Leveraging CNN for Automated Peak Picking in Untargeted Metabolomics without Parameter Dependencies

Master's thesis in Applied Data Science

Vivian Wang and Lidia Yalew

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Leveraging CNN for Automated Peak Picking in Untargeted Metabolomics without Parameter Dependencies

Vivian Wang, Lidia Yalew

Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
Gothenburg, Sweden 2023

Leveraging CNN for Automated Peak Picking in Untargeted Metabolomics without Parameter Dependencies
VIVIAN WANG, LIDIA YALEW

Supervisors:
Carl Brunius, Department of Life Sciences
Gabriel Reder, Department of Computer Science and Engineering
Examiner: Peter Damaschke, Department of Computer Science and Engineering

Master's Thesis 2023
Department of Computer Science and Engineering
Division of Data Science and AI
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Leveraging CNN for Automated Peak Picking in Untargeted Metabolomics without Parameter Dependencies
Vivian Wang, Lidia Yalew
Department of Computer Science and Engineering
Chalmers University of Technology

0.1 Abstract

Metabolomics is a scientific discipline that involves the thorough analysis of small molecules, known as metabolites, found within a biological system. Furthermore, liquid chromatography-mass spectrometry (LC-MS) is a commonly used analytical technique in metabolomics for analysing biological samples due to its broad coverage of the measurable metabolome. The technique is widely used and generates a large amount of raw data, covering a broad spectrum of metabolites. Consequently, there is a need to transform this raw data into a structured, tabular format that can be readily utilised for further analysis. Despite the existence of software tools offering automated peak detection, the necessity for visual inspection and manual corrections frequently arises, a process that is both time-consuming and demands specialised knowledge in the respective domain. In order to enhance data processing efficiency and automation, this project establishes, optimises, and evaluates a deep learning approach to perform regions of interest (ROI) detection utilising faster Region-based Convolutional Neural Network (R-CNN). The integration of deep learning within LC-MS analysis has the potential to enhance the overall efficiency and accuracy of metabolomic studies. Moreover, it can assist in constructing reliable predictive models for diverse LC-MS applications.

The ROI detection was performed on reversed-phased positive liquid chromatography provided by the Chalmers Mass Spectrometry Infrastructure (CMSI). The model underwent training using a dataset comprising 524 chromatograms, followed by evaluation using a separate set of 151 chromatograms. The model takes segments of a chromatogram as inputs and generates predicted coordinates as outputs, indicating the locations of the ROIs. The evaluation of the results was conducted through both quantitative and qualitative analyses, using precision and recall, F1-score, Intersection over Union as well as manual inspection. The results were an average precision of 0.591, an average recall of 0.648, an F1-score of 0.617 and a mean IoU of 0.558. The findings demonstrate promising outcomes with substantial potential.

Keywords: Deep learning, Convolutional neural network, LC-MS, Peak Detection, Metabolomics, Faster R-CNN.

Acknowledgements

We would like to express our sincere gratitude to our supervisors, Carl Brunius and Gabe Reder, for their invaluable guidance, expertise, and continuous support throughout the process of conducting this research and writing this thesis. Their insightful feedback, patience, and encouragement have been instrumental in shaping the direction and quality of this work. Lastly, we would like to acknowledge Peter Damaschke, our examiner, for his careful evaluation of our thesis and for providing valuable comments and suggestions that helped improve the final version.

Vivian Wang, Lidia Yalew, Gothenburg, June 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ANN	Artificial Neural Networks
AP	Average Precision
CMSI	Chalmers Mass Spectrometry Infrastructure
CNN	Convolutional Neural Networks
FC	Fully Connected
GC-MS	Gas Chromatography-Mass Spectrometry
GDPR	General Data Protection Regulation
IoU	Intersection over Union
LC-MS	Liquid Chromatography-Mass Spectrometry
MAE	Mean Absolute Error
mAP	mean Average Precision
MRE	Mean Relative Error
m/z	mass-to-charge-ratio
NMS	Non-Maximum Suppression
QC	Quality Control
qTOF	quadrupole Time-Of-Flight
R-CNN	Region-based Convolutional Neural Network
RPLC	Reversed-Phase Liquid Chromatography
RPN	Region Proposal Network
YOLO	You Only Look Once
ROC-AUC	Area Under The Curve-Receiver Operating Characteristics
rt	retention time
UHPLC	Ultra-High-Performance Liquid Chromatography
XIC	Extracted Ion Chromatograms

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Metabolomics refers to the systematic investigation and appraisal of metabolite profiles, reflecting biochemical processes associated with e.g., exposures (e.g., diet, physical activity, or medication) and/or health (e.g., disease conditions or intermediate risk markers such as cholesterol or blood pressure) [1]. This is achieved through the measurement and quantitation of molecules involved in metabolic reactions. The field of metabolomics has grown steadily in the last two decades mainly due to advances in analytical instrumentation as well as tools and algorithms suitable for pre-processing and analysis of high-dimensional data [2]. Nationally and globally, infrastructure units such as the Chalmers Mass Spectrometry Infrastructure (CMSI) perform liquid chromatography-mass spectrometry (LC-MS) based metabolomics analyses of tens of thousands of samples per year, projected to increase steadily. There is thus a continuous need for the development of adequate computational tools for robust, accurate, and automatable processing of instrument data into actionable (tabularised) format.

LC-MS is an instrument analytical technique widely applied in metabolomics to analyse biological samples, since it provides wide coverage of the measurable metabolome, i.e., the comprehensive pool of measurable metabolites. The technique has gained popularity due to high sample throughput, high sensitivity, and broad coverage of metabolites [3]. For each sample, this technique generates raw instrument data in a 3D topological map structure consisting of retention time (from chromatographic separation), the mass-to-charge ratio (from mass spectrometric analysis), and signal intensity [4], where the mass-to-charge ratio is an inherent property of a molecule's chemical structure. Post-acquisition raw data processing analysis thus generates an abundance of complex data that is rich in chemical information.

A crucial aspect of this process is to process the data in a manner that allows for the efficient extraction of relevant information. To automate the pre-processing of the raw data into tabular format, there have been several algorithms developed. These algorithms commonly consist of two steps, (1) peak picking and (2) alignment. In the process of peak picking, a comprehensive inventory of peaks identified by their corresponding mass and retention time is generated. A peak refers to a three-dimensional (m/z, retention time, and intensity) LC-MS signal. Each entry in the list is assigned a signal intensity value, which signifies the area (or sometimes height) of the peak. To maintain consistency across samples, alignment techniques are employed to correct for differences in retention time and mass values, resulting in the representation of each peak, as a single chemical compound, by uniform mass and retention time values across all samples. This commonly found analyte after aligning peaks to a common grid across all samples is called a feature. The result of the peak detection and the alignment is represented in a table presenting the common features [5].

However, there are some limitations to current peak picking algorithms such as manual algorithm parameterisation of several mostly non-intuitive (black box) parameters which can lead to uncertain impacts on data quality [6]. Furthermore, parameter tuning is both time-consuming and requires domain knowledge expertise [7]. This presents the opportunity for further development of peak-picking algorithms which address these limitations. This thesis focuses on addressing current issues in peak picking, by aiming to develop an algorithm capable of accurately detecting regions of interests (ROIs) while requiring little-to-no parameter optimisation.

1.2 Problem

To achieve optimal efficiency in peak picking, currently employed software tools such as XCMS and MZmine require appropriate parameter configurations that align with the structure of the given dataset. In practice, this requires visual inspection and manual corrections, although computer-assisted optimization algorithms can aid in this procedure [8]. The optimisation of this parametrisation is a complex task that requires balancing the need to identify as many true features as possible with the need to minimise false positives [7]. These false peaks are neither contaminants nor low-quality peaks and their differentiation from true positive features via conventional filtering methods or control samples is difficult. Therefore, they typically have to be removed through manual review [6]. Approaches to eliminate the need for manual parameter tuning have been made, of which some attempts include CNN-based peak detection [9] and peak qualification [10]. Despite the promising results yielded by these approaches, they were also found to have certain limitations. These limitations lie in the sensitivity and general applicability as well as in the narrow training material, i.e., the high degree of manual curation of the training material. In addition, several of these approaches suffer from a lack of computational power, requiring up to one hour per sample, effectively rendering algorithms impractical for large-scale operations [9]. These drawbacks offer an opportunity to enhance the peak identification algorithms.

Peak detection is one of the defining steps in the LC-MS data processing pipeline. The algorithms used for peak detection are, however, primarily designed to be highly sensitive to detect peaks, i.e., to mitigate false-negative peak detection. Furthermore, this high sensitivity can lead to a high inclusion of false-positive, normally low-intensity peaks. This inclusion of false-positive peaks will complicate downstream data processing and analysis [2] [7]. An additional concern about the efficiency of peak picking algorithms is that they usually operate on the premise that the parameters remain constant within the 3D topological map of a single sample as well as across multiple samples within the same analytical batch. Nevertheless, this assumption is often not reflected in practice. Furthermore, conventional peak-picking algorithms operate by dividing the m/z axis into discrete intervals, known as bins, and identify peak shapes in the resulting two-dimensional chromatograms with retention time on the x-axis and bin intensity on the y-axis [11]. Nonetheless, binning can have possible disadvantages. If the bin divisions are not chosen with precision, the detection of a particular analyte may be distributed across two adjacent bins. Moreover, the presence of multiple distinct analytes may combine to result in a single bin's measurement, hindering the ability to distinguish significant variations among individual analytes [12].

1.3 Aim

1.3.1 Research Question

Within the context of LC-MS data pre-processing for metabolomics purposes, this thesis focuses on the identification of peaks in a complex 3D data structure. More specifically, on the issue of ROI detection with little-to-no pre-determined parameters. ROIs are defined by their respective coordinates, related to distinct peaks located on the mass-to-charge ratio (m/z) and retention time (rt) axes.

1.3.2 Objectives

The main aim of this project is to train a convolutional neural network (CNN) to identify distinctive chromatographic peak patterns directly from 3D data, thus reducing or even eliminating the need for manual parameter adjustments. This deep learning algorithm should circumvent the issue of assuming parameter stability within and between injections, which should greatly simplify the pre-processing of LC-MS data. Such a parameter-free peak selection approach thus has vast potential to significantly reduce resource requirements in terms of domain expertise and time, thereby improving the automation of bioinformatics pipelines to accelerate the delivery of metabolomics data to end users. Finally, the goal is to contribute to open science.

2
Theory

The primary objective of this chapter is to establish a foundational comprehension of the subject matter and the underlying models employed in the project. This is accomplished by introducing essential concepts that are crucial for interpreting the outcomes presented in the thesis.

2.1 Pre-processing

Methods and tools used in metabolomics generate large quantities of data. Handling this complex metabolomic data is of crucial importance as it has an impact on the extent and quality at which the identification and quantification of the metabolite can be made [13]. An LC-MS data file consists of sequentially recorded histograms, commonly referred to as scans, where each histogram represents hits of ionized molecules (a collection of m/z and intensity data points) on the detector during a short time frame [14]. The main purpose of the pre-processing step is to transform the raw data into formats (typically tabularised) that are easy to use in the consecutive data analysis steps. While approaches that enable data pre-processing of untargeted metabolomics keep being improved, the general steps have persisted across the different tools, see Figure 2.1 [15].

