

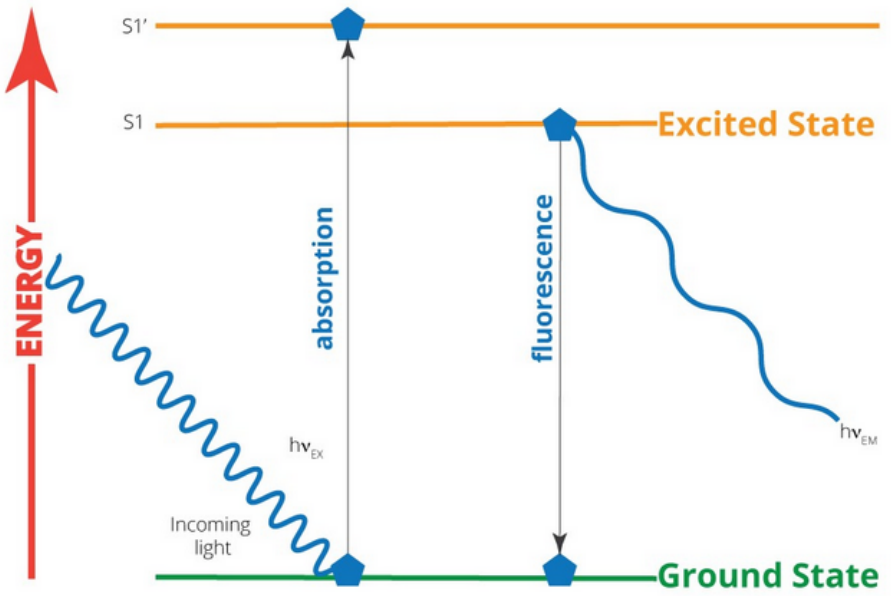

Figure 2.1: A Jablonski Diagram, showing the energy transitions involved in fluorescence [13].

of light for a particular dye is incident on the camera. In order to achieve optical sectioning, a computer-controlled motorized micropositioner is used to translate the specimen along the z-axis. This enabled 2D images along the XY plane to be collected for several z-values. This process is illustrated in Figure 2.2 below.

2.1.2 Dataset

The acquired tissue images were stored as IMARIS files [imaris citation], which could be read on Python using the package h5py. This package enables the reading and writing of Hierarchical Data Format (HDF) files. With a compatible structure as HDF files, the IMARIS files contained images for each fluorescent dye, the acquisition and image metadata, and the locations of HSCs. The image intensities for each channel or fluorescent dye were of the type 8-bit unsigned integer; they were constrained to intensity values within the range [0, 255].

The stored image contained a crop of the full specimen. This crop was characterized by two terms contained in the metadata: lim_{lower} and lim_{upper}. These represent the crop's lower and upper bounds respectively of the Cartesian coordinate values x, y, z in micrometers, The metadata also contained the cropped image's dimensions in voxels ($n_{voxels,axis}$). The resolution R (in $\mu m/voxel$) for a particular Cartesian axis was calculated by:

$$R_{\text{axis}} = \frac{\text{lim}_{\text{upper}} - \text{lim}_{\text{lower}}}{n_{\text{voxels,axis}}} \tag{2.1}$$

The HSCs were annotated manually by members of the CNA lab as spherical spots. For each annotated cell, the spots were stored with the format $[c_x, c_y, c_z, r]$ where c denotes the center coordinates of the spot r denotes the radius of the sphere. These four values are all in micrometers. The spots in the IMARIS file include spots which are outside the image crop limits. In order to convert the spots to bounding boxes in voxels, two steps were taken. First, spots that lie within the crop limits were selected. Then, the spot centers and their radii were converted from Cartesian distance in micrometers to voxels.

$$\text{c}_{\text{axis,vx}} = \frac{\text{c}_{\text{ax},\mu m} - \text{lim}_{\text{lower}}}{R_{\text{ax}}} \tag{2.2}$$

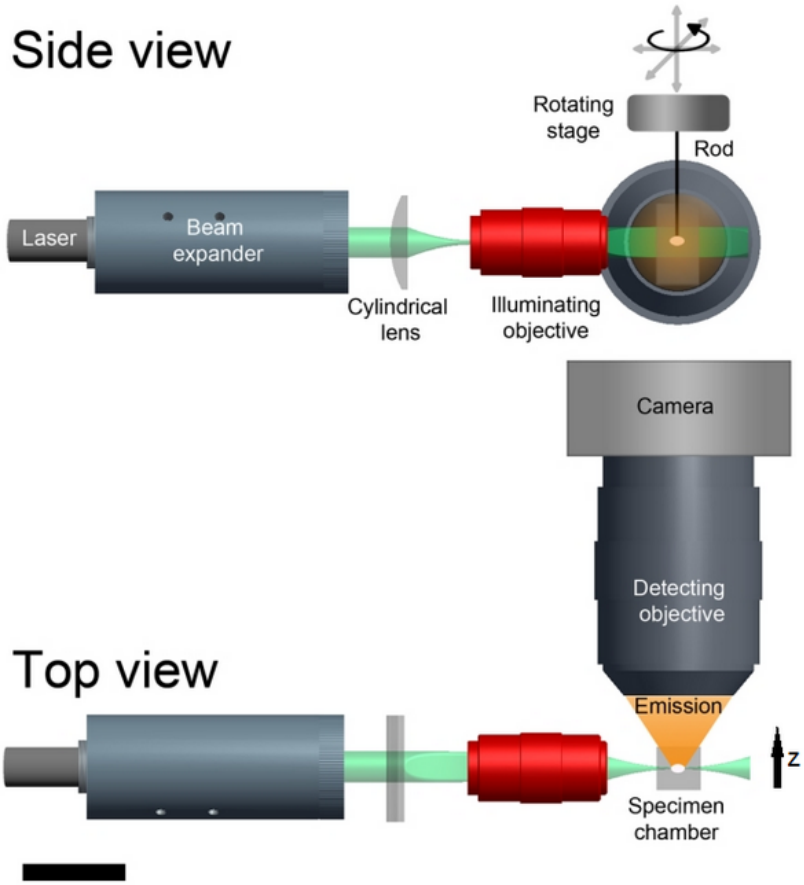

Figure 2.2: A diagram showing a side and top view of the basic components of a light sheet fluorescence microscope (LSFM) [14].

$$r_{\text{axis,vx}} = \frac{r_{\text{ax},\mu m}}{R_{\text{ax}}} \tag{2.3}$$

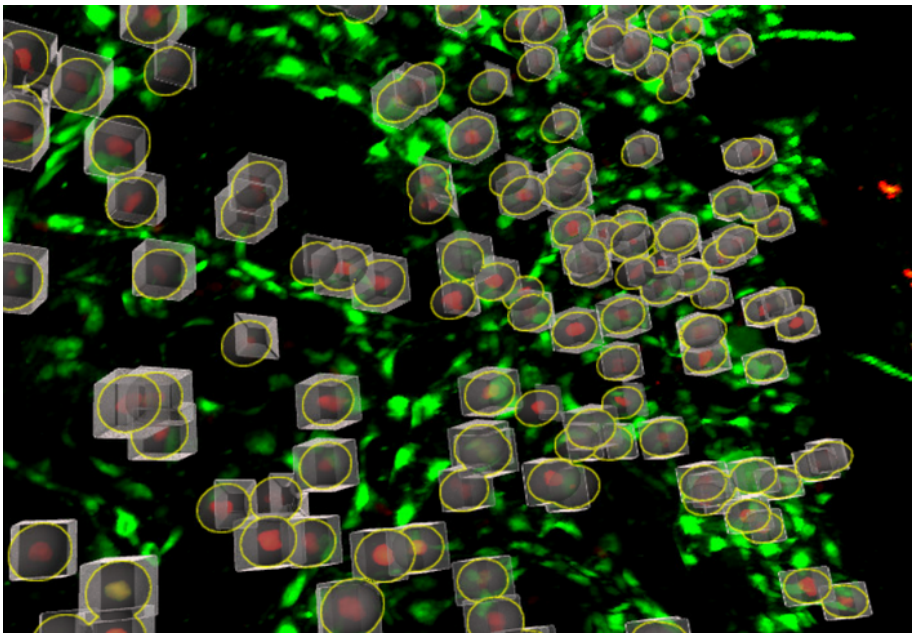

Figure 2.3: A section of an IMARIS file visualized, with spots and computed bounding boxes displayed.

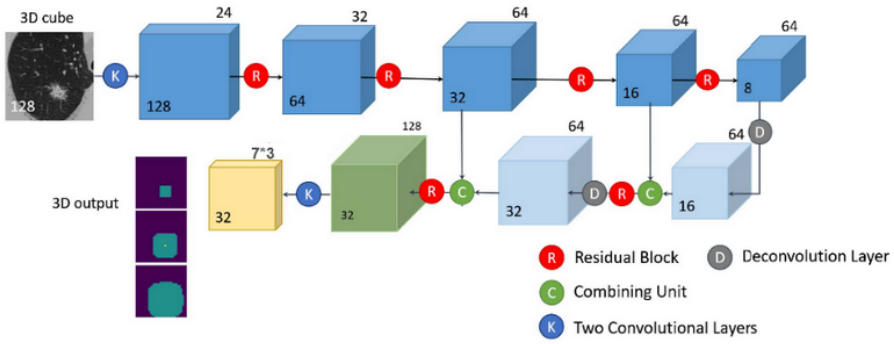

Figure 2.4: The model architecture for NNet. The connections pointing downwards towards the combining unit are the skip connections.

2.2 Methods

2.2.1 Patch Generation

For the training of each of the below models, the entire IMARIS images were not used due to memory restrictions. Patches of the images were generated to circumvent the issue and have been used for further processing. While the sizes of the patches may have differed depending on the models, the principles used to generate the patches were the same.

The patch sizes were chosen such that their size would be at least double the size of the largest ground truth bounding box. This was done so that each patch contains adequate information for the respective model to learn.

The selection of patch size and step size ensured that if one patch contained a partial bounding box along some axis, the adjacent patch along the same axis would contain that bounding box in its entirety. In order to do this, the step size was selected to be the size of the largest bounding box in the dataset. For partial bounding boxes, if the size of such boxes (measured as the number of voxels contained by it) was less than one-fourth of the largest bounding box, then they would be ignored in training.

2.3 Computer Vision

2.3.1 Nodule-Net (NNet)

I used a 3D UNet-based architecture inspired by a model called NoduleNet [15], which has been an innovative development in recent times and was also awarded the Data Science Bowl prize in 2017. The developers of this model aimed to automatically detect suspicious lesions in 3D CT lung data. Their model had two components: detecting the suspicious lesions and evaluating the malignancy. For the scope of the thesis, the focus was placed on the former, through which I would detect cells of a single class. The architecture for this is in Figure 2.4 below.

Model Architecture

The network architecture for nodule detection is a 3-D convolutional neural network that utilizes a modified 3-D U-net as its backbone. The network is divided into two main paths: a feedforward path and a feedback path with skip connections, enabling it to capture multiscale information essential for detecting nodules of varying sizes.

In the feedforward path, the network processes small 3-D patches of size [insert size here]. It begins with two $3 \times 3 \times 3$ convolutional layers with 24 channels. This is followed by four

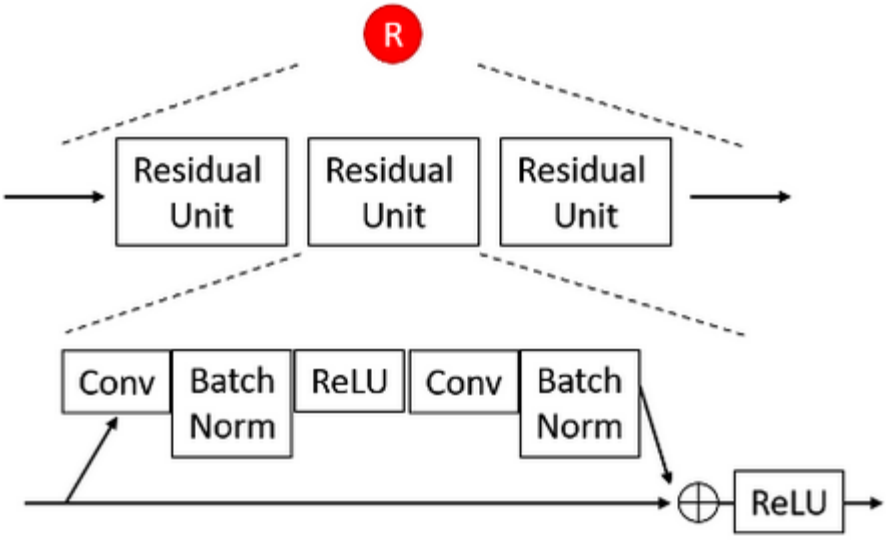

Figure 2.5: The residual block used by NNet.

3-D residual blocks, each containing three residual units, which are interleaved with four 2 $\times 2 \times 2$ max-pooling layers. All kernels in the feedforward path have a size of $3 \times 3 \times 3$ with padding of 1.

In the network architecture, the residual blocks are a critical component. Each residual block is composed of three residual units. The structure of a residual unit includes a convolutional layer followed by batch normalization and a ReLU activation function, another convolutional layer followed by batch normalization, and a shortcut connection that adds the input of the residual unit to the output before the final activation. This design allows the network to learn residual mappings instead of directly learning unreferenced mappings [15]. By using shortcuts or skip connections, residual blocks help to mitigate the vanishing gradient problem, which is common in very deep networks [16]. This enables the training of much deeper networks and improves convergence rates. Residual blocks ease optimization by allowing gradients to flow through the network more smoothly during backpropagation, making it easier to optimize very deep networks [16]. Secondly, they improve accuracy by learning residual functions, which helps the network to more effectively refine the features learned at each layer, leading to better overall performance.