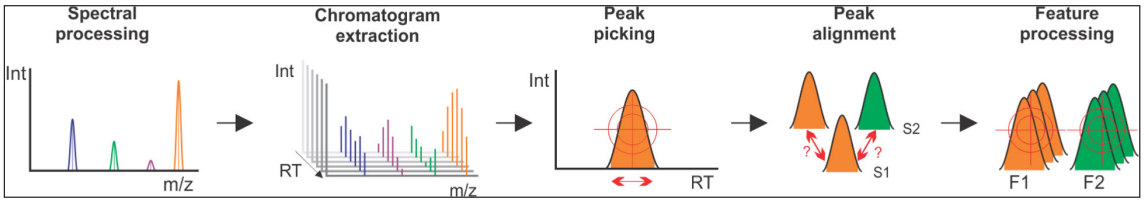

Figure 2.1: General steps of nontargeted data pre-processing [15].

Data management in metabolomics is often divided into two steps: data pre-processing and data analysis as seen in Figure 2.2. For this project, the focus is on the data pre-processing part. The data pre-processing part consists of filtering, feature detection, alignment, and normalisation. Downstream data analysis includes both statistical as well as machine learning-based analyses for the discovery of features differentiating between sample groups or clustering of individuals into metabolite profiles [13]. The subsequent sections about the pre-processing of LC-MS data will be explained based on the structure and references provided by Katajamaa and Orešič's paper from 2007 [13].

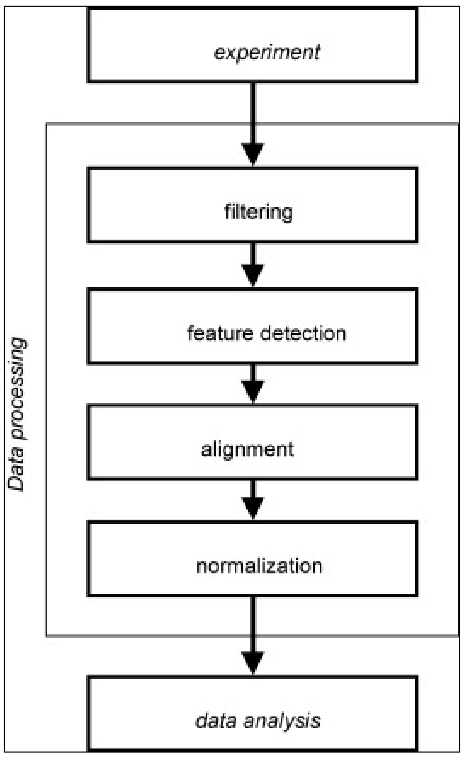

Figure 2.2: Summary of metabolomic data processing workflow [13].

2.1.1 Raw Data Pre-Processing and Filtering

Raw instrument data at the point of detection has complete coverage of intensities across the entire m/z range. However, this takes a lot of space, therefore, centroiding is frequently performed, such that Gaussian shapes in each scan are represented only by the peak m/z and not the entire data. Centroided data is frequently used in the data pre-processing step as it allows for the creation of smaller and easily manageable data files. However, centroiding may also attribute to complications pertaining to noise level estimation [16]. This centroiding effectively makes the data much sparser and can be applied already at data generation or in later processing steps, e.g., when converting raw instrument files in vendor format to open file formats, such as mzML [13]. This thesis project is geared toward the processing of centroided data, which is the data type employed at CMSI and used within the project.

Filtering methods are used to process the raw measurement signals with the aim of removing measurement noise or baseline. LC-MS data has both random noise and chemical noise. Random noise is most often created by the detector whereas chemical noise usually is caused by molecules in buffers and solvents that may be particularly strong at the beginning and the end of the elution [14]. To counteract the presence of noise, noise reduction methods are used. Common practice is to implement traditional signal processing techniques. Some of these techniques are wavelet transformation [17] and polynomial smoothing filter, e.g. based on the Savitsky-Golay algorithm [18]. Baseline removal is often conducted in a two-step process: (1) finding the baseline shape and; (2) subtracting the shape from the raw signal [13].

2.1.2 Feature Detection

The main aim of the feature detection step is to identify all measurable true peaks while minimising false positives. Furthermore, the objective is also to offer as quantitively accurate information about the ion concentrations as possible. There are three main strategies for performing feature detection: (1) detection in two directions (i.e., in rt and m/z), (2) detection through XIC slices (i.e., in rt only through binning of the m/z axis), and (3) detection through model fitting [13], see Figure 2.3.

The method of detection in two directions means that peaks are detected independently in both the retention time direction as well as the m/z direction. The peaks are detected if their intensities are above a certain threshold [16] [18]. Additionally, Bellew et al. [19] developed another two-direction method that operates in three steps: (1) Using wavelet additive decomposition to find local maxima within each scan, (2) identify peaks that are sustained over multiple scans as well as smooth peaks over time, (3) assemble all the peaks into different isotope groups which appear, maximises and disappears at the same time [13].

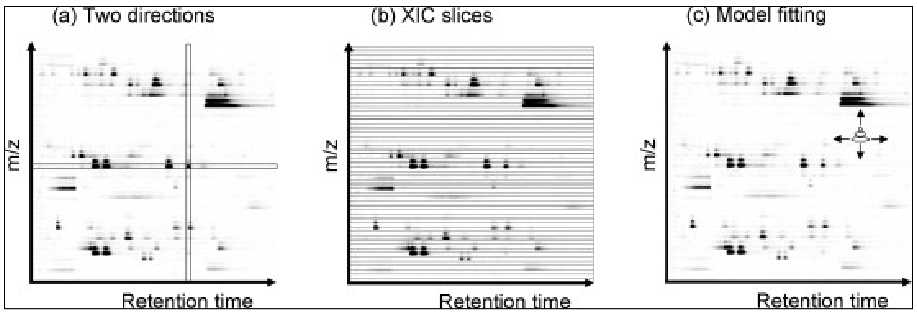

Figure 2.3: Examples of peak detection strategies [13].

The second strategy, detection through XIC slices, operates by slicing the data into extracted ion chromatograms (XIC), where each slice covers a narrow m/z range and thus avoids the issue of searching for peaks in the m/z direction. The chromatograms can further be processed using a second-order Gaussian filter to identify peak inflections points for integration [12] or by using the mean or median of the chromatogram to calculate a threshold level and thereafter search for areas in the chromatogram above the threshold level [20].

The third and last strategy is detection through model fitting against the raw signal. One example is by fitting a three-dimensional model of a generic isotope pattern to the highest peak in raw signal and deducting the fit from signal [21]. This method may be advantageous as it could improve detection results by reducing the amount of detected false positives [13].

2.1.3 Alignment

The purpose of the alignment step is to correct any differences in retention time across the varied sample runs as well as combine the data from the different samples.

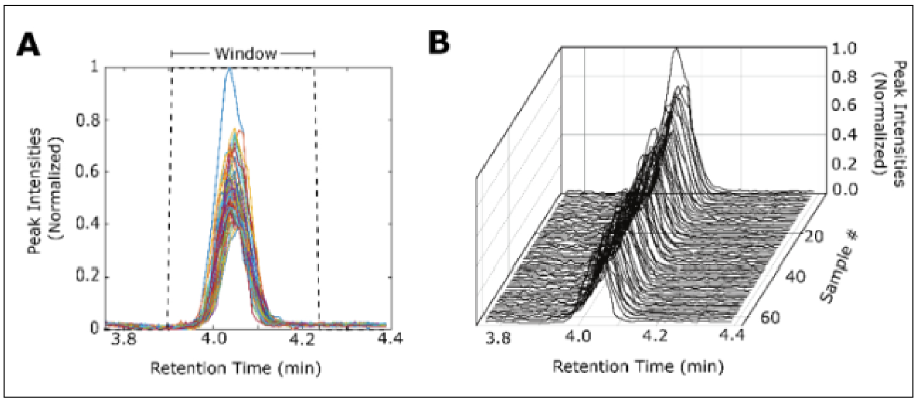

Figure 2.4: Alignment of peaks in 2d (A) and 3d (B). The Figure presents a set of stacked peak plots, which were generated using peak information extracted from 63 samples where the candidate peak exhibited the highest intensity. The stacked peak plots are used to represent the structure of each extracted feature, whereas, in the right graph, a 3D isometric representation is used for clearer representation [6].

There are a few alignment methods but the majority of them operate in a pairwise manner. Either through aligning pairs of samples or by aligning several samples against a chosen reference sample or a pattern and typically, the selection of the reference sample affects the alignment results. There are two categories of methods: (1) inputting raw data and generating a collection of mappings that transform the retention time axis of the different runs into one common retention time axis or (2) clustering detected features and producing a matrix where every row match to a cluster and the columns contains a measurement for every sample, e.g., peak area. Some alignment methods combine both categories 1 and 2. Furthermore, the decision on which alignment method to use typically has a direct effect on what kind of downstream data analysis that is required. If the first category is chosen, it would be required to compare the raw aligned signals to find differences among the samples. However, if the second category is chosen, multivariate analysis is needed [13].

2.1.4 Normalisation

Normalisation is used to eliminate undesired systematic bias in ion intensities among measurements while also preserving the interesting biological variation. There are two categories of strategies for normalisation: derive optimal scaling factors for each sample based on the complete dataset using statistical models and normalisation by a single or multiple internal or external standard compounds according to empirically derived guidelines, such as specific retention time intervals [13]. Normalisation is required as the signal intensity fluctuates between injections within an analytical batch (e.g., due to the build-up of contaminations in the ion source) as well as between batches (e.g., that cleaning procedures between batches are not equally effective in returning the instrument to a similar status). In addition, there is wear and tear on instrumentation contributing to a combination of random and systematic drift in sensitivity over time.

2.2 Deep Learning

Artificial Neural Networks (ANN) have their name and architecture inspired by the human brain. An ANN emulates a biological neural network however they use a smaller group of principles derived from the biological neural systems. ANNs simulate the electrical activity of the nervous system and the brain. Processing entities such as nodes or perceptrons are connected with other processing elements. Nodes are usually organized in a layer or vector. The output from one-layer acts as input to the next layer, as well as potentially to other layers. A node may be linked to all or some of the nodes in the subsequent layer where each such connection has an associated threshold and weight, mimicing the synaptic connections in the brain. Furthermore, a node is said to be activated when the output of this node is above its specific threshold, thus sending the data to the next layer within the network. It imitates the electrical excitation of a nerve cell, which leads to the transfer of information within the network, or in the brain [22]. If the output doesn't reach its threshold value, no data is passed along [23].

Convolutional neural networks (CNN) are a class of ANNs often used for classification and computer vision tasks. ANNs are one of the simpler variants of neural networks where information is passed in one direction through various input nodes until the output nodes are reached. CNNs on the other hand use variations of multilayer perceptron and they contain one or more convolutional layers [24]. CNNs are distinguished from other ANNs by their outstanding performance with speech, audio signals, and image inputs (IBM 2, n.d.). This is because the network specialises in processing data that has grid-like data [25]. Furthermore, their use has been proven successful in other medical research areas such as diabetes and skin cancer research [26] [27]. The network typically has three main types of layers: convolutional layers, pooling layers, and fully connected (FC) layers. The first two-layer types, conduct feature extraction, whereas the FC layer maps these extracted (latent) features into a final output, such as classification. Whilst one layer's output is being fed to the next layer, the extracted features can gradationally become increasingly complex [24]. A typical CNN architecture can consist of several convolutional and pooling layers followed by one or more FC layers as seen in Figure 2.5.

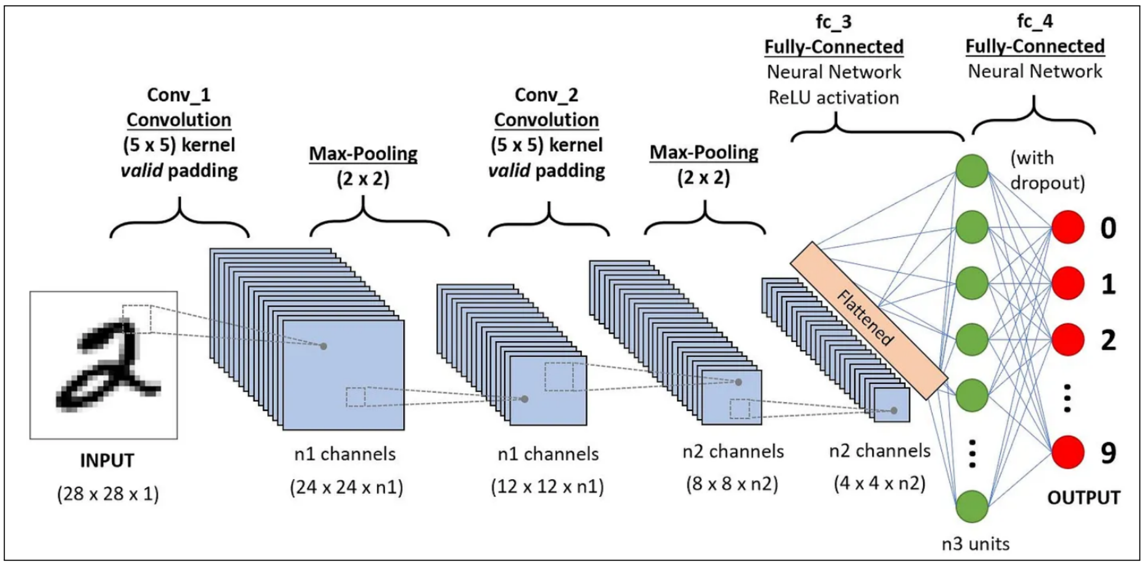

Figure 2.5: The architecture of a CNN [28].

2.2.1 Convolutional Layer

The convolutional layer has a central role in the network as it carries the main portion of the computational load. Its most common use is for detecting features by using a filter to scan the input image and outputting a feature map that classifies such discovered features. The feature extraction is done using two operations, i.e., convolution and activation [29].

In this layer, an element-wise multiplication between two matrices is performed: One of the matrices consists of a restricted portion of the image as an input. The other is the kernel, i.e., a filter of weights used to extract the features of the input. The kernel is passed through the entire image, each time performing the elementwise multiplication between each element of the kernel and the input matrix. Thereafter, the values are summed to acquire the output value of the corresponding position of the output, creating a feature map. The procedure can be repeated using several kernels representing different feature extractors, which in turn results in several feature maps. A feature map is a mapping of where certain features are found in the image. A CNN will look for features such as edges, objects, or straight lines [30].

The size of the kernel will prevent it from fully overlapping with the entire input matrix and zero padding is usually applied to address this issue. This consists in adding on all the sides of the input matrix, thus allowing the centre of the kernel to fit the outermost element of the input matrix as seen in Figure 2.6 [25]. Furthermore, the stride refers to the distance between two consecutive positions of kernels. The stride is commonly set to 1, however, a bigger stride is possible. Lastly, the output of the convolutional layer is passed through an activation function, most commonly the rectified linear unit [24].

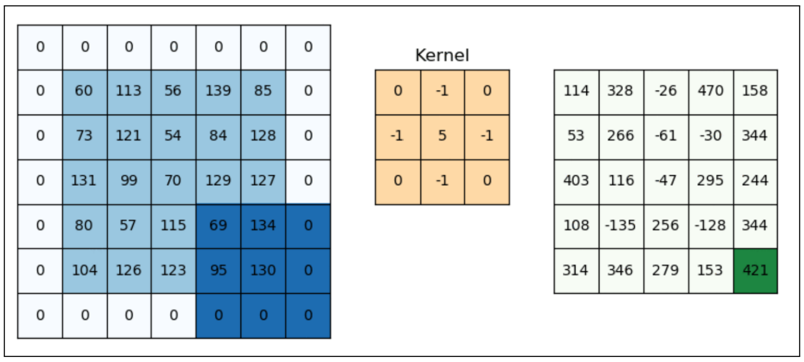

Figure 2.6: Zero padding. A matrix representation of the input data with zero padding = 1, which refers to the placement of outer zeros surrounding the matrix. The application of the kernel with a stride of one is demonstrated by the dark blue square, computing a singular numerical value within the right square shaded in green [31].

Hyperparameters are parameters that play a pivotal role in controlling the learning process of a machine learning algorithm, ultimately determining the model parameters that are acquired by the algorithm. The use of the prefix “hyper“ indicates that these parameters are considered to be at a higher level of abstraction, exerting direct control over the learning process and the resulting model parameters [32]. The most important hyperparameters to tune are the size of the kernel and which should be smaller than the input [25], the number of kernels[24], the stride length as well as the padding.

2.2.2 Pooling Layer

Pooling layers are responsible for reducing the spatial extent of the convolved features and have the purpose of decreasing the computational power that is required to process the data through dimensionality reduction [28]. The hyperparameters of the pooling layer are kernel size, stride length, and padding (as above).

Max pooling is one of the more popular operations for pooling. In this operation, each maximum value of the input feature map that the kernel covers are extracted and returned. Average pooling can otherwise be applied, which returns the average value which the kernel covers as visualised in Figure 2.7.

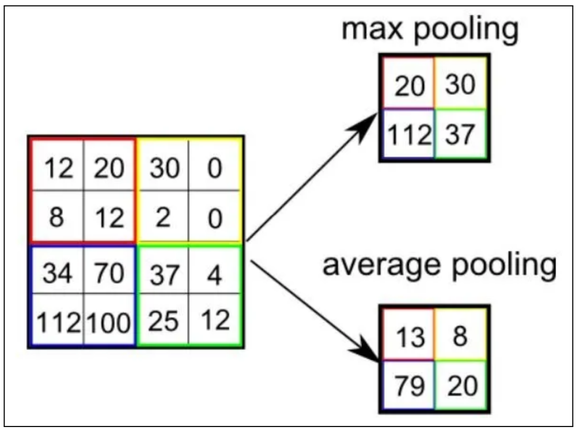

Figure 2.7: Different types of pooling [28].

2.2.3 Fully Connected Layer

The resulting features after the convolutional and pooling layers are then typically flattened into a vector and connected to one or several fully connected (FC), which maps them to the final outputs of the neural network, including class probabilities for classification tasks. The final fully connected layer usually has output nodes equal to the number of classes being classified. Nodes in these layers are fully connected to neurons in both the preceding and succeeding layers, and each fully connected layer is followed by a nonlinear function [24].

The hyperparameters to tune in this layer are the activation function, number of neurons, and dropout [29].

2.2.4 Faster R-CNN

In the domain of image classification, the Region-based Convolutional Neural Network (R-CNN) architecture was developed as a solution for identifying and classifying multiple objects in an image, where the standard CNN falls short. The R-CNN architecture is designed to detect the presence and location of different classes of objects in an image by utilising the concept of bounding boxes [33].

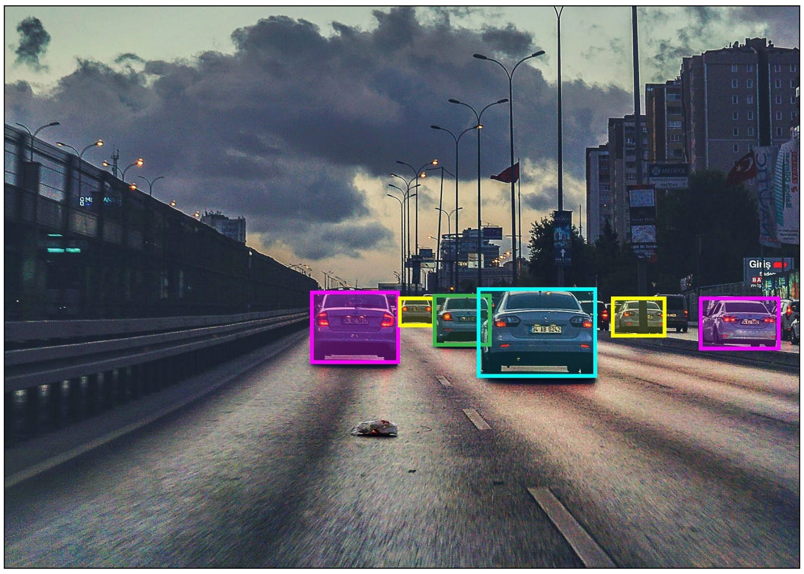

Figure 2.8: Cars with bounding boxes [34].

The basic workflow of R-CNN involves two major steps. Initially, the candidate regions of an image where objects may be located are determined through a selective search process (described below). Following the identification of regions, they are each passed through a CNN model, and predictions are made for both the object classes and their respective bounding boxes [35].

In 2014, Ross Girshick et al. [35] proposed the R-CNN to address the issue of efficient object localisation in object detection. Traditional methods employ the Exhaustive Search technique, which employs sliding windows of various scales on the image to suggest region proposals. In contrast, Girshick proposed using a Selective Search algorithm, which exploits object segmentation and Exhaustive Search to efficiently identify region proposals. The selective search algorithm proposes around 2000 region proposals for each image, which are then fed into the CNN model. Nonetheless, the selective search algorithm can be a time-consuming process. In addition, the R-CNN suffers from various drawbacks, including the requirement for multiple stages of training requiring prolonged training time, complex procedures, and significant amounts of disk space [36].

Strategies have been devised to improve its speed, such as the use of Region Proposal Networks (RPNs) in Faster R-CNN [37]. This method eliminates the requirement for a selective search algorithm by enabling the network to learn the region proposals on its own. The architecture of Faster R-CNN contains 2 networks, the Region Proposal Network (RPN) and the Object Detection Network. As in the standard R-CNN architecture, an input image is fed into a convolutional network, called “backbone”, which generates a convolutional feature map. When utilising a faster R-CNN, it is possible to incorporate a pre-trained model as the underlying backbone. Consequently, the principle of transfer learning can be effectively employed within this model. The goal of transfer learning is to enhance the performance of target learners in specific domains by transferring knowledge from related but distinct source domains [38].

Thereafter, a distinct network, called “RPN”, is employed to forecast the region proposals. The projected region proposals are then reshaped utilising an RoI pooling layer, which is further used to classify the image within the suggested regions and estimate the offset values for the bounding boxes[37]. A comparison between the different R-CNN types is shown in Table 2.1, highlighting that Faster R-CNN outperforms its predecessor.

	R-CNN	Fast R-CNN	Faster R-CNN
The mAP on Pascal VOC 2007 test dataset (%)	58.5	66.9 (when trained with VOC 2007 only) 70.0 (when trained with VOC 2007 and 2012 both)	69.9(when trained with VOC 2007 only) 73.2 (when trained with VOC 2007 and 2012 both) 78.8(when trained with VOC 2007 and 2012 and COCO)
The mAP on Pascal VOC 2012 test dataset (%)	53.3	65.7 (when trained with VOC 2012 only) 68.4 (when trained with VOC 2007 and 2012 both)	67.0(when trained with VOC 2012 only) 70.4 (when trained with VOC 2007 and 2012 both) 75.9(when trained with VOC 2007 and 2012 and COCO)

Table 2.1: Comparison Between R-CNN [39], Fast R-CNN [35] and Faster R-CNN [37]. The mAP is a performance metric that evaluates the accuracy of object detection models across all object classes in a given dataset. It is calculated as the average of the individual average precision (AP) values for each object class, where the AP is a measure of the precision-recall tradeoff for that particular class. Essentially, the mAP provides an overall indication of how well an object detection model performs across all object classes in a dataset [40]. VOC 2007, VOC 2012, and COCO are three datasets that have been used in object detection challenges (COCO Detection Challenge, 2023) [41].