Batch normalization (BN) is a technique used to normalize the inputs of each layer so that they have a mean of zero and a variance of one. This normalization is done for each mini-batch during training. The benefits of batch normalization are numerous. It stabilizes the training process by normalizing the inputs of each layer, reducing internal covariate shift and allowing for higher learning rates [17]. It also has a regularization effect, as BN adds some noise to the training process by computing statistics over mini-batches, which can reduce the need for other forms of regularization like dropout [17]. Furthermore, BN improves gradient flow by helping maintain well-behaved gradients, making it easier to train deeper models [17].

In summary, the residual blocks and batch normalization play essential roles in enhancing the performance and training stability of deep neural networks. Residual blocks address the challenges of training very deep networks by ensuring better gradient flow, while batch normalization stabilizes and accelerates the training process by normalizing layer inputs and reducing internal covariate shift.

The feedback path consists of two deconvolutional layers with a stride of 2 and a kernel

size of 2. It also includes two combining units that concatenate feedforward and feedback blobs. The feedback path is supplemented with additional $1 \times 1 \times 1$ convolutions to reduce the output tensor size.

Loss Computation

The final output is a 4-D tensor reshaped to $32 \times 32 \times 32 \times 4 \times 7$, representing the predicted proposals. Each location has three anchors of different scales (the largest bounding box sized multiplied by 1/3, ½, 3/4 and 1), with seven regression values $(\hat{o}, \hat{d}_z, \hat{d}_y, \hat{d}_x, \hat{d}_{rz}, \hat{d}_{ry}, \hat{d}_{rx})$. A sigmoid activation function is used for the first one and no activation function is used for the others:

$$\hat{p} = \sigma(\hat{o}) = \frac{1}{1 + \exp(-\hat{o})} \tag{2.4}$$

Denote the ground truth bounding box of a target nodule by $(G_z, G_y, G_x, G_{rz}, G_{ry}, G_{rx})$ and the bounding box of an anchor by $(A_z, A_y, A_x, A_{rz}, A_{ry}, A_{rx})$. The bounding box regression labels are defined as:

$$d_z = \frac{G_z - A_z}{A_{rz}} \tag{2.5}$$

$$d_y = \frac{G_y - A_y}{A_{ry}} \tag{2.6}$$

$$d_x = \frac{G_x - A_x}{A_{rx}} \tag{2.7}$$

$$d_{rz} = \log\left(\frac{G_{rz}}{A_{rz}}\right) \tag{2.8}$$

$$d_{ry} = \log\left(\frac{G_{ry}}{A_{ry}}\right) \tag{2.9}$$

$$d_{rx} = \log\left(\frac{G_{rx}}{A_{rx}}\right) \tag{2.10}$$

Anchor boxes which have an intersection over union (IOU) of greater than 0.5 with the ground truth bounding boxes are assigned a probability of 1 and are called positive samples. Anchors with an IOU of less than 0.02 are assigned a probability of 0 and are called negative samples. Anchors with an 0.02¡=IOU¡0.5 are ignored in training. The predicted probability and label for an anchor box is denoted by $(\hat{p})$ and (p), respectively. Note that $p \in \{0, 1\}$ (0 for negative samples and 1 for positive samples). The classification loss for this box is then calculated via binary cross entropy (BCE):

$$L_{\text{cls}} = p \log(\hat{p}) + (1 - p) \log(1 - \hat{p}) \tag{2.11}$$

The total regression loss is defined by:

$$L_{\text{reg}} = \sum_{k \in \{z,y,x,rz,ry,rx\}} S(d_k, \hat{d}_k) \tag{2.12}$$

where the loss metric $S(d, \hat{d})$ is a smoothed $L1$-norm function:

$$S(d, \hat{d}) = \begin{cases} |d - \hat{d}|, & \text{if } |d - \hat{d}| > 1 \\ (d - \hat{d})^2, & \text{otherwise} \end{cases} \tag{2.13}$$

The loss function for each anchor box is defined by:

$$L = L_{\text{cls}} + pL_{\text{reg}} \tag{2.14}$$

This equation indicates that the regression loss only applies to positive samples because only in these cases $p = 1$. The overall loss function is the mean of the loss function for some selected anchor boxes, using positive sample balancing and hard negative mining for selection.

Negative hard mining is performed during training in order to select the negative samples with the highest output probabilities. This is done so that the model can learn to distinguish the most difficult samples.

2.3.2 YOLOv5

YOLOv5 (You Only Look Once version 5) builds upon previous YOLO versions, bringing significant improvements in speed and accuracy. Unlike earlier versions that used Darknet, YOLOv5 is implemented in PyTorch via the Ultralytics module [ultralytics]. Its architecture includes the CSPDarknet53 backbone, which enhances feature extraction through cross-stage partial connections. YOLOv5 also introduces a new Focus layer, replacing the first three layers of YOLOv3 to reduce parameters and increase processing speed. The model comes in various sizes (YOLOv5s, YOLOv5m, YOLOv5l, YOLOv5x), allowing for flexible use cases ranging from lightweight to more complex object detection tasks.

Overall Concept

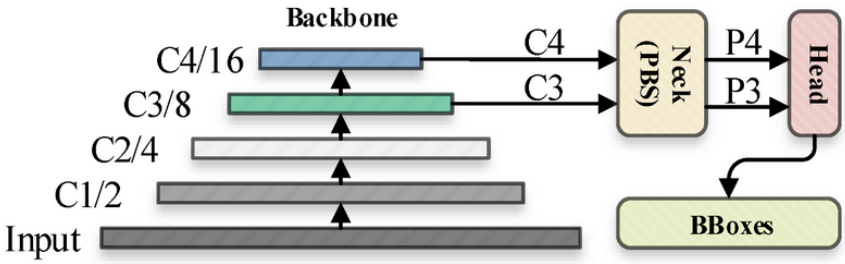

Figure 2.6: An overview of YOLOv5 [12].

YOLOv5 performs object detection by splitting the input image into a grid of cells. Each cell predicts a fixed number of bounding boxes, along with confidence scores and class probabilities. The process involves four key steps:

1. **Residual Blocks:** The image is divided into grid cells, each responsible for detecting objects within its region.
2. **Bounding Box Regression:** Each cell predicts bounding box coordinates and confidence scores.

3. **Intersection Over Union (IOU):** IOU is used to discard overlapping boxes by computing the overlap between predicted boxes and the ground truth.

4. **Non-Maximum Suppression (NMS):** NMS further refines predictions by keeping only the boxes with the highest confidence scores, eliminating redundant detections.

Backbone

The backbone of YOLOv5 is based on the CSPDarknet53 architecture [18]. CSPDarknet53, or Cross Stage Partial Darknet, is optimized for extracting rich and dense feature maps from the input images [18]. It incorporates a cross-stage partial network that splits the feature map into two parts: one part undergoes transformation while the other part remains as a shortcut connection. This design helps in reducing computational complexity and improving gradient flow, leading to better learning of the network.

C3 Module in the YOLOv5 Backbone: The C3 module in the YOLOv5 backbone is a sophisticated feature extraction component. It enhances the efficiency and performance of the network by integrating two significant design elements: residual blocks and cross-stage partial connections. It is a specific implementation of CSPNet (Cross Stage Partial Network) (Figure 2.7) [18]. It not only ensures that the backbone has excellent feature extraction ability, but also curbs the problem of gradient information duplication in the backbone [18].

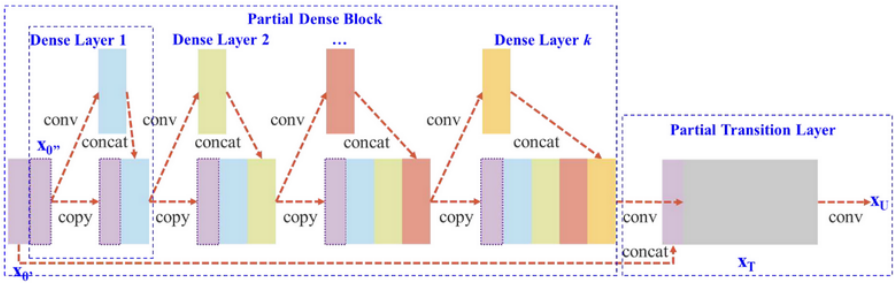

Figure 2.7: CSPNet module architecture [12].

Cross-Stage Partial Connections: The C3 module splits the input feature map into two parts. One part undergoes several residual transformations, while the other part bypasses these transformations, acting as a shortcut connection. This dual-path approach preserves gradient flow and maintains feature diversity by ensuring that some of the original features are directly propagated through the network without transformation. By merging these two paths, the C3 module helps achieve leading to a better feature representation [12].

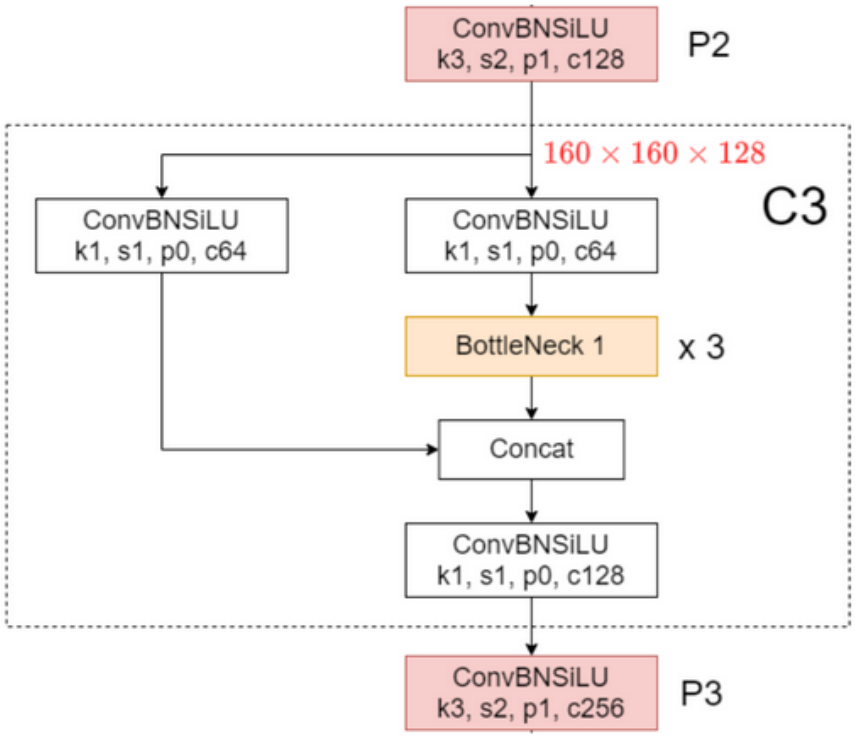

Figure 2.8: C3 module architecture [12].

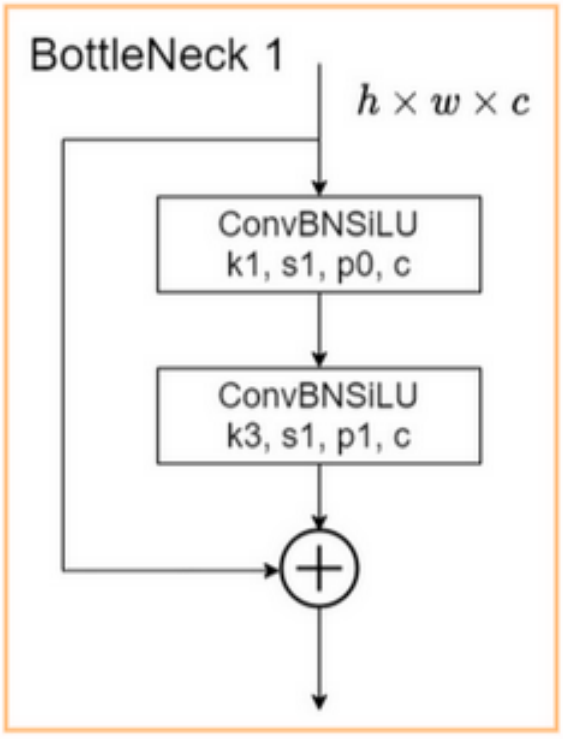

Figure 2.9: Residual block used in C3 [12].

Residual Blocks: These blocks allow the network to learn residual functions with reference

to the layer inputs, which helps in mitigating the vanishing gradient problem and enables the training of deeper networks. Each residual block typically includes a series of convolutional layers followed by batch normalization and activation functions. This configuration helps the network learn more complex patterns and features from the input data.

Efficiency and Performance: The C3 module's design reduces the computational burden by strategically balancing the processing load between transformed and shortcut features. This not only speeds up the training and inference processes but also helps in achieving better detection performance, especially for small objects and in real-time applications. The combination of these enhancements makes the C3 module a crucial component in the YOLOv5 architecture, contributing significantly to its state-of-the-art performance in object detection tasks.

Neck

The neck of YOLOv5 integrates two powerful feature fusion techniques: Path Aggregation Network (PANet) and Bi-directional Feature Pyramid Network (BiFPN) [12]. PANet enhances the information flow across different layers of the network by using lateral connections and a bottom-up path augmentation. This helps in capturing finer details which are crucial for detecting small objects. BiFPN further improves this by adding bidirectional cross-scale connections and weighted feature fusion, allowing the model to better aggregate features from various scales. The basic idea of FPN is to up-sampling the output feature map (C3, C4, and C5) generated by multiple convolution down sampling operations from the feature extraction network to generate multiple new feature maps (P3, P4, and P5) for detecting different scales targets.These improvements enable the neck to produce more robust feature representations that enhance detection accuracy.