2.3 Existing Peak Picking Software

At present, several pre-processing tools and packages are available for handling and processing LC-MS data. New tools are being released and existing tools are being updated as the demand for improved tools and packages is increasing [13]. However, research results show that current algorithms generate an abundance of false positives [42]. Despite the attempts to increase the level of automation in pre-processing and reduce the incidence of false positives and false negatives, these algorithms still necessitate significant manual parameter tuning and are prone to generating inaccurate results [43]. The outputs of peak detection can vary significantly based on the degree of restrictiveness applied when adjusting the parameters within the workflow. When less restrictive settings are used, the total number of features produced tends to increase, but so does the number of false positive peaks. Consequently, a larger fraction of the total spectral signals is comprised of false peaks. Conversely, if highly restrictive settings for parameter tuning are chosen, a smaller feature list is generated, consisting mostly of high-quality features. However, this approach may result in the exclusion of many good features due to a higher rate of false negative categorization [6].

The peak picking tool XCMS was selected as the benchmarking tool for this project as it was identified as one of the most cited tools out of 21 pre-processing tools in metabolomics papers published between 1995-2013 [44] and since it is employed at the Chalmers Mass Spectrometry Infrastructure and is availability as open-source software in the R environment.

A comparison was conducted by Coble JB et al. [44] to evaluate the performance of MetAlign, MZmine 2, and XCMS software tools using nominal mass GC-MS and accurate mass LC-MS data. According to the outcomes, all three tools generated an excessive number of false positives ranging from 10.4%- 33.8%. Higher detection accuracies were achieved when two software were combined. The study by Li C. et al. [45] shows an evaluation of the detection and quantification of true features by the different software XCMS, MZmine 2, MS-DIAL, MarkerView, and compound discovery. The authors used a real-case metabolomics dataset and a benchmark dataset consisting of 1100 compounds with specified concentration ratios between two standard mixtures, including 130 discriminating. The analysis reveals that all software had similar performance when detecting true features, however, XCMS' identification slightly surpassed the other four software (Table 2.2). Furthermore, in terms of quantification of true features, XCMS had the second-highest quantification accuracy for the QE HF dataset (Table 2.3).

[H]

			Total features	Consensus features	True features	True feature ID rate (%)
TripleTOF 6600 dataset	Targeted		-	-	970	-
	Untargeted	MarkerView	20.000	9718	833	85.9
		MS-Dial	26.185	15,582	871	89.8
		MZmine 2	24,472	23,677	876	90.3
		XCMS	28,168	25,386	896	92.4
QE HF dataset	Targeted		-	-	836	-
	Untargeted	Compound Discoverer	10,525	10,525	748	89.5
		MS-Dial	21,545	17,726	799	95.6
		MZmine 2	20,021	18,871	769	92.0
		XCMS	35,215	30,680	820	98.1

Table 2.2: Feature identification from the benchmark dataset [45].

			Accurately quantified true features	Quantification accuracy rate (%)	True discriminating markers	False discriminating markers(%)
TripleTOF 6600 dataset	Targeted		970	100	68	0
	Untargeted	MarkerView	737	88.5	41	17
		MS-Dial	683	78.4	47	60
		MZmine 2	798	91.1	59	4
		XCMS	588	65.6	55	191
QE HF dataset	Targeted		836	100	50	0
	Untargeted	Compound Discoverer	482	64.4	41	111
		MS-Dial	654	81.9	42	42
		MZmine 2	761	99.0	48	3
		XCMS	731	89.2	45	51

Table 2.3: Feature quantification and discriminating marker selection from the benchmark dataset [45].

In a study by Gürdeniz G. et al. [5], the three software XCMS, MZmine and MarkerLynx were compared. The overlap between features extracted by each software ranged from 37%-46%. Similarly, Tautenhahn et al [7] found that 46%-52% of the extracted features were common between MZmine and XCMS (Centwave). In fact, the Centwave algorithm is available within both XCMS and MZmine, and employs the continuous wavelet transformation to fit spectral peaks into a Gaussian shape [6]. Gürdeniz G. et al. [5] used a dataset consisting of rat plasma whereas Tautenhahn et al [7] made use of leaf and seed extracts. The different in the results could possibly be to the different natures of plasma samples in comparison to plant extracts [5].

A comprehensive overview of three software tools i.e., MZmine, XCMS, and MS-DIAL, commonly used for handling and processing LC-MS data is presented in Table 2.4.

Name	Features	Main application field	License Type	Platform
MZmine	Noise filtering, peak detection, alignment, normalisation and visualisation. Distributed computing noise filter, centroiding, peak detection, alignment and visualisation	Metabolomics with LC-MS and GC-MS data	GNU General Public License	Implemented in Java
XCMS	Noise filtering, peak detection, alignment	Metabolomics with LC-MS and GC-MS data	GNU General Public License	Implemented in R statistical language
MS-Dial	Noise estimation, Spectral deconvolution, peak identification, allignment	Metabolomics with GC/MS, GC/MS/MS, LC/MS, and LC/MS/MS) data	GNU General Public License	mplemented in C# language

Table 2.4: Freely available software for metabolomic data processing. The table summarises the key features of each tool, including its capabilities for peak detection and alignment, as well as their primary application, licensing type, and programming language [13] [46] [47].

.

3

Previous related Work

3.1 Deep Neural Networks for Classification of LC-MS Spectral Peaks

Kantz et al. [6] developed a machine learning pipeline that utilises a CNN for the classification of spectral characteristics observed in ROIs detected using other mechanisms. The dataset obtained by applying the MZmine 2 workflow to 78 raw data files consisted of 2770 features, which the authors referred to as peak groups. A subsequent manual inspection and labelling of the peak groups was carried out by an expert, who classified them as either "good" or "bad". The CNN was trained using complete peak shapes extracted from the LCMS data and windows were defined based on the upper and lower limits in m/z and retention time to enable the extraction of peak shapes.

The CNN was built using Python and Keras [48]. Furthermore, the CNN underwent training using 1304 instances of manually classified true and false peaks. These instances consisted of an equal number of false and true peaks, with 652 instances each. The model was further calibrated using 740 additional instances and was subsequently evaluated on independent 726 instances.

The performance of the deep learning algorithm was subsequently evaluated on 3000 candidate peaks using both “more restrictive” and “less restrictive” settings. The dissimilarities between these settings primarily pertain to the Chromatogram deconvolution segment of MZmine2. For instance, the more restrictive configuration necessitated a minimum absolute height value of 4.00E+05, while the less restrictive configuration required a minimum absolute height value of 1.50E+05. An evaluation of true and false peaks identified using conventional peak selection (MZmine 2) vs the CNN (Figure 3.1) showed that the CNN drastically reduced the occurrence of false peaks, at the expense of also reducing true peaks, albeit to a much lower extent [6].

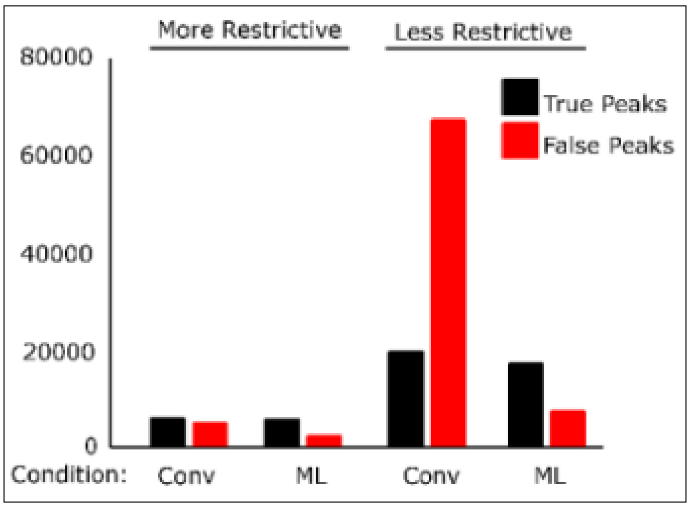

Figure 3.1: A comparison of true positive and false positive peaks retained comparing MZmine 2 vs the CNN. The Figure depicts, that regardless of settings, The term "Conv" denotes the process of peak extraction utilising conventional MZmine 2 workflows, while "ML" represents the count of peaks that were preserved following the implementation of the CNN [6].

Through the creation, refinement, and implementation of an image-based deep neural network model for peak classification, the researchers determined that this method has the potential to significantly enhance existing peak selection workflows, with a reduction in false peaks of around 90%.

3.2 Deep Learning for Precise Peak Detection in High-Resolution LC-MS Data

Melnikov et al. [9] employed a dual CNN approach to enhance the peak detection and peak integration step of the data analysis pipeline in metabolomics. The initial CNN classified time intervals from a specific chromatogram obtained using a modified centwave algorithm into three potential categories: (1) ROI does not include peaks, only noise, (2) ROI contains one or more peaks (3) ROI contains somewhat of a peak however special attention from a specialist is required. The subsequent CNN was responsible for peak integration. To build a dataset for their project, the authors manually annotated more than 4000 ROIs. In addition, the authors accentuated that distinguishing between indistinct peaks with low intensities (class 2) and noises or minute signals that cannot be accredited to a peak (class 3) was challenging. As a result, the boundary between these two classes is unclear, and it is possible that similar peaks in the resulting dataset could be classified differently. This underscores the challenge of accurately categorising peaks (Figure 3.2).

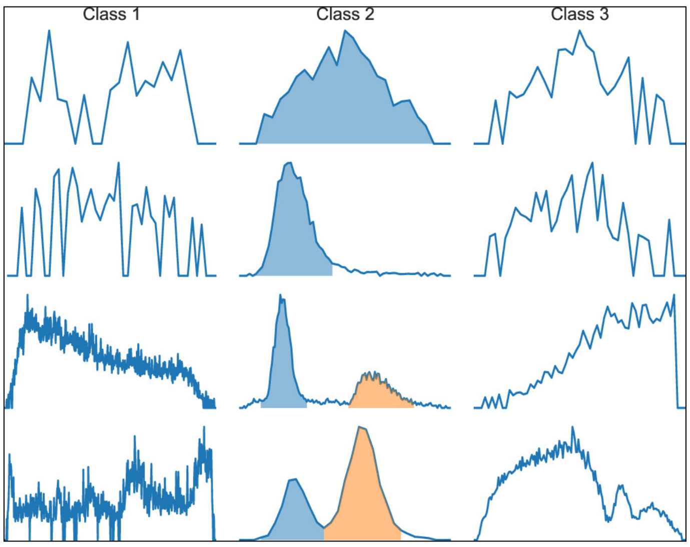

Figure 3.2: Examples of ROIs. The Figure illustrates examples of ROIs from each class. Class 1 corresponds to ROIs identified as noise, Class 2 corresponds to ROIs classified as one or more peaks, and Class 3 corresponds to uncertain peaks. The regions for peak integration are visually distinguished through the use of blue and orange fillings. The ROIs were taken from the training set [9].