Head

The head of YOLOv5 is responsible for generating the final predictions. It outputs bounding box coordinates, objectness scores, and class probabilities for each grid cell in the image. The bounding box coordinates include the center position, width, and height of the box, while the objectness score indicates the likelihood of an object being present within the box. Class probabilities are used to classify the detected objects into predefined categories. The head is designed to be efficient and accurate, ensuring that the model can perform real-time object detection even for small and densely packed objects.

Spatial Pyramid Pooling: The Spatial Pyramid Pooling (SPP) layer in YOLOv5 plays a crucial role in improving the model's ability to handle objects of various scales and aspect ratios. SPP achieves this by applying multiple pooling operations with different kernel sizes (e.g., 1x1, 3x3, 5x5) on the feature maps, which capture contextual information from different scales [12]. This multi-scale pooling approach allows the model to retain spatial information and recognize objects regardless of their size or position in the image. By aggregating these pooled features, the SPP layer provides a richer and more comprehensive feature representation, enhancing the network's overall detection accuracy [12].

Model Output

In the YOLOv5 object detection algorithm, the bounding box regression process adjusts the predicted bounding boxes to more accurately match the ground truth boxes. The variables involved in this process are explained as follows:

- σ: This represents the sigmoid function, which is used to constrain the predictions to be within a certain range.

- s_x and s_y: These are the offsets predicted by the model for the center coordinates of the bounding box. They are passed through the sigmoid function to ensure they lie between 0 and 1.
- r_x and r_y: These are the unadjusted coordinates of the predicted center point of the bounding box within the grid cell.
- g_x and g_y: These represent the adjusted center coordinates of the predicted bounding box. The adjustment formula is:

$$g_x = 2\sigma(s_x) - 0.5 + r_x$$
$$g_y = 2\sigma(s_y) - 0.5 + r_y$$

  This formula shifts and scales the sigmoid output to align the bounding box center with the grid cell location and predicted offset.
- g_w and g_h: These represent the adjusted width and height of the predicted bounding box. The adjustment formula is:

$$g_w = p_w(2\sigma(s_w))^2$$
$$g_h = p_h(2\sigma(s_h))^2$$

Here, p_w and p_h are the width and height of the prior anchor boxes. The sigmoid function ensures that the width and height predictions are positive and reasonable.

Loss

YOLOv5 computes its total loss as a combination of three distinct loss components [11]:

1. **Classes Loss (Binary Cross-Entropy Loss - BCE)**:

$$L_{\text{cls}} = -\sum_{i=1}^{C} (y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i))$$

   This loss measures the accuracy of the predicted class probabilities for each detected object. YOLOv5 also employs binary cross-entropy (BCE) loss for multi-label classification, as objects can belong to multiple classes.

2. **Objectness Loss (Binary Cross-Entropy Loss - BCE)**:

$$L_{\text{obj}} = -\sum_{i=1}^{S \times S \times B} (p_i \log(\hat{p}_i) + (1 - p_i) \log(1 - \hat{p}_i))$$

   This component penalizes the confidence score of the predicted bounding boxes, which indicates the likelihood of the presence of an object. YOLOv5 uses binary cross-entropy (BCE) loss for this purpose, effectively distinguishing between object and non-object predictions.

3. **Location Loss (Complete IoU Loss - CIoU)**: This loss measures the difference between the predicted bounding box coordinates and the ground truth coordinates. YOLOv5 uses CIoU (Complete Intersection over Union) loss, which takes into account the overlap area, the distance between the centers of the predicted and ground truth boxes, and the aspect ratio consistency.

$$L_{\text{loc}} = 1 - \text{IoU} + \frac{\rho^2(\mathbf{b}, \mathbf{b}^g)}{c^2} + \alpha v$$

   Where ρ is the Euclidean distance between the predicted and ground truth box centers, c is the diagonal length of the smallest enclosing box, α is a positive trade-off parameter, and v is an aspect ratio consistency measure. CIoU loss measures the error in localizing objects within grid cells.

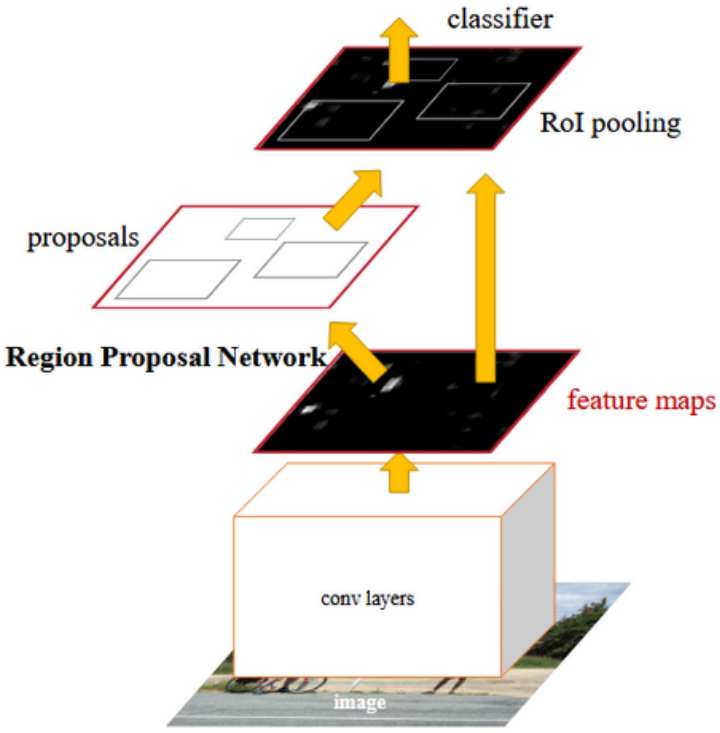

Figure 2.10: Overall flow of Faster-RCNN [9].

2.3.3 Faster R-CNN

The Faster R-CNN model aims to improve the speed and accuracy of object detection in images. It achieves this by integrating region proposal and object detection into a single, unified network, thus reducing the computational overhead associated with traditional methods that handle these tasks separately. The key innovation of Faster R-CNN is the introduction of Region Proposal Networks (RPNs), which enable the model to generate region proposals with high efficiency, effectively bridging the gap between region proposal and detection processes. The ultimate goal of Faster R-CNN is to provide a real-time object detection system that can be used in various applications, ranging from autonomous driving to surveillance and beyond.

Model Architecture

Faster R-CNN consists of two primary modules: the Region Proposal Network (RPN) and the Fast R-CNN detector. The architecture begins with a deep fully convolutional network that extracts feature maps from the input image. This network is called the backbone, and a ResNet50 architecture was chosen for this. The number 50 indicates the depth of the network.

The RPN slides a small network over these shared feature maps to generate region proposals, which are essentially bounding boxes that might contain objects. Each proposal is accompanied by an objectness score, which indicates the likelihood of the proposal containing an object.

The output feature maps from the ResNet are fed to the next portion of the RPN. From the feature map, the RPN uses a set of predefined anchor boxes of different scales and aspect ratios to obtain proposals. At each spatial location of the feature map, the RPN outputs k proposals (where k is the number of anchor boxes), with each proposal parameterized by its coordinates and an objectness score. These proposals are then fed into the Fast R-CNN detector, which performs region-wise classification and regression to refine the proposals and

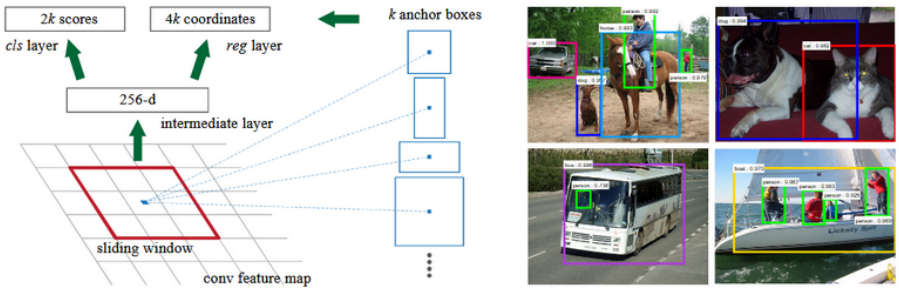

Figure 2.11: An illustration of sliding windows and object detection by Faster-RCNN [9].

classify them into different object categories. This two-stage process ensures that the model maintains high accuracy while being computationally efficient. The Faster R-CNN method also ensures translation invariance i.e. the same object would be detected regardless of its spatial position on the input image [9].

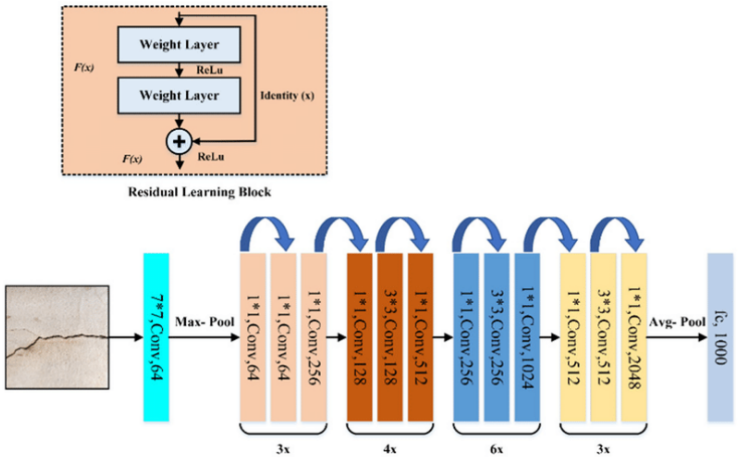

Figure 2.12: ResNet Module [19].

ResNet, or Residual Network, is a deep convolutional neural network designed to facilitate the training of very deep networks by addressing the vanishing gradient problem. Its key innovation is the introduction of residual blocks, which incorporate identity connections, or skip connections, that bypass one or more layers [19]. These connections allow the input of a block to be added directly to its output, making it easier for gradients to flow during backpropagation. Another important feature of ResNet is the extensive use of batch normalization, which helps stabilize and accelerate training [9]. Additionally, ResNet employs global average pooling at the end of the network to reduce the spatial dimensions of feature maps before passing them to the fully connected layer, which reduces the number of parameters and prevents overfitting [9].

Loss

The output of the Faster R-CNN model includes both the bounding boxes of detected objects and the corresponding class scores. The RPN generates a set of region proposals, each with an objectness score, which are then refined by the Fast R-CNN detector to produce the final object detections. The Fast R-CNN detector outputs a softmax probability distribution over the object classes and the refined bounding box coordinates for each proposal.

The loss function used in Faster R-CNN is a multi-task loss that combines classification and

regression losses. For the RPN, the loss function includes a binary classification loss (object vs. non-object) and a regression loss that measures the accuracy of the bounding box coordinates. For the Fast R-CNN detector, the loss function includes a softmax loss for object classification and a smooth L1 loss for bounding box regression. This combined loss ensures that the model optimally balances the tasks of object classification and localization, leading to precise and accurate detections.

$$L_{cls}(p_i, p_i^*) = -\log\left(p_i^* p_i + (1 - p_i^*)(1 - p_i)\right) \tag{2.15}$$

$$L(\{p_i\}, \{t_i\}) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*) \tag{2.16}$$

2.3.4 RetinaNet

The RetinaNet detector integrates a backbone network with two subnetworks: one for object classification and the other for bounding box regression. The backbone is a Feature Pyramid Network (FPN) built on top of a ResNet architecture. The classification subnetwork performs convolutional object classification on the FPN output, while the box regression subnetwork predicts the offset from each anchor box to a nearby ground-truth object [10]. Both subnetworks share a similar structure but operate independently.

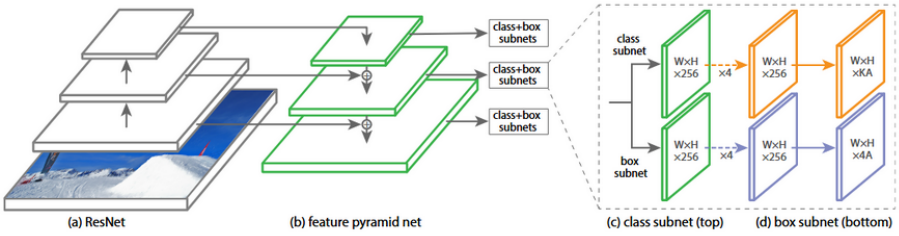

Figure 2.13: A schematic of RetinaNet [10].

FPN was designed to enhance object detection by utilizing multi-scale feature maps [20]. FPN builds a top-down architecture with lateral connections, which allows it to construct high-level semantic feature maps at various scales. This design is aimed at detecting objects of different sizes, as it enables the network to use multi-resolution features for more accurate predictions [20]. By combining low-resolution, semantically strong features with high-resolution, spatially precise features, FPN significantly improves the performance of detectors like RetinaNet, especially in handling small objects in complex scenes [20].

During inference, RetinaNet processes an image through the network and applies non-maximum suppression to yield final detections. A novel focal loss [10] is applied to all approximately 100k anchors in each image, contrasting with the heuristic sampling used in other methods. This approach ensures that the vast number of easy negatives do not dominate the training process. The network is trained using stochastic gradient descent with specific initialization strategies to handle the large number of background anchors, ensuring stable training [10].

Focal Loss

The Focal Loss addresses the significant class imbalance between foreground and background classes that one-stage object detectors face during training. Traditional cross-entropy (CE) loss is inadequate in this scenario as it is overwhelmed by easily classified negatives, contributing

to inefficient learning. To mitigate this, a modulating factor was introduced to the CE loss that down-weights the loss assigned to well-classified examples. The proposed Focal Loss is defined as:

$$FL(p_t) = -(1 - p_t)^{\gamma} \log(p_t) \tag{2.17}$$

Where p_t is the model's estimated probability for the true class, and γ is a tunable focusing parameter. γ reduces the loss contribution from easy examples, focusing the training on hard negatives. The focal loss is visualized for various values of γ (Figure 2.14), showing that as γ increases, the effect of the modulating factor increases, thus reducing the impact of easily classified negatives. The paper by *Lin et al* [10] found that a value of $\gamma = 2$ works best in experiments. Additionally, the focal loss can be extended with an α-balancing factor to adjust the importance between positive and negative examples, further improving accuracy.

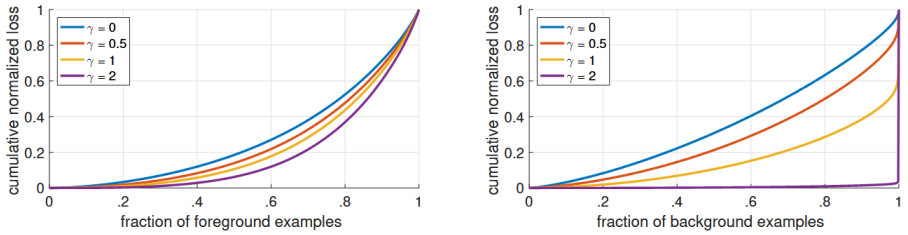

Figure 2.14: Plot displaying the effect of γ on the focal loss [10].

2.3.5 Metrics to Evaluate Object Detection Algorithms

Localization Quality: Intersection Over Union (IOU)

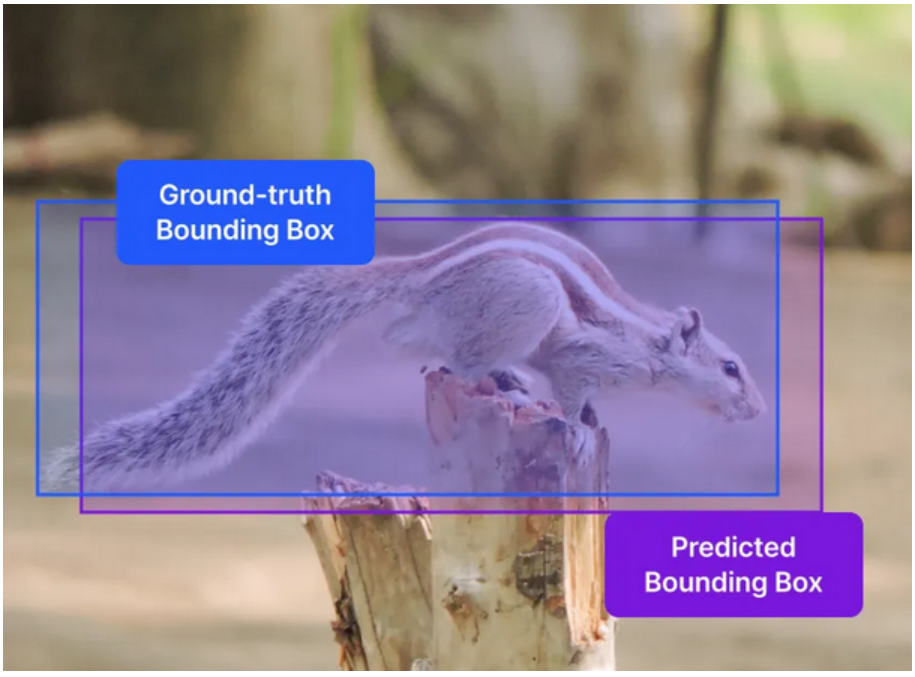

Figure 2.15: An example of a prediction and ground truth bounding box [21].

One of the most commonly used metrics to evaluate object detection algorithms is IOU. This is a way to measure the amount of overlap between two bounding boxes, and it serves as a

quality measure for the localization i.e. the location of a bounding box on an image. If G is a ground truth bounding box and B is a prediction given by a model, the IOU can be found via:

$$IOU(G,B) = \frac{|G \cap B|}{|G \cup B|}; 0 \leq \text{IoU} \leq 1 \tag{2.18}$$

The IOU is the ratio of the overlapping area between G and B to the union of the areas of G and B. If the prediction is perfect i.e. identical to the ground truth, then the IOU will be 1 as the overlapping area is equivalent to the union of areas. If the prediction does not overlap at all with the ground truth, then the IOU is 0 and the prediction is considered poor.

When using IOU as an evaluation metric, a threshold is often used. If the IOU of a prediction with a ground truth bounding box is greater than or equal to this threshold, then the prediction is considered valid. For the purposes of this thesis, the threshold for IOU was 0.5.

2.3.6 Classification Quality: Interpreting Predicted Classes

Confidence

Object detectors, along with predicting the locations of bounding boxes, will also predict a class for the object and a confidence value. This confidence denotes the model's certainty that a particular bounding box contains a relevant object. One component of an object detection study is finding a confidence threshold such that when samples with confidences above the threshold are considered valid, certain quality metrics may be maximized (see Sections 2.3.7 and 2.3.8).

Confidence Score: This value is typically between 0 and 1, and indicates how confident the model is that a bounding box contains an object as opposed to just background. A higher confidence score means the model is more certain that the bounding box contains an object.

Objectness Score: In some models, like YOLO, the confidence score is also known as the objectness score, representing the likelihood that a bounding box contains any object, irrespective of its class.

Combined Confidence:The final confidence score for a predicted bounding box is the product of the objectness score and the class probability. This combined score indicates both the likelihood that the bounding box contains an object and the likelihood that the object belongs to a specific class.

Positive and Negative Predictions

Let us take the example of a binary classifier, which returns 1 if a particular outcome is detected and 0 if it is not. In the above diagram (Figure 2.16), legitimate activities are denoted by the value 1 and fraudulent activities are denoted by 0. The metrics for such a classifier can be explained by:

True positives (TP): when the ground truth is 1 and the prediction is 1. **False positives (FP)**: when the ground truth is 0 and the prediction is 1. **False negatives (FN)**: when the ground truth is 1 and the prediction is 0. **True negatives (TN)**: when the ground truth is 0 and the prediction is 0.

Three key metrics can be obtained from the above: True Positive Rate (TPR), False Positive Rate (FPR) and False Negative Rate (FNR).

$$TPR = \frac{TP}{TP + FN} \tag{2.19}$$

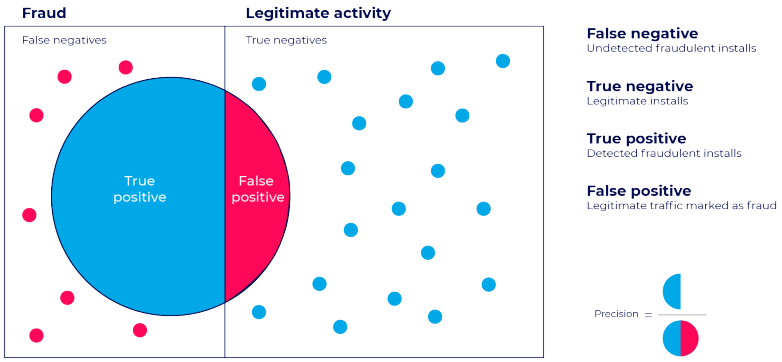

Figure 2.16: An illustration of true positives, false positives, false negatives and true negatives [22] in a binary classifier. In this diagram, blue represents legitimate activity and red represents fraudulent activity.

$$FPR = \frac{FP}{FP + TN} \tag{2.20}$$

$$FNR = \frac{FN}{TP + FN} = 1 - TPR \tag{2.21}$$

These metrics can be crucial depending on the application. For instance, if a doctor is attempting to diagnose whether a patient may have cancer (a binary classification problem where 1 denotes cancer), it is crucial that their FNR is is minimized, as an incorrect diagnosis can be the difference between life or death. Minimizing FNR would maximize TPR. A second priority would be minimizing the FPR. If chemotherapy is the chosen treatment for the particular cancer, the doctor would like to avoid it if it isn't necessary as such therapy may be painful for the patient. It is not always possible to minimize both FNR and FPR. Depending on the application, one or the other may need to be prioritized.

When applying these metrics to multi-class problems, a one-vs-all approach may be taken. Given a label L, all predictions or ground truths with the label L are assigned the value 1, and any other labels are assigned the value 0.

ROC Curve: A Receiver Operating Characteristic curve is a way to visualize the tradeoff between TPR and FPR. To make such a plot, different confidence thresholds are used. For each threshold, the predictions are divided into TPs, FPs and FNs. These are then used to calculate the FPR and TPR, which are plotted on the x and y-axes respectively.

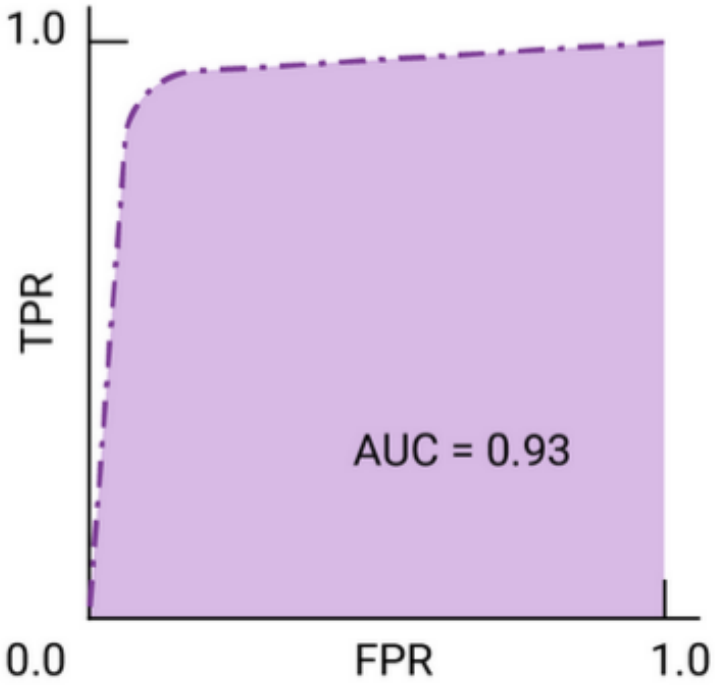

Figure 2.17: A sample ROC Curve [23].

A perfect classifier will predict the correct outcome 100% of the time. This means that regardless of the threshold chosen, the TPR will be 1. The area under the ROC curve (AUC) will then be 1. A poor classifier will have a low AUC as most of its predictions will be wrong.

The ROC curve is a useful way to select a confidence threshold for a classification problem. One metric that may be used is Youden's J Statistic [24]. This aims to minimize the difference between TPR and FPR. The confidence that accomplishes this is used as the threshold for classification.

Precision, Recall and F1-Score

Precision is a way to measure the quality of predictions and recall can measure the quantity of truly relevant results returned [25]. They are given by:

$$Precision = \frac{TP}{TP + FP} \tag{2.22}$$

$$Recall = \frac{TP}{TP + FN} \tag{2.23}$$

Precision gives the ratio of predicted positives that are actually positive to the total predicted positives. A high precision indicates a low number of false positives. Recall gives the ratio of predicted positives that are actually positive; this is the same as TPR. A high recall indicates a low number of false negatives.

Precision-Recall Curve: A Precision-Recall (PR) curve is another way to visualize the results of a detector. Similar to an ROC curve, TP, FP, FN and TN are extracted for multiple confidence thresholds. For a PR curve, the precision and recall are calculated and plotted on the y and x-axes respectively.

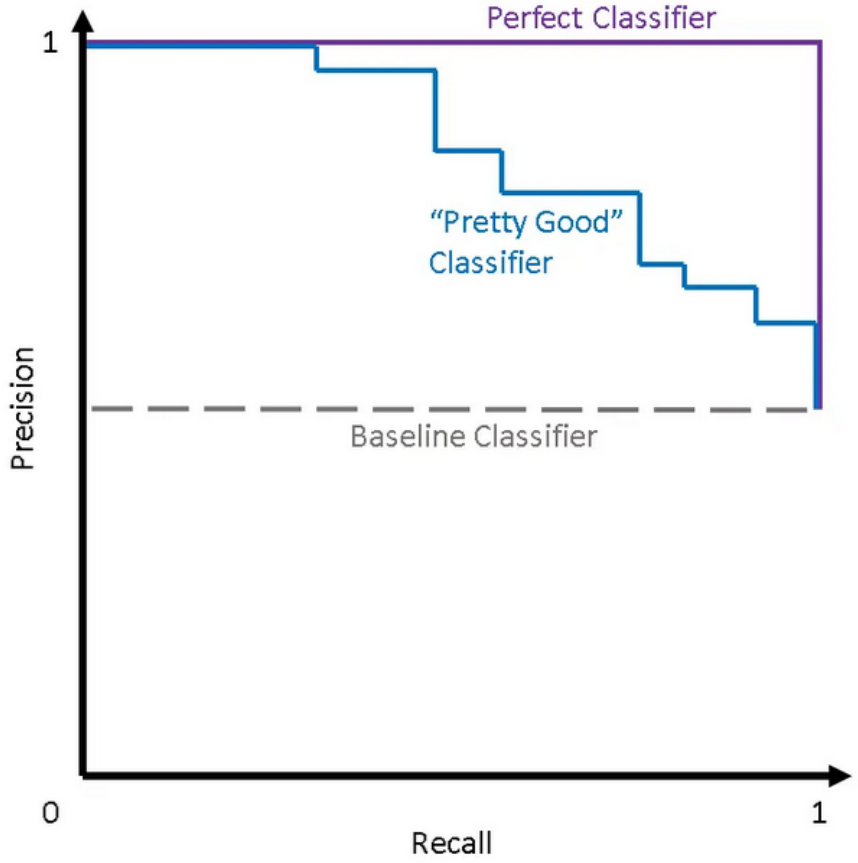

Figure 2.18: A sample PR Curve [26].