To standardise the size of each ROI, linear interpolation was applied, resulting in a uniform size of 256 points. Furthermore, the signal intensities in the ROIs were normalised to a maximum value of unity. This approach ensured that the neural network made predictions solely based on the shape of the peak, as it was not influenced by variations in signal intensity. The output of the CNN is a set of probabilities (ranging from 0 to 1) that represent the assigned class for each ROI. These probabilities are calculated for each of the three classes and the sum of the probabilities for all three classes always adds up to 1.

The performance of the final model was evaluated on a hold-out test set, showing an accuracy of 87%. It was further shown that the model rarely misclassified peaks as noise, since only 0.5% of manually labelled peaks were misclassified. However, the majority of model errors involved incorrect ROI assignments to class 3 [9].

3.3 Deep Learning-Assisted Peak Curation for Large-Scale LC-MS Metabolomics

Gloaguen et al. [10] designed an independent deep learning-based peak filter tool, NeatMS, to integrate it into existing analysis pipelines (Figure 2.10(a)). NeatMS utilises a CNN to classify peaks according to their quality. Specifically, the CNN was trained to differentiate between peaks characterized as high quality, acceptable quality, or poor quality (i.e., noise), similar to the two above algorithms. These authors also utilised third-party tools (centWave and MZmine) to perform peak detection. Furthermore, similarly to the two above algorithms, they also emphasised that the full signal was retrieved from the raw data and used for the classification to prevent any possible bias applied by the different peak detection tools.

Before any data transformation, NeatMS excludes any unacceptable peaks, i.e., peaks that did not have at least 5 scans (configurable minimum scan number input filter). Furthermore, during the pre-processing step, the raw signal was subjected to a min-max normalisation and linear interpolation.

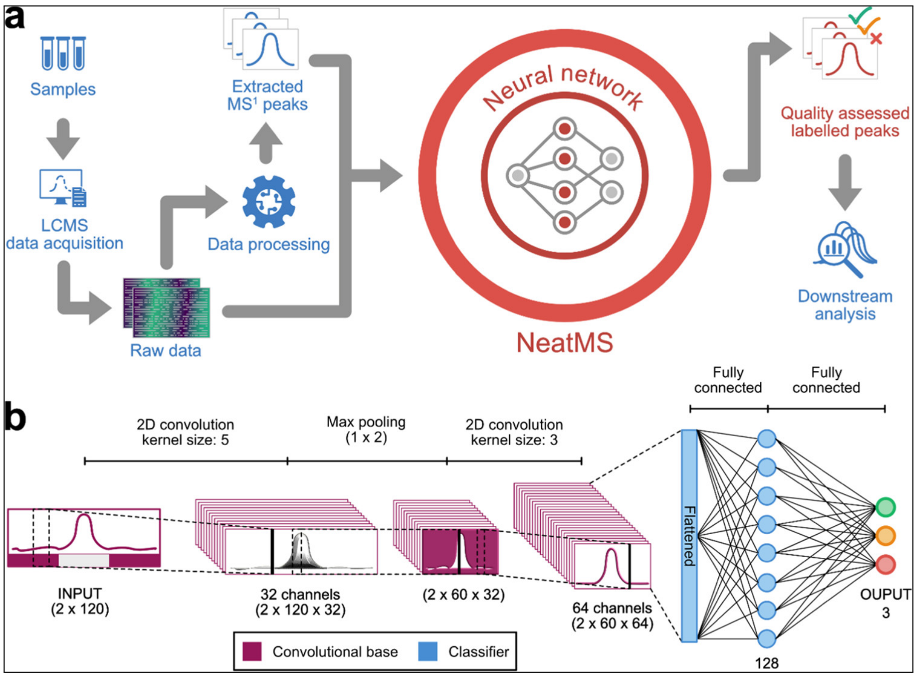

Figure 3.3: (a) shows the Integration of NeatMS (red) into an existing workflow (blue). (b) shows the architecture of the CNN. Figure (a) showcases the integration of NeatMS (in red) with an already established workflow (in blue). (b) depicts the architecture of the CNN which comprises a two-dimensional (2D) convolutional base for feature extraction and a classifier consisting of two fully connected layers [10].

NeatMS was written using Python 3.6 and the CNN was constructed using Keras and TensorFlow [49]. The architecture of the CNN is displayed in Figure 2.10 (b). During the training of the CNN Gloaguen et al., [10] utilised transfer learning.

The ROC-AUC is a performance measure that quantifies the ability of the model to distinguish between positive and negative samples. The ROC curves were generated for three distinct group separations in NeatMS, and the results demonstrate that the model closely aligns with the expert knowledge of the trainer. These findings suggest that NeatMS can serve as a faster, more consistent, and reproducible alternative to human expert evaluation, particularly for large-scale studies. The AUC scores for the ROC curves were 0.991 for high quality vs noise, 0.971 for high quality vs acceptable quality vs noise, and 0.940 for high quality vs acceptable quality.

3.4 Convolutional Neural Network for Automated Peak Detection in Reversed-Phase Liquid Chromatography

Contrary to the previously mentioned algorithms, Kensert et al. [50] implemented a CNN to perform automatic peak detection in reversed-phase liquid chromatography (RPLC). The model receives a complete chromatogram as input and subsequently generates outputs that include predicted locations, probabilities, and areas of the peaks. The CNN was exclusively trained on simulated chromatograms, constituting a training dataset comprising 1,000,000 chromatograms.

Regarding label encoding, the authors partitioned the chromatogram into 256 segments and assigned each segment a label of either peak or no-peak (i.e., 1 or 0), based on whether a peak apex was situated within the segment. Furthermore, if a peak was detected within a given segment, two additional labels were assigned: the relative position of the peak within that segment (a value between 0 and 1) and the overall peak area as seen in Figure 3.4.

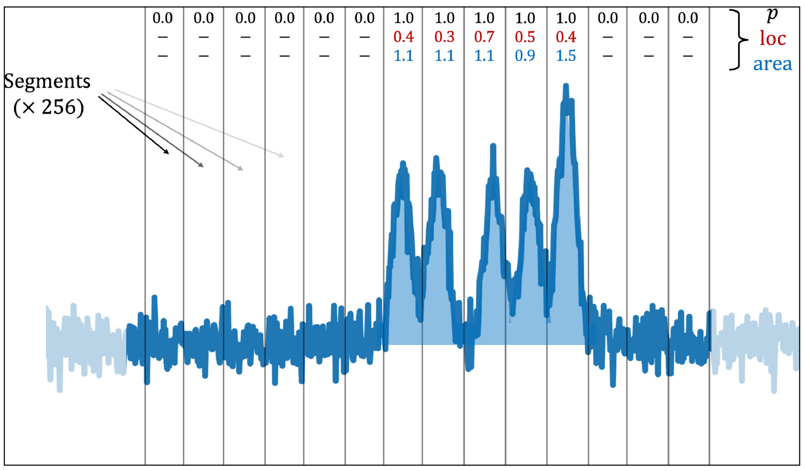

Figure 3.4: Labelling of a chromatogram. The Figure depicts a clear illustration of the labelling procedure employed for the chromatograms is presented. Initially, the chromatogram is divided into 256 segments. Subsequently, each segment is assigned the peak probability (p), the relative location of the peak (i.e., relative to the segment, loc), and the area of the peak (area). It is worth noting that the location and area of the peak are only assigned if a peak exists in the given segment, i.e., if its apex is contained within the segment. It is important to emphasize that the chromatogram used in this illustration is not representative of a realistic case [50].

For the CNN, the authors used TensorFlow (version 2.4) to implement a one-dimensional CNN model based on YOLO (Figure 2.12), a computationally fast approach to detecting objects in natural images, by giving a whole image as the input and predicting bounding boxes and associated class probabilities [51].

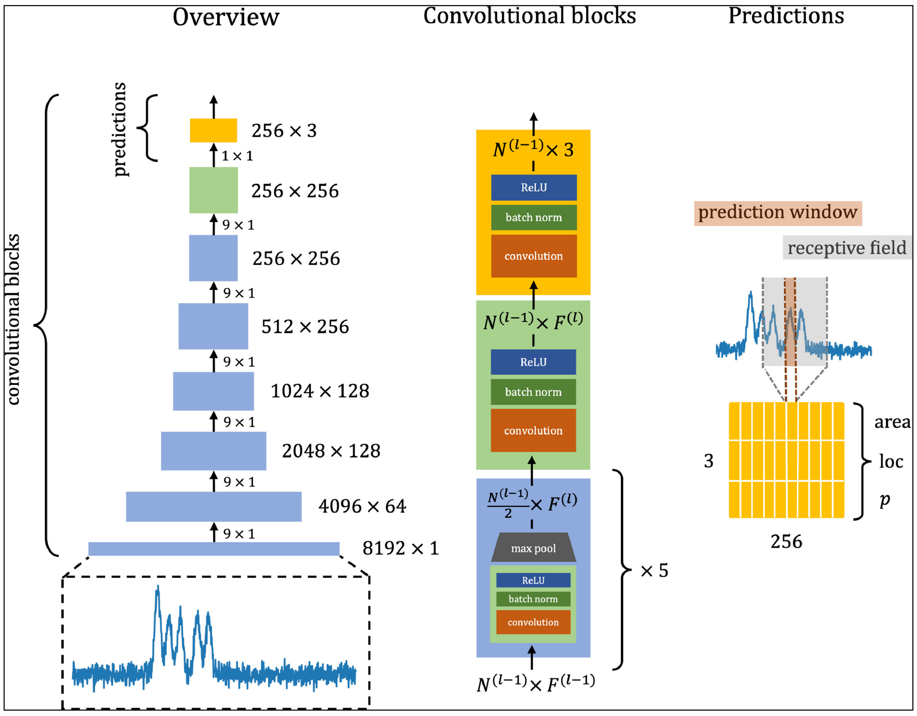

Figure 3.5: The architecture of the CNN. The CNN takes a raw chromatogram of 8192 (x 1) data points as an input and gives out a 256 × 3 array with predictions as the output. The CNN takes an 8192 (x 1) raw chromatogram as input and produces a 256 × 3 array with predictions as output. The output includes peak probability (p), relative peak location (loc), and peak area (area) for each segment. The dimension of the data is denoted next to the blocks, while the dimensions indicated next to the arrows between the convolutional blocks represent the filter size [50].

To evaluate the performance of the CNN, a validation set containing 10,000 simulated chromatograms was used. This resulted in a ROC-AUC at 0.996, indicating that the model has a high ability to correctly classify true peaks from false peaks. Furthermore, the mean relative error (MRE) between true peak areas and predicted peak areas was calculated at 0.1445; and the mean average error (MAE) between true peak locations and predicted peak locations was calculated at 0.062. Additionally, the MRE measures the average relative deviation between the predicted and observed values of a variable. In this case, the MRE of 0.1445 between the true and predicted peak areas indicates that on average, the predicted peak areas are 14.45% different from the true peak areas. Moreover, the MAE measures the average absolute difference between the predicted and actual values of a variable. In this case, the MAE of 0.062 between the true and predicted peak locations indicates that on average, the predicted peak locations differ from the true peak locations by 0.062 units (in the same units as the variable being measured).

Lastly, the authors mention that a possible drawback of using simulated chromatograms is the risk of not capturing the actual "chromatogram space" and it may therefore compromise the precision of peak detection in authentic chromatograms. Nonetheless, the clear advantage of employing simulated data is that the absolute (ground) truth is readily available, and there is no need to devote effort towards gathering and labelling chromatograms [50].