A perfect classifier will predict all true positives correctly and have no false positives. Therefore, the precision will always be 1 and the area under the curve (AUC) will be 1. A poor classifier will have low precision and therefore a low AUC.

When using a PR curve to choose a confidence threshold for a classifier, one method is to choose the confidence that maximizes the F1-Score as the threshold. This score is a measure for the tradeoff between precision and recall, and can be found by:

$$F1 = 2 \cdot \frac{\text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.24}$$

Confusion Matrix

The confusion matrix is a good way to visualize the relationship between ground truths and predictions after choosing a confidence threshold. This plots a grid of the true classes along the y-axis and the predictions along the x-axis. The values in each square represent the fraction of total samples that have been classified a certain way, given the ground truth. These values may also represent the total number of classifications rather than a fraction.

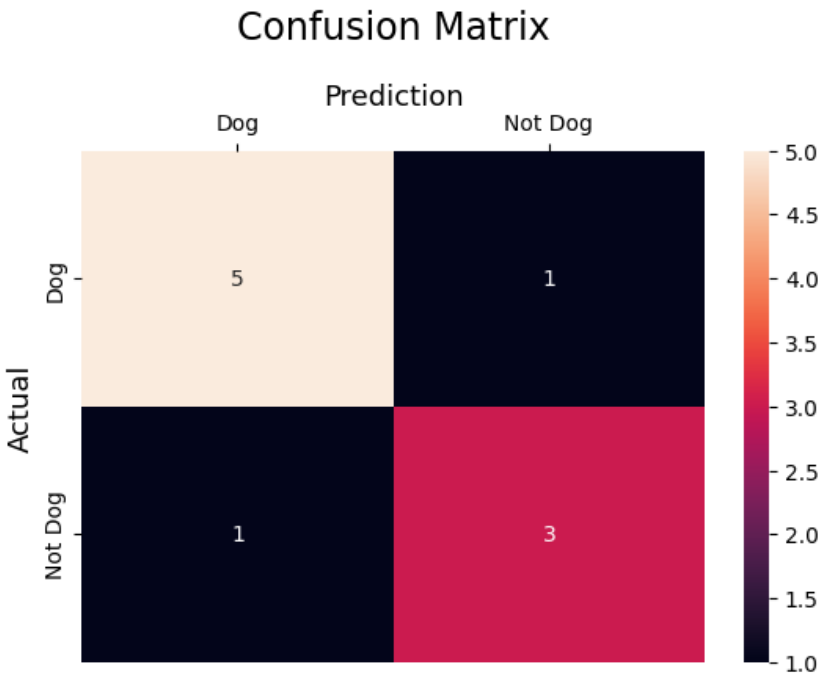

Figure 2.19: A sample confusion matrix [27].

The values inside a confusion matrix can be used to calculate precision, recall, TPR, FPR and FNR.

2.3.7 Implementation

Four object detection networks were implemented for this thesis. All of them were coded on Python with *PyTorchv*2.2.0 [28] and executed via bash scripts on University of Zurich's Science Cluster [29].

Adapted NoduleNet: This was inspired by [15].

YOLOv5: Used an implementation by Ultralytics [11]

Faster-RCNN: Used the Faster-RCNN library of torchvision [30]

RetinaNet: Used the RetinaNet library of torchvision [31]

Dataset

The data provided by the CNA Lab contained four lightsheet fluorescence images stored as IMARIS [32] files, each taken from a different mouse's bone marrow tissue. These were accessed and read via the Python module *h*5*pyv*3.11.0 [33]. Three channels - DAPI, GFP and Hlf-tdTomato - were used for the object detection. These channels were depicted as blue, green and red respectively. The data contained two cell types. The first was HSPCs (hematopoietic stem cells) that had not undergone differentiation yet. The second was DPs (double positives) which had begun differentiation. These cells' spots were annotated manually by members of the CNA Lab.

For NoduleNet, a dataset of 4000 3D patches was used. For the other three networks, 2D patches were extracted along the X, Y, and Z axes. The dataset size was 15000 images for each axis. The train/test/val split for all networks was 0.8/0.1/0.1.

Analysis and Visualization

The analysis of training results was done using the Python module $sci-kit-learnv1.5.1$ [34]. This module was used to make the confusion matrices as well as the PR and ROC curves. Visualizations were done using the Python module $matplotlibv3.8.4$ [35]. All calculations and array manipulations for the analysis were done using the Python module $numpyv1.26.4$ [36].

Chapter 3

Results

3.1 NoduleNet

The adapted NoduleNet was then trained on a training set of 3200 3D patches. However, the loss seemed to consistently be plateauing around 0.69 (Figure 3.1). The cause of this was the classification loss plateauing at 0.69. For the resulting predicted bounding boxes, all of the predictions had a probability of 0.5, regardless of the ground truth.

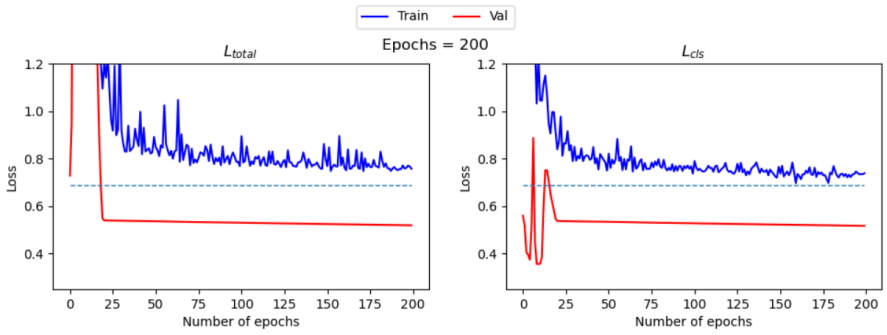

Figure 3.1: Training and validation losses for NoduleNet for the complete training set.

To diagnose this issue, the process of patch generation was then investigated to determine any fallacies in the process itself. All the patches were then fitted and their bounding boxes was then placed back to the original crop locations. Using visual inspection, the bounding boxes of the patches in the correct location relative to the ground truth spots from the raw IMARIS files was then determined. As shown in Figure 3.2, the bounding boxes were henceforth determined to be in the correct places.

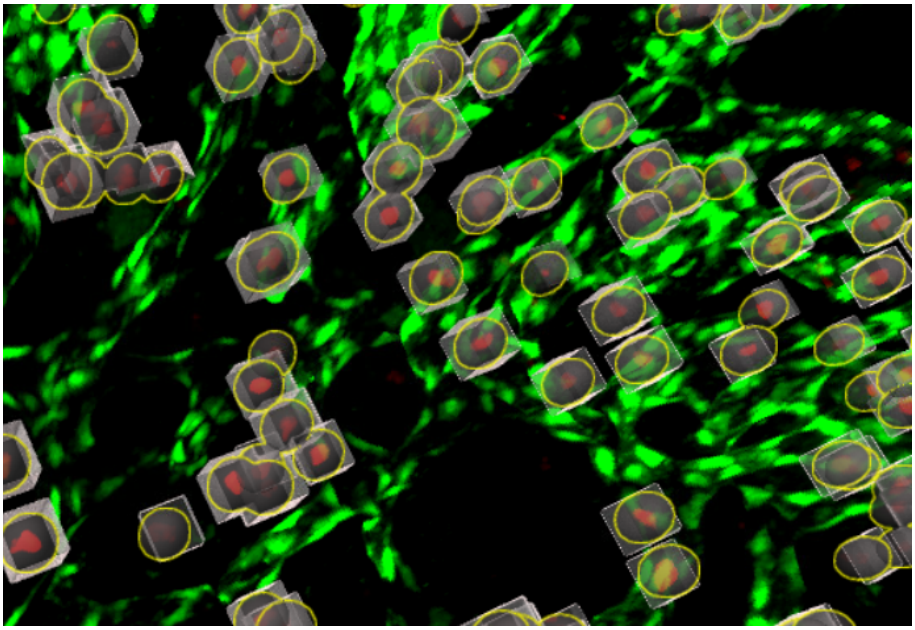

Figure 3.2: Stitched bounding boxes overlaid on the ground truth IMARIS data.

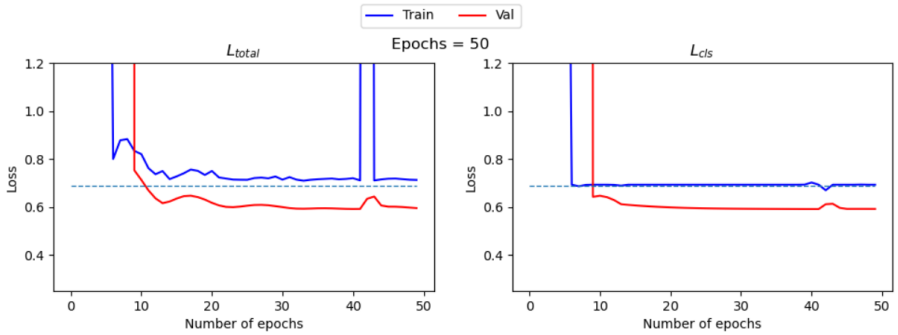

Figure 3.3: Attempted overfitting of NoduleNet.

After this, overfitting the model on a single patch was performed in order to see whether the model trained as intended. Again, the classification loss would plateau at 0.69 while the regression loss would train fully which required further investigation.

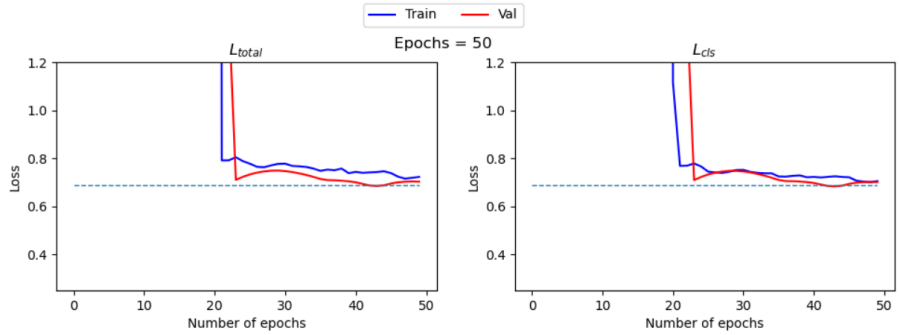

Figure 3.4: Attempted training with He initialization for the parameters

The parameters were by default mean initialized. He initialization was then attempted for debugging, however, the issue was persistent. [16]

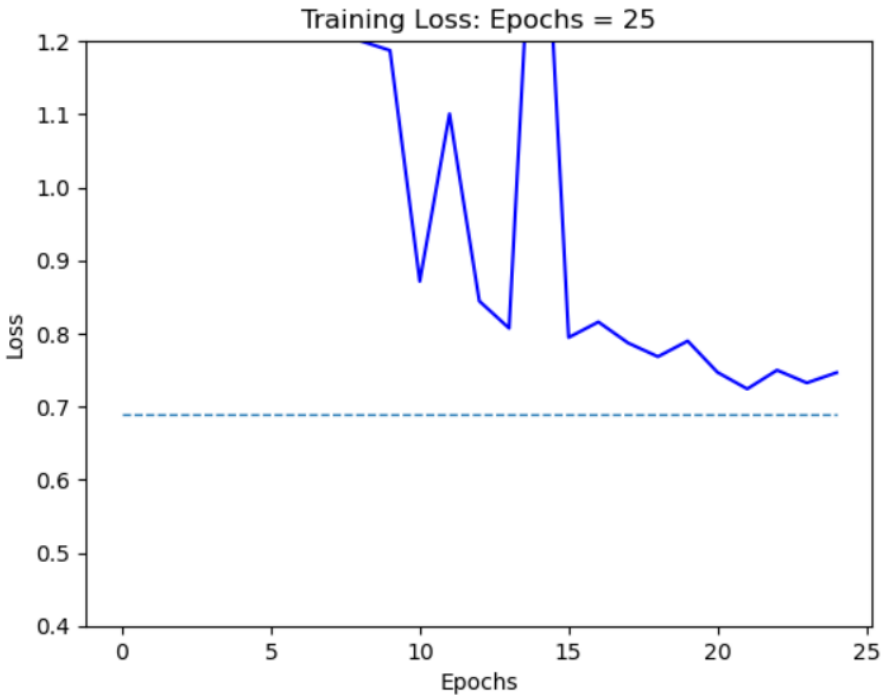

Figure 3.5: Attempted training with an Adam optimizer.

The default optimizer (used so far) was stochastic gradient descent (SGD) As part of further diagnosis, a training with Adam optimizer was attempted. This was chosen as it is robust for sparse data which was caused by the hard negative mining during loss calculations [37].

This plateauing was hypothesized to be due to vanishing gradients. In order to combat this, the ReLU activations in the network were replaced with LeakyReLU. This was done so that negative gradients would not be set to 0 but would still retain a value proportional to the magnitude of the negative value. However, the issue of the classification loss remained. As part of further diagnosis, despite trying Hinge loss instead of binary cross entropy was also attempted.

To understand the root cause of this issue, looking further into the network architecture, it was then hypothesized that the cause of the loss issue could be the lack of a certain tensor that was present in the original paper [15] but not in this implementation. This tensor was the location crop, and it contained information about the location of each voxel relative to the entire lung scan. This tensor had to be removed for the purposes of this thesis because the patches were not registered to each other; the patches were from different locations on the murine femur. This paper had access to more than a thousand images from unique patients [15] whereas the dataset used for this thesis only had data from 4 mice, increasing the complexity of the problem due to low data availability.

3.2 YOLOv5

YOLOv5 was then trained on 2D patch data for 500 epochs.

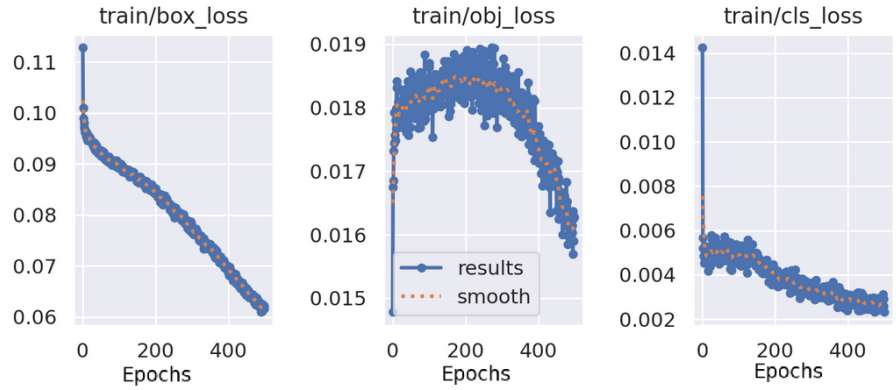

Figure 3.6: Training losses for YOLOv5 with patches extracted along the z-axis.

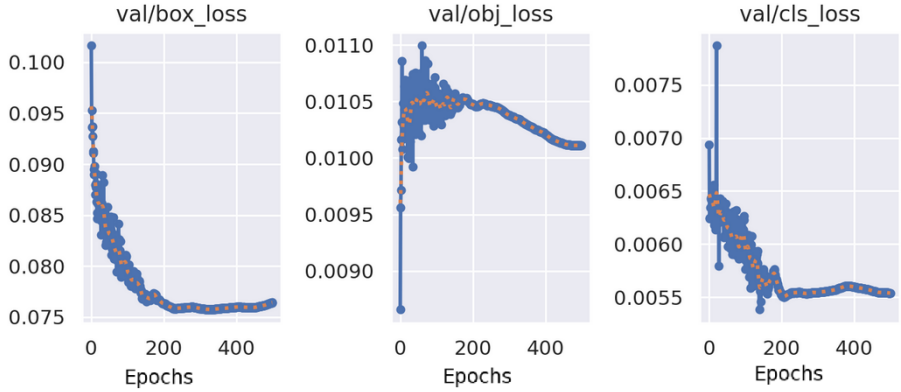

Figure 3.7: Validation losses for YOLOv5 with patches extracted along the z-axis.

The box and object training losses (Figure 3.6) did not stabilize even after 500 epochs. These losses for validation (Figure 3.7) did stabilize, with an uptick visible for the box loss during the last 50 epochs. This indicates that further training may cause overfitting as the validation loss would begin to increase.

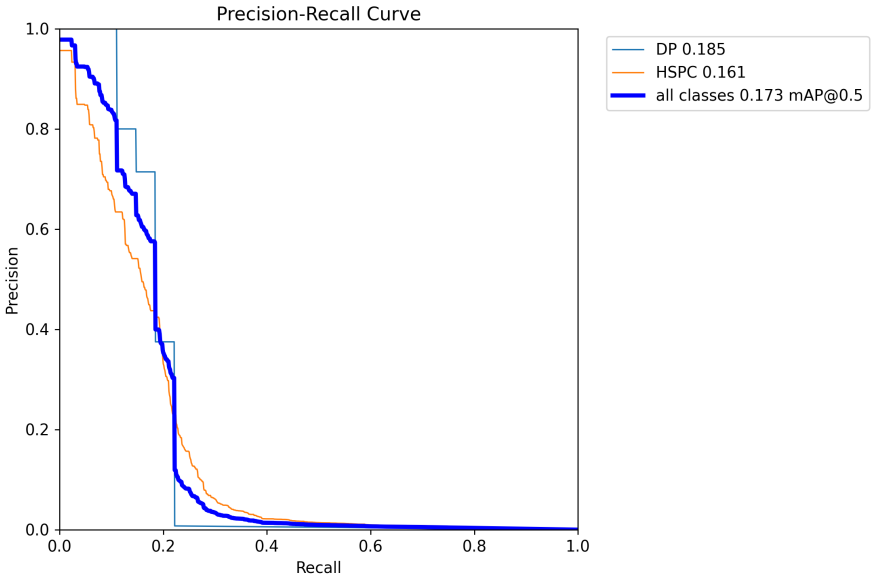

Figure 3.8: PR curve for YOLOv5 with patches extracted along the z-axis. The values next to the legend correspond to the area under the curve

The PR curve (Figure 3.8) indicates poor classification. This can be interpreted from the low areas under the curve for both classes DP and HSPC.

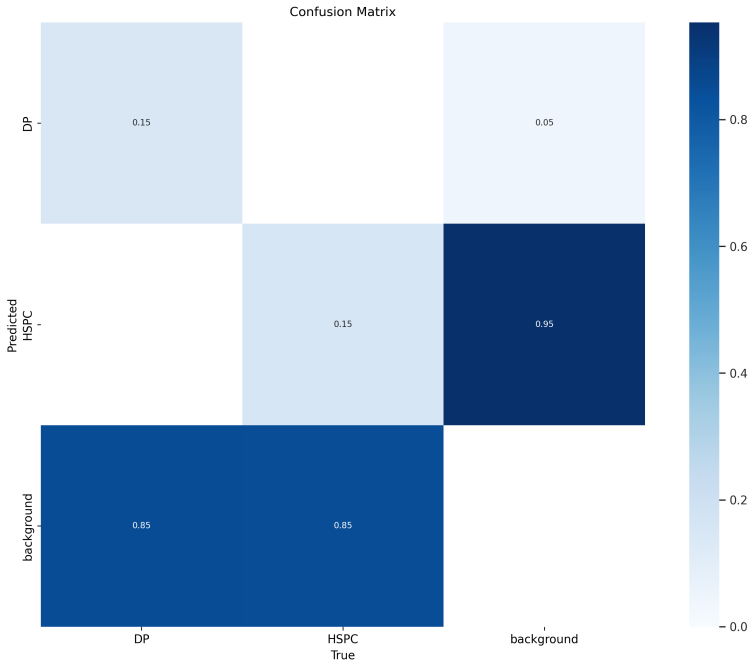

Figure 3.9: Confusion matrix for YOLOv5 with patches extracted along the z-axis.

The confusion matrix (Figure 3.9) shows that both cell classes had a very high proportion (85% for both) of false negatives, wherein cells were classified as background. This is consistent with the low recall shown in the PR curve (Figure 3.8). The proportion of true positives is 15% for both classes; this is also consistent with the low precision from the PR curve (Figure 3.8).

3.3 Faster-RCNN

Faster-RCNN was trained with a stopping criteria applied to the validation loss: if the total validation loss did not improve (decrease) for 10 epochs, then training would be stopped. The trend in validation loss was calculated using the running average across 5 epochs. This running average would be evaluated for the previous 10 epochs to determine whether the stopping criteria was met. This is illustrated in Figure 3.10

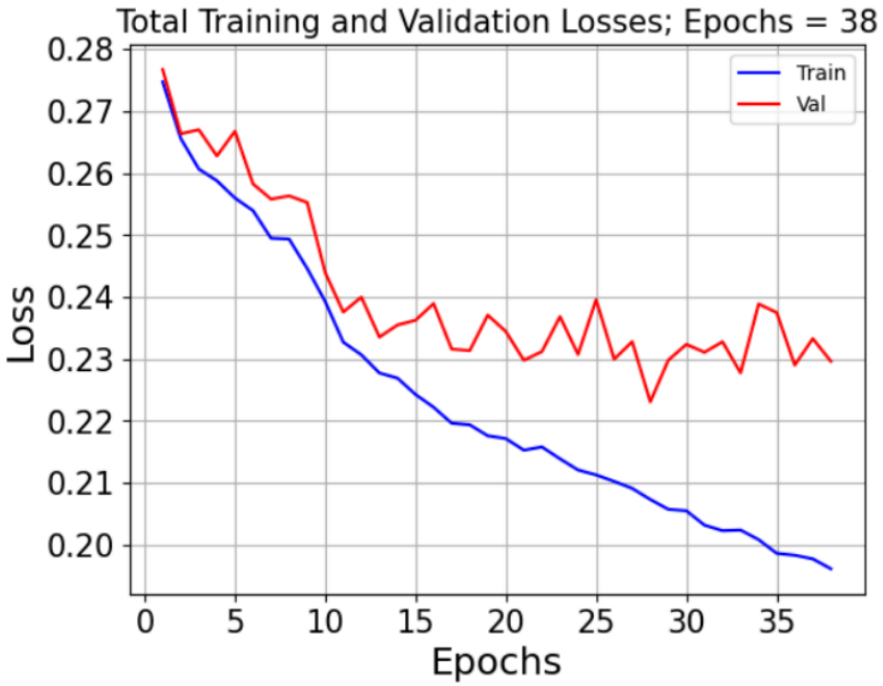

Figure 3.10: Training and validation losses for Faster-RCNN with patches extracted along the z-axis.

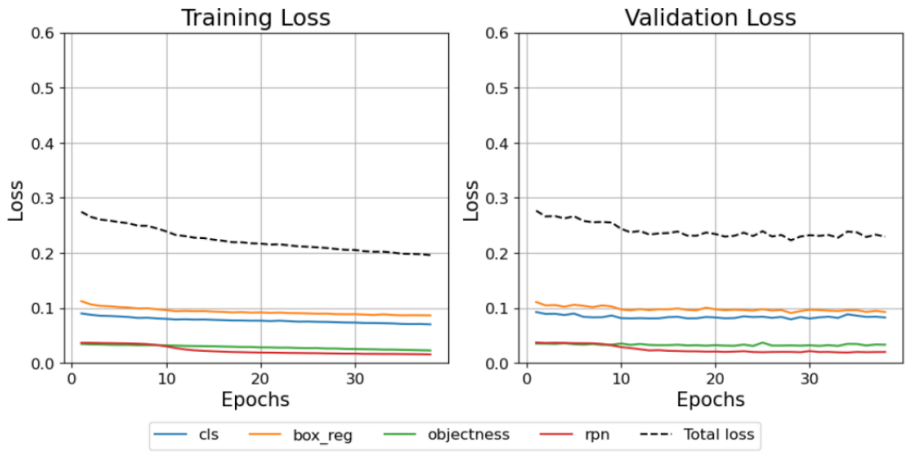

Figure 3.11: Breakdown of the training and validation losses for Faster-RCNN with patches extracted along the z-axis.

Figures 3.10 and 3.11 indicate that the model achieved a stable validation loss, with room for improvement for the training loss.

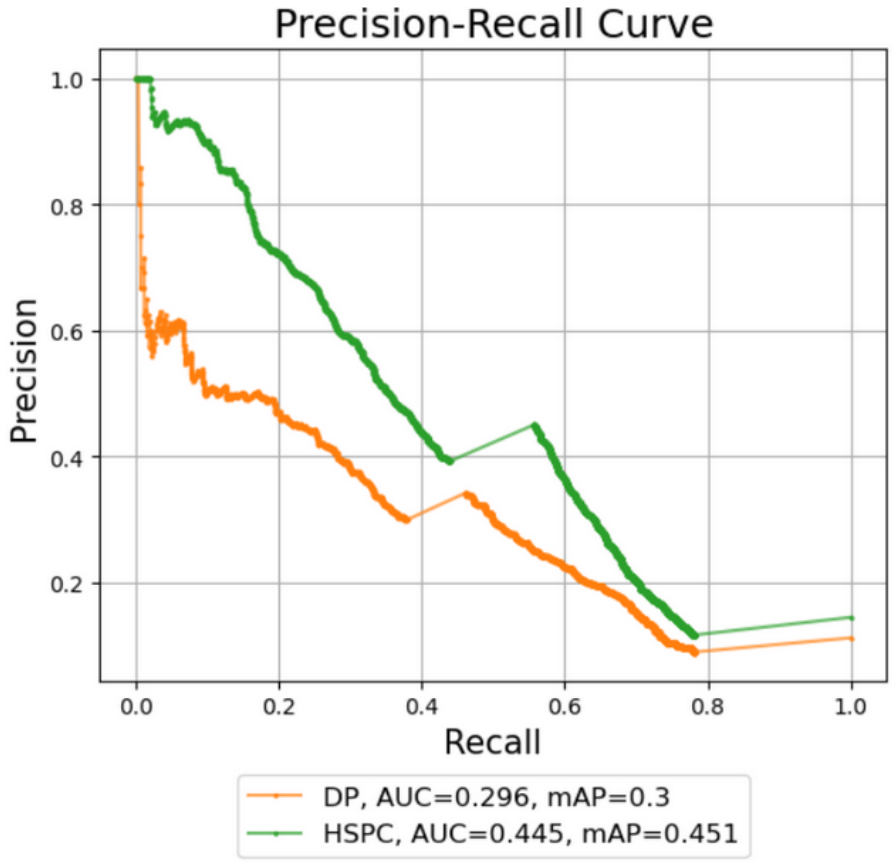

Figure 3.12: Precision-Recall curve for Faster-RCNN with patches extracted along the z-axis.