4 Methods

This chapter describes the process of acquiring the LC-MS dataset, while also highlighting the ethical considerations that require careful consideration.

4.1 Dataset

The Chalmers Mass Spectrometry Infrastructure (CMSI) has over the years generated roughly 1e6 injections (3D topological maps), of which approximately 10% are quality control (QC) samples. All QC samples are obtained from pooling aliquots of blood plasma from several individuals, thereby breaking the traceability to discrete individuals. These samples are used at the instrument platform to monitor instrument performance and to correct data for intensity drift. Furthermore, the samples are also used to detect metabolites that show up consistently across individual samples. At CMSI, there is thus available data from approximately 10.000 QC injections, each containing approximately 5.000 peaks, thereby constituting a training material of 10^7-10^8 data points that can be used for bulk training of automatically generated labels (ground truth).

All QC injections used in this project (for both training and testing purposes) were analysed at CMSI and were generated from three different QC samples that were available at CMSI for long-term reference purposes at the time (October 2017). The three different QC samples were prepared as technical replicates and analysed repeatedly, one after another, for a total of 527 injections over 6 batches by liquid chromatography-mass spectrometry (LC-MS).

Before injection on the LC-MS system, samples were prepared by precipitation of proteins with acetonitrile followed by centrifugation and filtering, in order to move the higher mass components (proteins) in the samples.

The LC-MS system (Agilent Technologies) consisted of a 1290 Ultra-High-Performance Liquid Chromatography (UHPLC) coupled to a 6550 quadrupole time-of-flight (qTOF) mass spectrometer. Centroided MS data were acquired with Mass Hunter vB.08.00. Study samples were injected on a C18 column (Waters Acquity UPLC HSS T3, 100 x 2.1 mm, 1.8 µm). Raw data files were converted from Agilent (.d) data format to .mzML format using the Proteowizard MSconvert software [52].

The raw data files were converted into mzML file format and subsequent extraction of chromatographic features was performed using XCMS and later assessed for peak quality using the CPC algorithm. The QC control data were converted into CSV format to extract the required attributes (rt, rtmin, rtmax, mz, mzmin, mzmax, TruePeak) for evaluation. Among all available data, they were organised based on their chromatography (H/R for HILIC and Reversed phase, respectively) as well as their polarity or ionization (P/N for Positive and Negative, respectively) for six batches each. Given the large availability of raw data the focus was narrowed down to one analytical batch of mzML raw data files (96 injections, collectively containing around 1 million peaks) from one chromatography (Reversed Phase), and one mode (Positive ionization). The data set for training and testing consisted of data from the first batch of RP, while data on peaks from the sixth batch were used for validation.

4.2 Ethical Considerations

Ethical considerations are of paramount importance when it comes to omics data, particularly when the data contains phenotypic information that can be traced back to individuals. One of the most pressing concerns is the issue of privacy and the potential for the data to be misused. This is particularly concerning in light of the General Data Protection Regulation (GDPR) laws, which came into effect 25th of May 2018 enforced by the European Parliament [53].

The quality control data utilised in this project, which was supplied by the Chalmers Mass Spectrometry Infrastructure, is not subject to GDPR. While quality control data comprises of phenotypic data, it cannot be linked to particular individuals.

5
Implementation

This chapter offers a thorough account of the project's implementation, encompassing the diverse stages involved in pre-processing the LC-MS data and establishing the requisite framework for evaluating the models.

5.1 Pre-Processing Data for ROI Detection Using Faster R-CNN

The primary aim of this step is to pre-process raw input data and convert it into a format suitable for training a Faster RC-NN model for ROI detection. This involves processing two distinct data sources: i.e., raw LC-MS data in the open mzML format, combined with information on peak location and quality assessment generated with the XCMS and CPC R packages on the same raw data.

The Pyteomics package was employed to manage mzML files to read files and convert spectra into a dictionary, allowing for fast accessibility to specific information for further pre-processing. The peaks were collected and made available as a CSV file, which was created outside the scope of this thesis. The table contained the peaks' location in the rt-mz-region for each mzML file. The last column denoted whether the peak passed certain criteria, labelling it as a peak or not by the CPC algorithm, whereby 95% of the XCMS peaks were identified as of sufficient quality.

The distributions of the most critical attributes of detected peaks in the QC data, namely retention time (rt) and mass-to-charge ratio (mz), were visualized and analysed (Figure 5.1). This showed that CPC and XCMS computed slightly different minimum and maximum values for rt, which describe where a peak starts and ends, respectively. CPC in general showed tighter RT intervals and was therefore chosen to represent ground truth for peak bounds. As a consequence, the rt characteristics from the CPC distribution will be denoted as rt in further references.

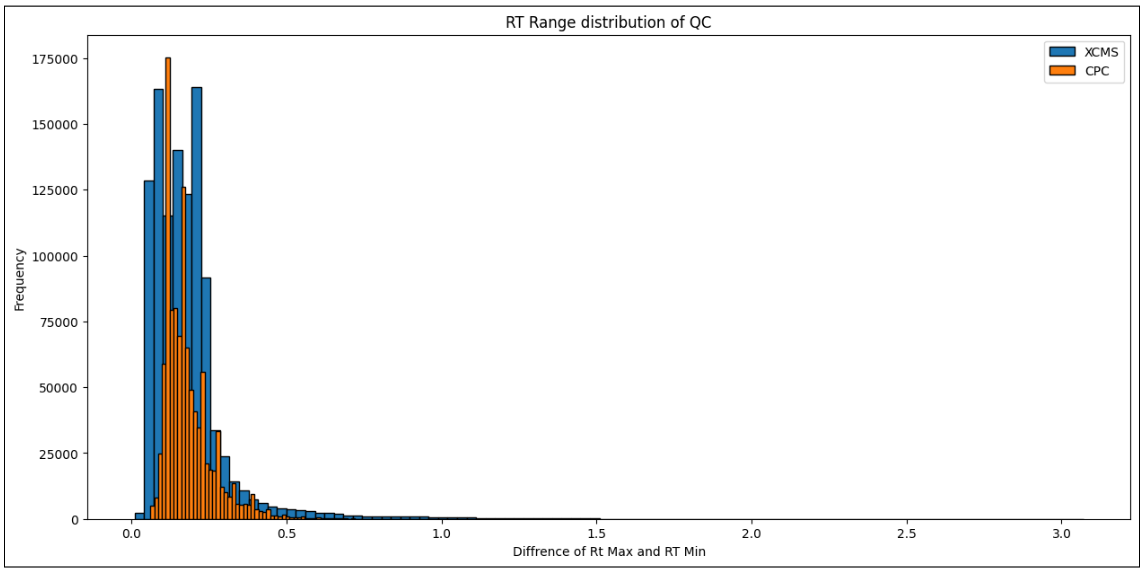

Figure 5.1: Distribution of peak widths (RT_{max}-RT_{min}) for the detected peaks in the training material. The vast majority of the peaks were less than 30 seconds wide.

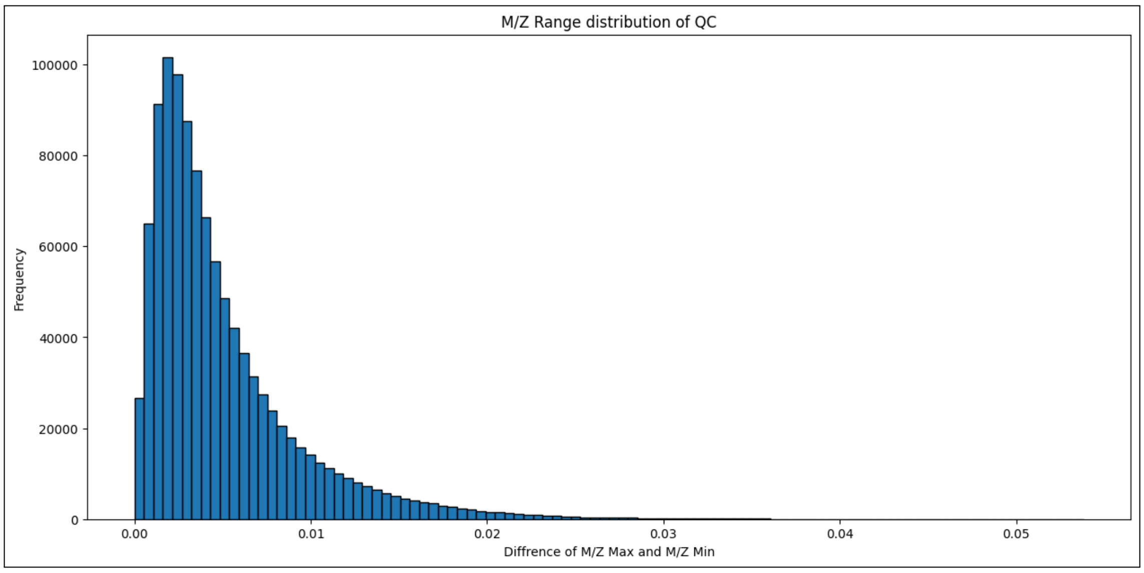

Figure 5.2: Distribution of peak widths (MZ_{max}-MZ_{min}) for the detected peaks in the training material. The majority of mz minimum and maximum values were found to be nearly identical, differing only by 0.01 or less. This finding points out the importance of handling these ranges with much care.

The sub-sectioning process involved scanning through the raw data along the retention time axis and, for each retention time window, along the mz axis. The aim was to extract smaller parts of an mzML file where one specific window would contain one or more peaks. To achieve this goal, the minimum and maximum retention time of each file, as well as each window's minimum and maximum mz values, were extracted, and windows were created between them. The parameters for window size, as well as step size in both directions, required careful tuning based on a trial-and-error approach and orientating by the distributions analysed above. Taking time for parameterisation is necessary to ensure that each peak was captured entirely on both axes without being split. Moreover, after visualising the peak shapes in the early (0.044 – 0.146 min) and late (10.895 - 10.997 min) retention times of a sample (total original range 0.044 - 10.997 min), these were excluded since they did not produce satisfactory peak shapes as input.

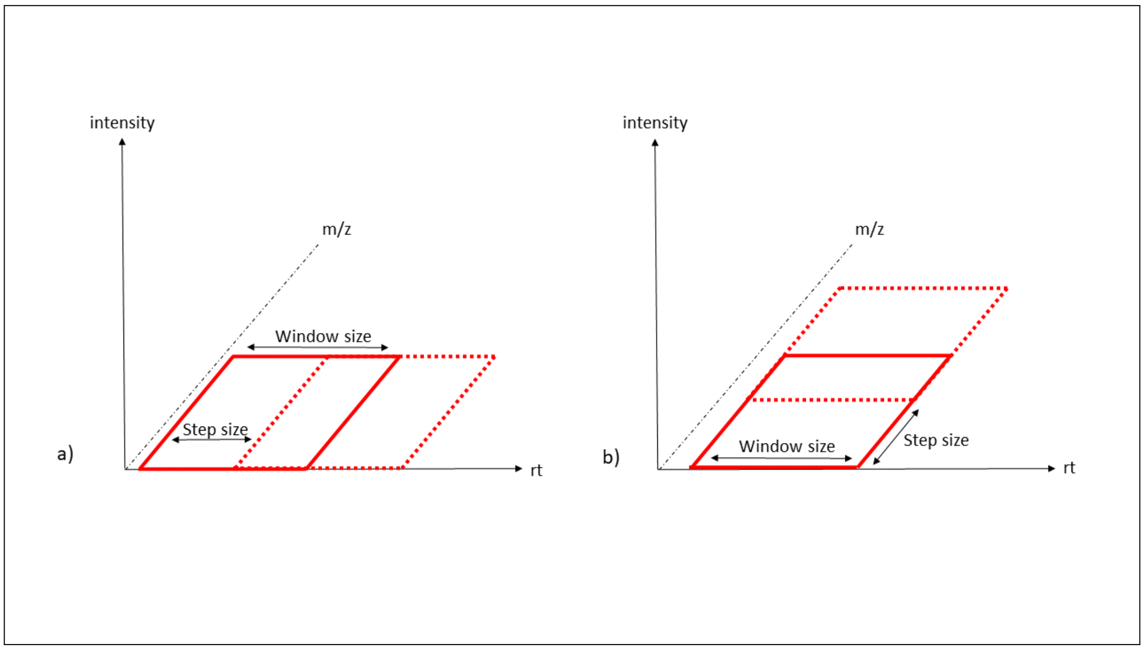

Figure 5.3: Extraction of subsections from LC-MS raw data. The Figure depicts the process of creating windows with uniform size along the rt axis (a), as demonstrated by the full rectangle indicating the initial window. Subsequently, the step size is parameterized, and the newly created window is shown with dashed lines. Notably, the size of each window along the rt axis remains constant. After creating the first retention time window (b), the mz windows are established with fixed window and step sizes, while the rt values remain unchanged until all possible mz windows have been generated.