This PR curve (Figure 3.12) showed promise compared to YOLOv5. This classifier was likely better for HSPCs than DPs as evidenced by the high area under the curve.

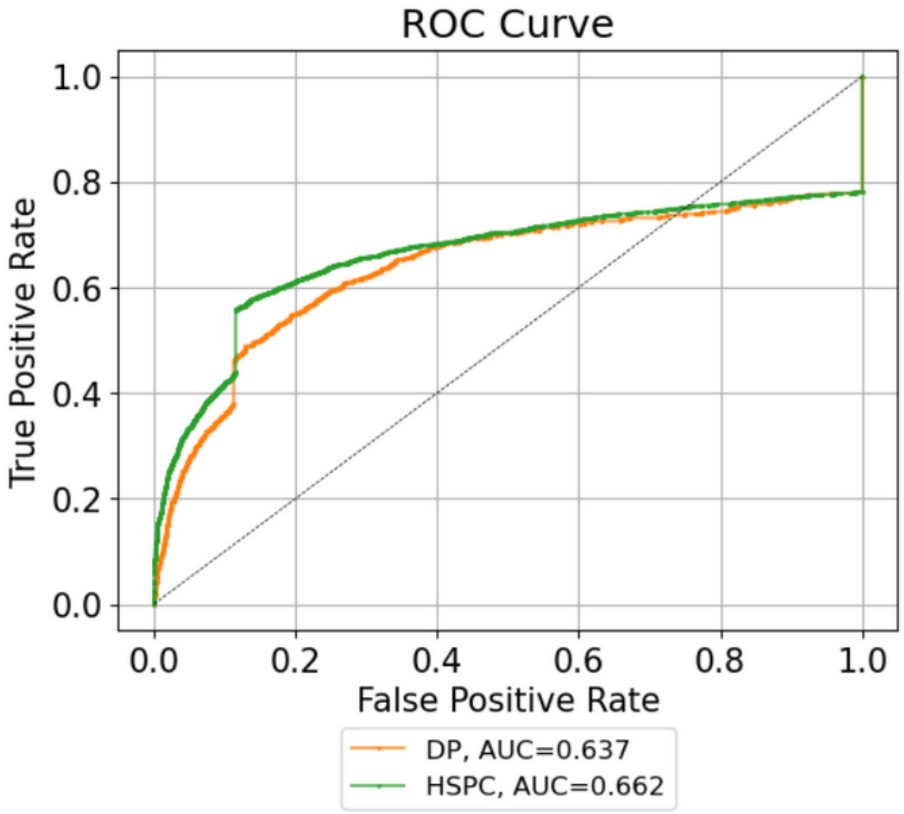

Figure 3.13: ROC curve for Faster-RCNN with patches extracted along the z-axis.

The ROC curve was encouraging as the curves, for FPRs and TPRs below 0.8, were above FPR=TPR for both classes. The AUCs are also high.

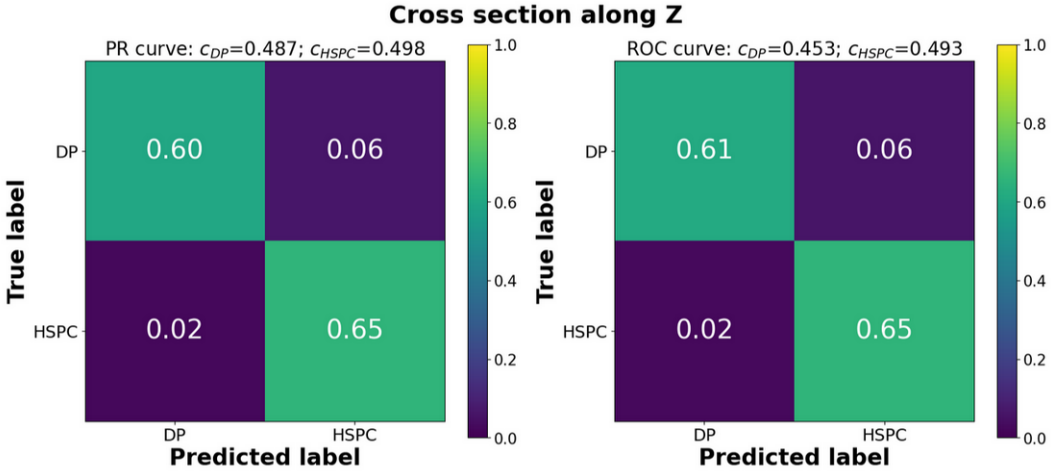

Figure 3.14: Confusion matrix for Faster-RCNN with patches extracted along the z-axis. c denotes the confidence threshold.

Youden's J statistic [24] was used to identify the optimal confidence threshold for the ROC curve. The threshold for the PR curve was found by finding the confidence threshold for which F1 was maximized. The confusion matrices indicated a minimum rate of 60% of true positives identified. This applied both classes and to confidence thresholds found from the PR and ROC curves.

Table 3.1: Faster-RCNN Metrics for PR and ROC categories with DP and HSPC subcategories.

	PR Thresholds		ROC Thresholds	
	DP	**HSPC**	**DP**	**HSPC**
TPR	0.601	0.653	0.612	0.653
FNR	0.399	0.347	0.388	0.347
FPR	0.179	0.593	0.193	0.579
Precision	0.297	0.393	0.285	0.391
Recall	0.601	0.653	0.612	0.653
F1-score	0.397	0.491	0.389	0.489

Across both methods of computing confidence thresholds and for both cell classes, Faster-RCNN yielded a better recall than precision, which was low. HSPCs had a higher precision than DPs whereas recalls were similar. The TPRs and FNRs were similar across classes. HSPCs had a significantly higher FPR than DPs.

3.4 RetinaNet

Similar to Faster-RCNN, RetinaNet was trained with a stopping criteria applied to the validation loss: if the total validation loss did not improve (decrease) for 10 epochs, then training would be stopped. The trend in validation loss was calculated using the running average across 5 epochs. This running average would be evaluated for the previous 10 epochs to determine whether the stopping criteria was met. This is illustrated in Figure 3.10

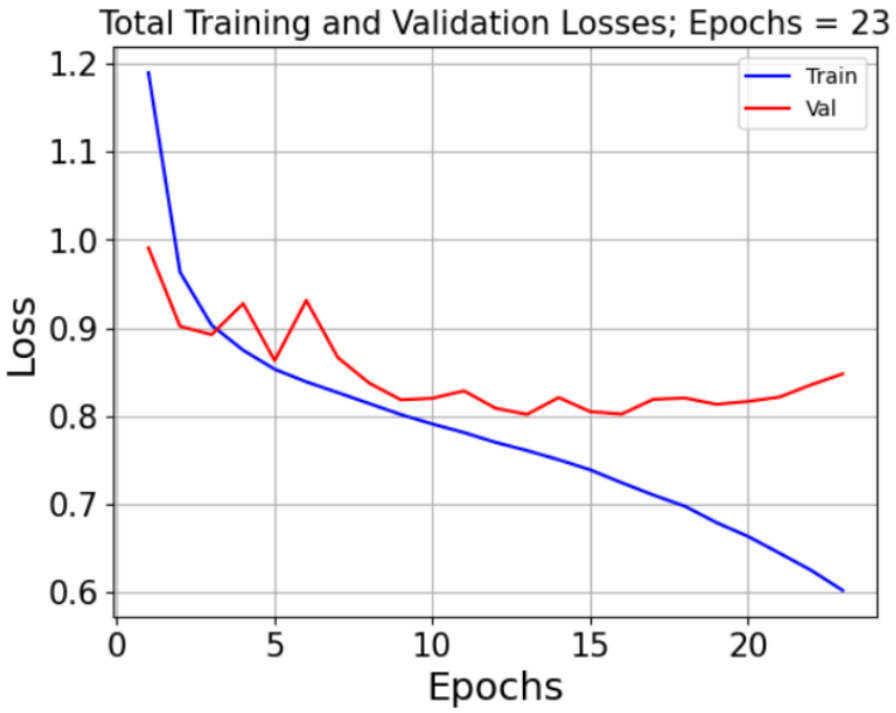

Figure 3.15: Training and validation losses for RetinaNet with patches extracted along the z-axis.

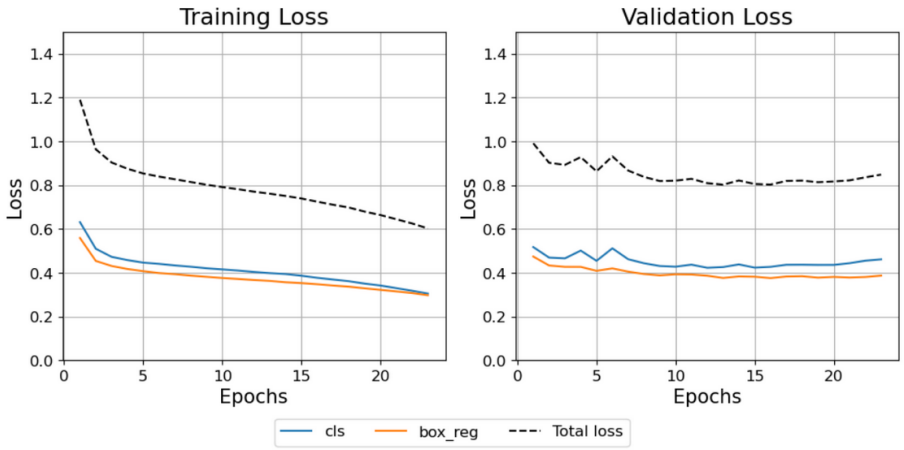

Figure 3.16: Breakdown of the training and validation losses for RetinaNet with patches extracted along the z-axis.

Figures 3.15 and 3.16 indicate that the model achieved a stable validation loss, with room for improvement for the training loss.

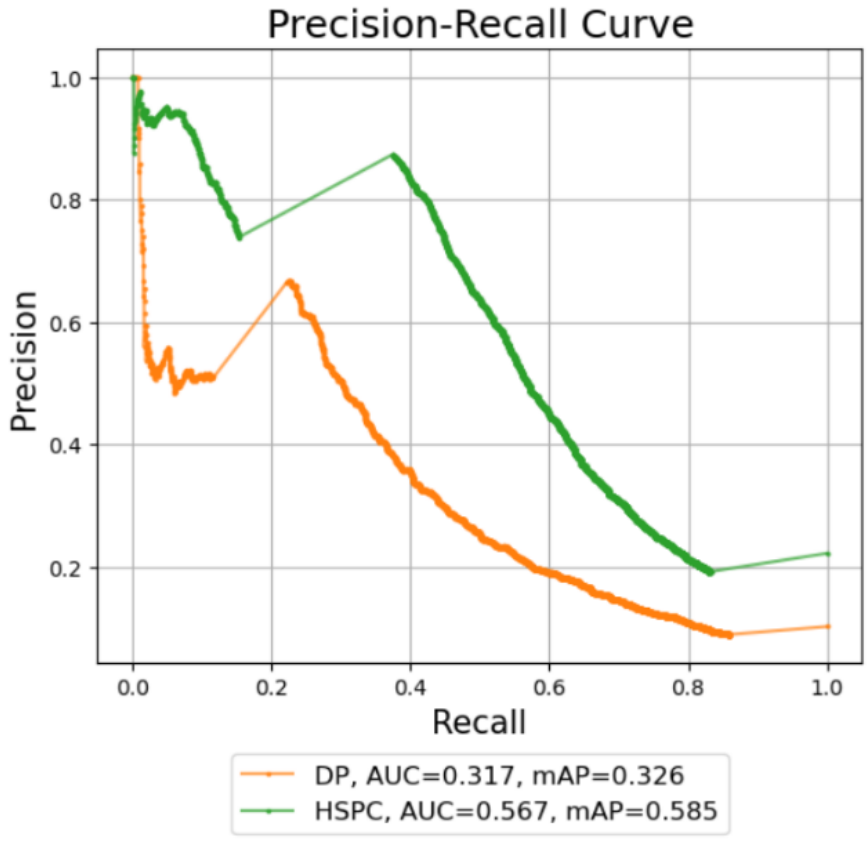

Figure 3.17: Precision-Recall curve for RetinaNet with patches extracted along the z-axis.

This PR curve (Figure 3.17) was not as encouraging as it was for Faster-RCNN (Figure 3.12), as the areas under the curve were lower. It is likely that the HSPCs will have a high FPR. This classifier was likely better for HSPCs than DPs as evidenced by the higher area under the curve.