The choice of an optimal window size was based on the consideration that the last 5% of the peak width distribution (Figure 5.1) largely consists of outliers, thus ending up with a window size of 0.64845 min. This ensured that the rt window size was sufficiently small as to limit the exponential increase in peaks included with increasing window size. Given that the total range of mz values within an rt window was much larger compared to the mz width for actual peaks, the creation of mz windows had to be halted by setting a parameter, "max iteration", to ensure the generation of enough windows throughout all retention times.

After extracting peak shape information from the quality control data and matching it with the corresponding raw data file, the spectral data were transformed into heatmaps by undergoing a binning process.

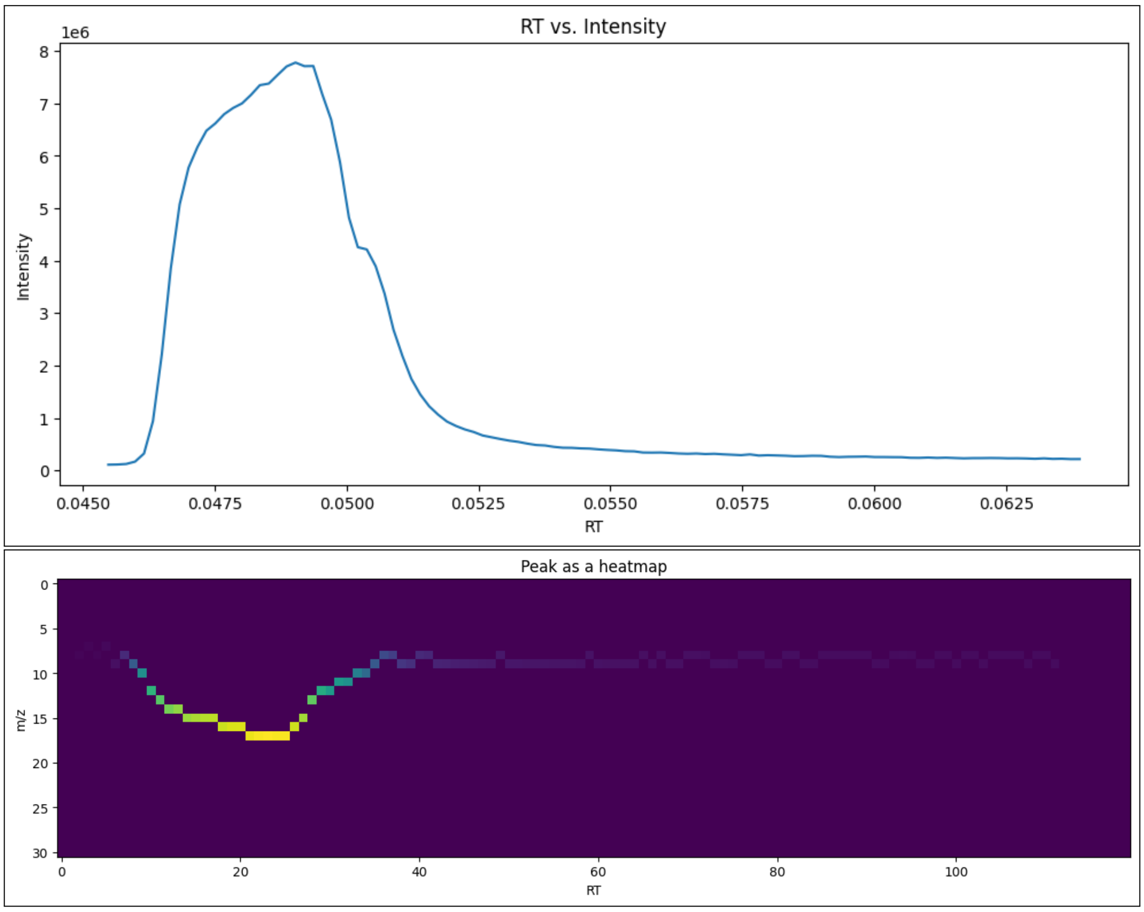

Figure 5.4: Peak visualized in 2D as a graph (upper) and in 3D as a heatmap (lower). The Figure displays the peak's two-dimensional projection on the upper, with the mz values omitted, plotting only the retention time and the corresponding intensity values. On the lower, the heat map illustrates the intended visual appearance of the images, where intensity is on the z-axis. Notably, the figure showcases the results of ROI detection, with a single peak selected to show a more detailed visualization.

Following that, the quality control peaks associated with each window were isolated. This isolation aimed to enable the conversion of the peak shape data from the extracted peak lists into images that can be interpreted by both machines and humans to train the faster R-CNN. The process involves converting the rt and mz values of peaks into the corresponding bin index within the specified rt/mz window. This scaling procedure facilitates the organisation and categorisation of the peaks based on their position within the designated window. Furthermore, each window may contain thousands of peaks, thus necessitating the use of binning. This entails binning both the raw data and the peak rt and mz values, and normalising the values to pixel values. Static binning is employed in this approach, whereby the number of bins in the rt and mz axes remains constant across all windows. Using the numpy digitize function, the length of each ROI is linearly interpolated by their bin indices. Intensities are summed when multiple intensities fall within the same bin.

Bins in the rt dimensions were estimated from the distribution of the number of scans per peak. Peaks obtained from the XCMS/CPC procedure were typically covered by no more than 25 scans. Thus, a multiple of this value should suffice to cover most peaks from start to finish. The width of the mz bins must correspond to the analysis of the deltas between the minimum and maximum mz values of the QC peaks. A larger mz window would result in all peaks falling within a single bin, as they would reside within the extended range of mz values, therefore rendering them indistinguishable from each other. After experimenting with different values, the final images were a size of 128 by 128 bins.

Once the peaks and corresponding raw data values were obtained, they were inserted into a heatmap by converting their raw data values into bin indices. The resulting heatmap was saved as an image, with the heatmap intensity (z-axis) represented as 1D grayscale intensity. Each saved image was assigned a unique traceable id, by combining rt and mz indices.

In summary, the ROI detection process involved iterating over the raw data with a set window size in both the rt and mz directions, creating heatmaps of occurring peaks and saving those images for the subsequent training process. Overall, this pre-processing methodology enabled the conversion of raw input data into a suitable format for ROI detection using Faster R-CNN in an efficient manner.

5.2 Faster R-CNN Package

In this project, a faster R-CNN model based on Ren et al.[37] was implemented with the fasterR-CNN_resnet50_fpn package from Pytorch [54]. This package requires the input to be a list of tensors. Each tensor should have a shape of [C, H, W], and correspond to a single image, with tensor values standardized within the range of 0 to 1. C refers to the number of channels or feature maps in the tensor, H denotes the height of the tensor, and W signifies the width of the tensor. During training, the expected input is the list of tensors as well as a target, which includes the coordinates for the ground truth bounding boxes along with the class label for each ground truth box. The model then outputs a dictionary of tensors that include the classification and regression losses for both the RPN and R-CNN. When used for prediction, the expected input is the list of tensors alone. Thereafter, a list containing dictionaries containing predicted bounding boxes, predicted labels, and scores of each detection is produced.

The package uses the ResNet-50-FPN [55] [56] backbone. However, a newer version called fasterR-CNN_resnet50_fpn_v2 is implemented in PyTorch. This model works similarly to fasterR-CNN_resnet50_fpn, although its backbone is instead based on Xie et al. [57]. In this project, we used the fasterR-CNN_resnet50_fpn implementation since it has been more extensively employed, and there is a considerable amount of online documentation available.

5.3 Training

During the pre-processing stage, a data frame was generated where each row contains a unique image ID and peak ID, which are necessary for tracing back to the original QC peak and its corresponding image. Additionally, the peak's pixel coordinate retention time and mass-to-charge ratio values were included, representing the bounding boxes. All boxes were structured as (x0, y0, x1, y1), where x denotes the retention time and y represents the mass-to-charge ratio. Another requirement is that the boxes must have a length and width greater than zero. This means that the minimum and maximum values of an axis cannot have the same bin index. If this occurs, those peaks will be excluded from the training, testing and validation processes. This exclusion has the advantage of removing very small peaks that would otherwise be displayed as a single pixel and do not provide meaningful insight. However, it also implies that potentially perfect peaks where no variation in mz values is observed (i.e., ideal scenario) are removed.

In the final step before training, the data frame was processed within a class designed to meet the specific requirements of the PyTorch framework's Faster R-CNN package. This class is defined as a subclass of torch.utils.data.Dataset, encompassing the necessary functionality to efficiently load and process the dataset. Within this class, there are two customized methods. The first method retrieves and loads the images from the dataset generated during pre-processing and transforms the images, bounding box coordinates, and labels into tensor format. The second method returns the total number of images, which is later used by the PyTorch DataLoader to split the data into training and test sets.

To train the Faster R-CNN model, 524 subsections extracted from chromatograms were generated as grayscale images and used as input. The labelled dataset was divided into an 80/20 training/test split. For the final validation step, a new set of images was created from the validation batch, resulting in 151 images. The deep learning model consisted solely of images that contained at least one peak and utilized the ResNet50 backbone. Alternative backbone models, such as MobileNetV3-Large, were subjected to training. However, their performance did not match that of ResNet50 and thus were not included.

To optimize the model, a learning rate scheduler was used, which controls and adjusts the learning rate during training, with the aim to improve the model's convergence [58]. Within this project, the model's hyperparameter was set to the default values [37]. Thus, the actual modelling can be performed "parameter-free," as ultimately desired in the research proposal. The only parameters that required tuning occurred while creating the ROI images during pre-processing in the previous step. Nevertheless, fine-tuning the Faster R-CNN model may improve the overall performance.

The method of non-maximum suppression (NMS) was initially applied to the resulting bounding boxes to focus the results on the smallest set of relevant ones. NMS makes use of Intersection over Union (IoU) as a measure of similarity (Figure 5.5), with values closer to 1 representing more similar boxes. After sorting, the box with the highest confidence was included in the output, effectively eliminating overlapping boxes [59]. Since many peaks are close to each other in an image, it could not be guaranteed that two bounding boxes belonged to the same object or correctly referred to different peaks. Rather than employing NMS, which introduced ambiguity in distinguishing the relationship between bounding boxes, an alternative approach was implemented instead. In this method, any predicted bounding boxes with a confidence level below 0.3 were excluded from the results. This ensured that less reliable predictions were eliminated from the output.