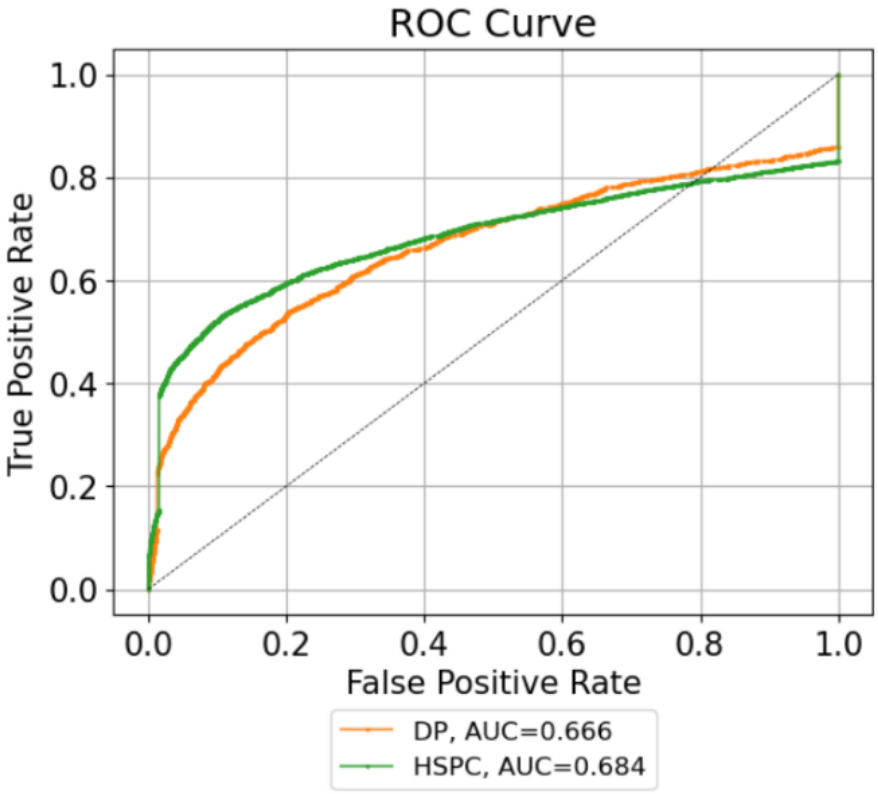

Figure 3.18: ROC curve for RetinaNet with patches extracted along the z-axis.

The ROC curve was encouraging as the curves, for FPRs and TPRs below 0.8, were above FPR=TPR for both classes. The AUCs are also high.

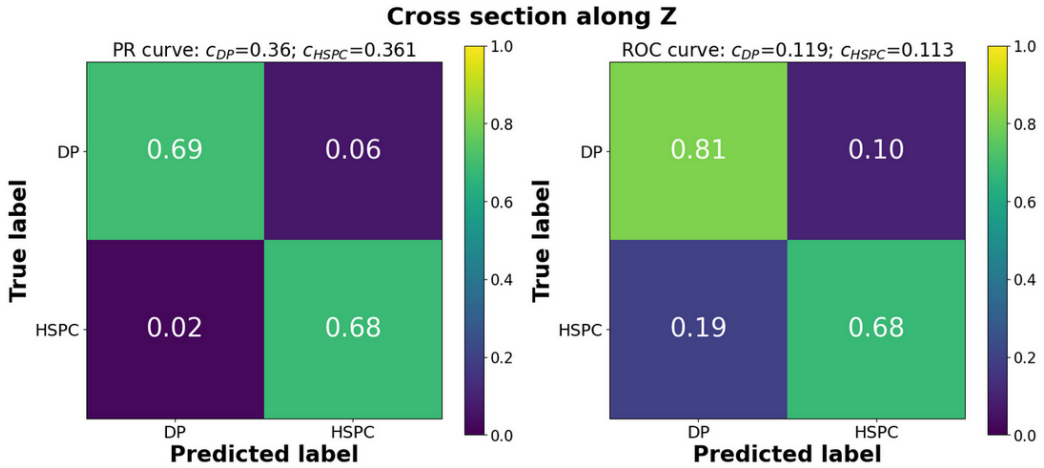

Figure 3.19: Confusion matrix for RetinaNet with patches extracted along the z-axis. c denotes the confidence threshold.

Youden's J statistic [24] was used to identify the optimal confidence threshold for the ROC curve. The threshold for the PR curve was found by finding the confidence threshold for which F1 was maximized. The confusion matrices indicated a similar proportion of true positives identified compared to Faster-RCNN. The threshold found via the ROC curve was especially encouraging for DPs, as the proportion of true positives was 0.81.

Across both methods of computing confidence thresholds and for both cell classes, RetinaNet

Table 3.2: RetinaNet Metrics for PR and ROC categories with DP and HSPC subcategories.

	PR Thresholds		ROC Thresholds	
	DP	**HSPC**	**DP**	**HSPC**
TPR	0.692	0.683	0.807	0.676
FNR	0.308	0.317	0.193	0.324
FPR	0.177	0.505	0.367	0.569
Precision	0.370	0.538	0.104	0.177
Recall	0.692	0.683	0.807	0.676
F1-score	0.482	0.602	0.184	0.281

yielded a better recall than precision, which was low. HSPCs had a higher precision than DPs whereas recalls were similar. The TPRs and FNRs were similar across classes. HSPCs had a significantly higher FPR than DPs. The confidence thresholds found via ROC were inferior to PR when it came to the precisions.

Chapter 4

Discussion and Conclusion

4.1 Performance of NoduleNet

NoduleNet, despite its initial promise based on its success in other domains, struggled significantly with the dataset used in this study. The classification loss plateaued early during training, and extensive troubleshooting did not yield improvements.

This suggests that the architecture of NoduleNet may not be well-suited to the particular characteristics of 3D fluorescence microscopy images of bone marrow or the amount of data available seemed to be unsuitable for this model.

The inability to effectively distinguish between HSCs and progenitor cells limits its applicability in this context. This performance could also be attributed to the limited size of the training dataset or the lack of features specific to the microscopy modality used.

4.2 Performance of YOLOv5

YOLOv5 also demonstrated difficulties in achieving stable training and exhibited signs of overfitting, as depicted by the divergence of training and validation losses. The poor performance in classification tasks, particularly in distinguishing between closely related cell types, suggests that YOLOv5 may not be the best choice for this application.

The low precision and recall observed indicate that the model is likely to produce many false positives and false negatives, which could compromise the accuracy of cell population analyses. The challenges observed with YOLOv5 highlight the need for potentially, more sophisticated techniques to handle the complexity of 3D biological images.

4.3 Performance of Faster R-CNN

Faster-RCNN showed a significantly more balanced performance compared to NoduleNet and YOLOv5, with relatively stable training and validation losses.

However, the model still exhibited a significant false positive rate, particularly in classifying Double Positive (DP) cells which remains undesirable. This suggests that while Faster-RCNN is better suited to the task, there is still a risk of mis-classification which could impact downstream analyses, such as quantifying the differentiation pathways of HSCs.

The model's balanced performance makes it a viable candidate for refinement, possibly through enhancements in training data or the use of further ensemble methods to reduce the false positive rate.

4.4 Performance of RetinaNet

Among the models tested, RetinaNet demonstrated the best overall performance, particularly in terms of recall, making it less likely to miss true positive detections. It still faced certain challenges in accurately identifying DP cells, which could lead to incorrect conclusions about cell differentiation processes.

RetinaNet's relatively strong performance suggests that it could be particularly useful in scenarios, where accurate and comprehensive detection is critical. However, there is scope for further optimization of the parameters to make it more robust for distinguishing very closely related cell types and adding additional contextual information or extra features to further improve the performance. Overall, RetinaNet had very good results for DPs. This is encouraging as there were far fewer DPs relative to HSPCs in the dataset. Training with a larger set of DPs could make RetinaNet a very good candidate for the CNA Lab to identify differentiating hematopoietic stem cells.

4.5 Implications for future research

The findings from this study have several important implications for future research in the field of computational biology. The limitations observed with the current models indicate a need for tailored approaches that better address the specific challenges posed by 3D fluorescence microscopy data. This could involve the development of new architectures which can better handle the spatial and intensity variations inherent in such data.

Secondly, the study highlights the critical role of dataset size especially for training such intensive models and diversity in training effective models. Expanding the dataset through collaboration or synthetic data generation could potentially improve model performance. Additionally, the use of transfer learning and ensemble methods could help mitigate the challenges posed by small datasets and improve generalization to unseen data.

Finally, the integration of biological context into model development and evaluation could lead to more accurate and biologically meaningful results. Multi-modal data or advanced post-processing techniques could further be used to incorporate deeper insights from the field of cell biology and tissue architecture, leading to added features, adding further complexity to the dataset and potentially reducing the problem of overfitting.

In summary, though the models like RetinaNet and Faster R-CNN showed significant results in classification, there is scope for improvement due to the aforementioned challenges. Addressing the causes identified here could be crucial for advancing the application of deep learning to the analysis of complex biological images, improving the currently available models and gaining more accurate insights into the cellular mechanisms underlying hematopoiesis and related processes.

Bibliography

[1] GEO Accession Viewer. Single cell rna-seq analysis of young and old mouse hematopoietic stem and progenitor cells, 2021.

[2] E. Duprez et al. Single cell rna-seq analysis of young and old mouse hematopoietic stem and progenitor cells, 2020.

[3] P. Frenette and Q. Wei. Loss of adrenergic nerves in bone marrow drives hematopoietic stem cell niche aging, 2018.

[4] Hematopoietic cytokines for cardiac repair: Mobilization of bone marrow cells and beyond. *Basic Research in Cardiology*, 2022.

[5] X. Hao et al. The impact of remote solid tumors on bone marrow ecosystem at single-cell resolution, 2021.

[6] V7 Labs. Object detection: The ultimate guide. V7 Labs Blog. Accessed: 2024-08-30.

[7] Ö. Çiçek, A. Abdulkadir, S. S. Lienkamp, T. Brox, and O. Ronneberger. 3d u-net: Learning dense volumetric segmentation from sparse annotation. In *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2016*, pages 424–432, 2016.

[8] Y. Zhou and O. Tuzel. Voxelnet: End-to-end learning for point cloud based 3d object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 4490–4499, 2018.

[9] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(6):1137–1149, 2015.

[10] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollar. Focal loss for dense object detection. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 2980–2988, 2017.

[11] Ultralytics. Yolov5 architecture description, 2024.

[12] Yolov5: Real-time object detection algorithm. *Sensors*, 22(15):5817, 2022.

[13] World Precision Instruments. Ca^{2+} detection in muscle tissue using fluorescence spectroscopy. WPI Blog, 2024. Accessed: 2024-08-30.

[14] Lightsheet microscopy. National Center for Biotechnology Information (NCBI). Accessed: 2024-08-30.

[15] Zhe Li Xiaolin Hu Fangzhou Liao, Ming Liang and Sen Song. Evaluate the malignancy of pulmonary nodules using the 3-d deep leaky noisy-or network. *IEEE Transactions on Neural Networks and Learning Systems*, 30(11):3484–3495, 2019.

[16] 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 (CVPR)*, pages 770–778, 2016.

[17] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning (ICML)*, pages 448–456, 2015.

[18] 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 (CVPR)*, pages 390–391, Seattle, WA, USA, June 2020. IEEE.

[19] M. Patel and J. K. Sing. Performance evaluation of deep cnn-based crack detection and localization techniques for concrete structures. *ResearchGate*, 2021. Accessed: 2024-08-30.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Feature pyramid networks for object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 936–944, 2017.

[21] V7 Labs. Intersection over union (iou) guide. https://www.v7labs.com/blog/intersection-over-union-guide, 2024. Accessed: 2024-08-31.

[22] AppsFlyer. Click flooding detection: The false positive challenge. https://www.appsflyer.com/blog/mobile-fraud/click-flooding-detection-false-positive-challenge/, 2024. Accessed: 2024-08-31.

[23] Google Developers. Roc and auc — machine learning crash course. https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc, 2024. Accessed: 2024-08-31.

[24] M et al Ruopp. Youden index and optimal cut-point estimated from observations affected by a lower limit of detection. *BioM J.*, 2008. Accessed: 2024-08-31.

[25] Scikit-learn. Precision-recall — scikit-learn documentation. https://scikit-learn.org/stable/auto_examples/model_selection/plot_precision_recall.html, 2024. Accessed: 2024-08-31.

[26] Douglas Steen. Precision-recall curves. https://medium.com/@douglaspsteen/precision-recall-curves-d32e5b290248, 2024. Accessed: 2024-08-31.

[27] GeeksforGeeks. Confusion matrix in machine learning. https://www.geeksforgeeks.org/confusion-matrix-machine-learning/, 2024. Accessed: 2024-08-31.

[28] PyTorch Contributors. Pytorch: An imperative style, high-performance deep learning library. https://pytorch.org/, 2024. Accessed: 2024-08-31.

[29] University of Zurich. Sciencecluster - high performance computing at the university of zurich. https://www.zi.uzh.ch/en/teaching-and-research/science-it/computing/sciencecluster.html, 2024. Accessed: 2024-08-31.

[30] TorchVision Contributors. Faster r-cnn implementation in torchvision. https://pytorch.org/vision/stable/models.html#faster-r-cnn, 2024. Accessed: 2024-08-31.

[31] TorchVision Contributors. Retinanet implementation in torchvision. https://pytorch.org/vision/stable/models.html#retinanet, 2024. Accessed: 2024-08-31.

[32] Bitplane AG. Imaris software for 3d and 4d image analysis. https://imaris.oxinst.com/, 2024. Accessed: 2024-08-31.

[33] Colin J. Williams et al. h5py: A python interface to the hdf5 binary data format. http://www.h5py.org/, 2024. Accessed: 2024-08-31.

[34] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in python, 2011.

[35] J. D. Hunter. Matplotlib: A 2d graphics environment, 2007.

[36] C.R. Harris, K.J. Millman, S.J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N.J. Smith, R. Kern, M. Picus, S. Hoyer, M.H. van Kerkwijk, M. Brett, A. Haldane, J.F. del Río, M. Wiebe, P. Peterson, P. Gérard-Marchant, K. Sheppard, T. Reddy, W. Weckesser, H. Abbasi, C. Gohlke, and T.E. Oliphant. Array programming with numpy, 2020.

[37] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.