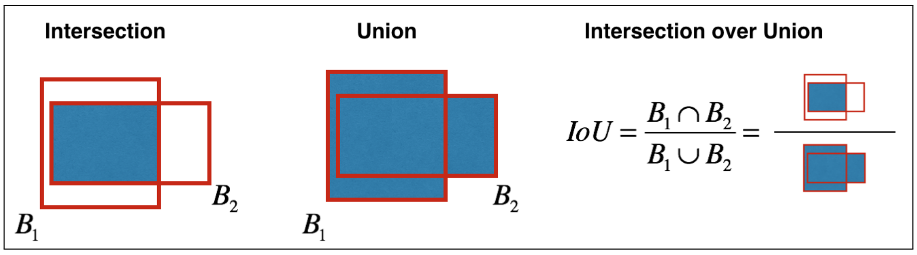

Figure 5.5: The Intersection over Union is calculated by dividing the shared area between a detection and ground-truth box by the combined area of both the detection and ground-truth boxes [60].

The Faster R-CNN of this project was trained on a GPU (Nvidia Tesla K80) using Google Colab [61]. Preserving the already trained weights and biases offer to reload them at a different time, which saves computational time and benefits to the aspect of reproducibility. The code was written in Python (V.3.10.11). The pre-processing of the raw data was done outside the scope of the thesis, though it was achieved using R [62] (V.4.3.0), XCMS [12] and CPC [2]. Furthermore, the data pre-processing for the model was performed using the libraries NumPy [63] (V.1.22.4), pandas [64] (V. 1.5.3), matplotlib [65] (V.3.7.1), Pyteomics [66] [67] (V.4.6.1a1).

5.4 Evaluation

In the field of Machine Learning, evaluating the performance of a model is a fundamental task. One important aspect during training is tracking the loss for each iteration, which provides insights into how the model's predictions deviate from the ground truth labels. To assess the model's performance, an instance of the Averager Class was used to compute the average loss for each epoch (Figure 5.6). An epoch refers to a complete pass through the entire training dataset during the training process of a machine learning model, signifying that the model has seen and processed all examples once [68].

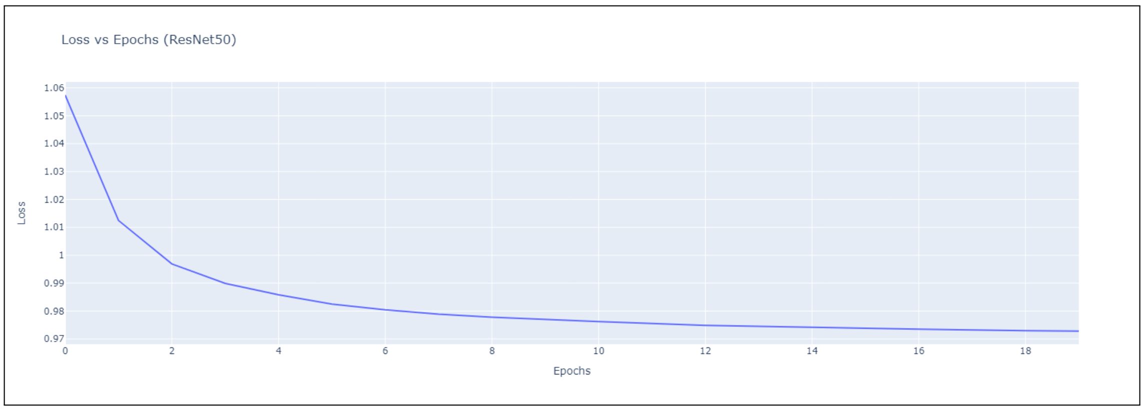

Figure 5.6: Loss as a function of training epochs for the ResNet50 model employed in this project. It refers to the discrepancy between the predicted and the actual output, serving as a measure of how well the model is performing during the training of epochs. Over the course of 20 epochs, the losses gradually decrease as the model learns to make better predictions, reducing the gap between predicted outputs and ground truth labels. Fast improvement is observed in earlier epochs, converging to around 0.972 after the 16th epoch, suggesting the model approaching optimal performance.

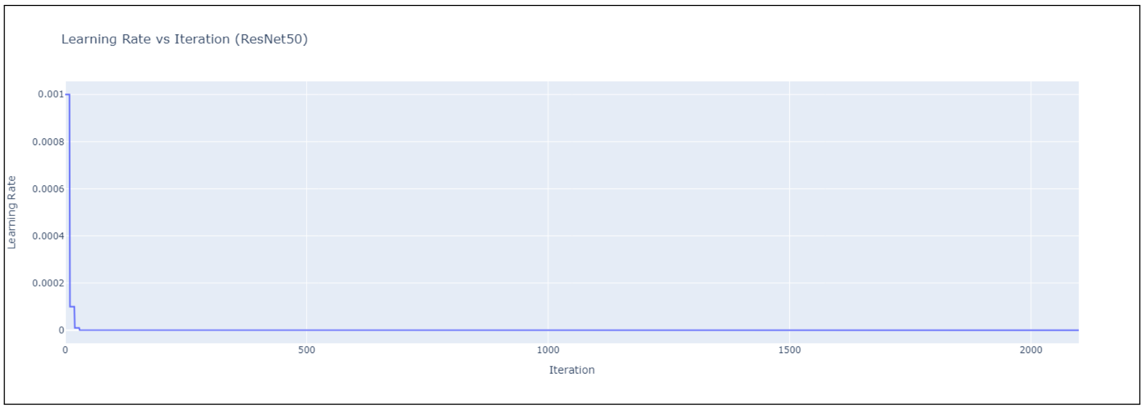

Figure 5.7: Adjusting the learning rate as a function of 2100 iterations during the training process for the ResNet50 model utilized in this project. Initially, for the first 10 iterations, the learning rate remains steady at 0.001. Subsequently, the values gradually decrease, indicating a finer adjustment towards convergence. The model proceeds with smaller steps, signifying its approach towards a relatively flat region in the loss landscape. The learning rate continues to decrease until it ultimately reaches a value around the two hundred decimal.

The training required 9:41.77 minutes using a GPU V100, which highlights the importance of utilizing powerful computational resources for training complex models like Faster R-CNN (data not shown).

Modelling performance can then be assessed using a confusion matrix, which contains counts of predicted and actual values, including true negatives (TN), true positives (TP), false positives (FP), and false negatives (FN) [69]. For assessing bounding boxes, labels, and scores, we used a threshold-based matching approach, which computes the Intersection over Union matrix between the ground truth and predicted coordinates. For each ground truth box, the predicted box with the highest IoU is selected if it exceeds a specified threshold (0.5 for this model). An alternative approach to consider is the Hungarian algorithm, which finds the assignment that minimizes the total cost by considering all possible pairings. It can be advantageous in scenarios where multiple ground truth boxes and predicted boxes need to be correctly matched, especially when the IoU values are close or overlapping [70].

In this scenario, where the aim is not to classify non-object regions correctly but rather to identify and locate objects within the image, true negatives are not relevant. In addition, varying box counts were managed by padding the true positives and false positives with zeros to match the maximum number of ground truth and predicted boxes. This ensures that the precision and recall calculations consider all boxes appropriately.

The average precision was 0.653 and the average recall was 0.728 for the test set, suggesting a relatively low false positive rate, minimizing the chances of erroneously identifying regions as ROIs. Furthermore, it demonstrated the ability to capture a significant portion of the true peaks. For the validation data, precision and recall decreased to 0.591 and 0.648, respectively. The F1 score, which provides a balanced measure of the model's accuracy, taking into account both the precision and recall [71], was 0.688 during testing and 0.617 for validation. Taken together, these metrics suggest that the model generalizes well and that the ROI detection thus shows promising results in accurately identifying and localizing the ROIs in LCMS chromatography.

Examining the distribution of Intersection over Union and computing the average IoU score, valuable insights can be gained about the overall accuracy of the ROI detection model during testing and validation, e.g., if the model achieves a high-level localization precision [70].

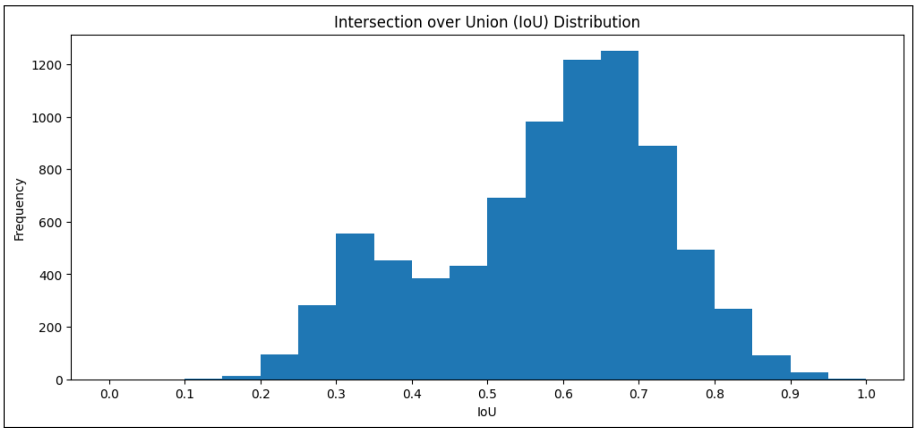

Figure 5.8: The distribution of Intersection over Union values in the test set showcases the number of bounding box pairs that represent the level of overlap between predicted and ground truth boxes. The model's predictions exhibit good alignment with ground truth ROIs, with a majority falling within the range of 0.2 to 0.8. The distribution reaches its peak around 0.6, signifying good performance in capturing ROIs. However, the model encounters difficulties in accurately predicting bounding boxes with high overlap, as the number of such pairs decreases significantly beyond 0.8. The average IoU was 0.579, while the validation score was 0.558, suggesting comparable performance on unseen data compared to test data.

The classification accuracy was 100%, which is an artifact of the fact that the model never encountered training images without peaks. However, since the primary focus of this baseline model was on accurately predicting bounding boxes, it has no influence on the project's aims.

The model discovered several peaks that did not correspond to the labelled peaks regions. There are various potential explanations for the occurrence of false peak selection by the model. It is plausible that the model characterizes regions with noise, low intensity, inadequate reproducibility, or imprecisely defined boundaries as potential peaks [10]. Furthermore, occurrences of falsely identified peaks frequently involve the presence of shoulder or double peaks, the emergence of spurious high background peaks, and peaks exhibiting low signal-to-noise ratios [72]. The purpose of an ROI detection algorithm is to find potential regions. This process can (and should) be complemented by a peak qualification algorithm (such as XCMS/CPC) to determine whether ROIs actually represent peaks. In order not to miss potential true peaks, it may be necessary to allow for oversampling of false peaks. However, it is important to note that while the occurrence of falsely identified peaks is a problem, it is not necessarily a major issue for the overall analysis. Thus, a randomized subset of predicted bounding boxes that could not be matched with ground truth coordinates was inspected manually (Figure 5.9).

	Precision	Recall	F1 score	mean IoU
Test set	0.653	0.728	0.688	0.579
Validation set	0.591	0.648	0.617	0.558

Table 5.1: Summary of the scores of each evaluation metric performed on the test set and validation set.

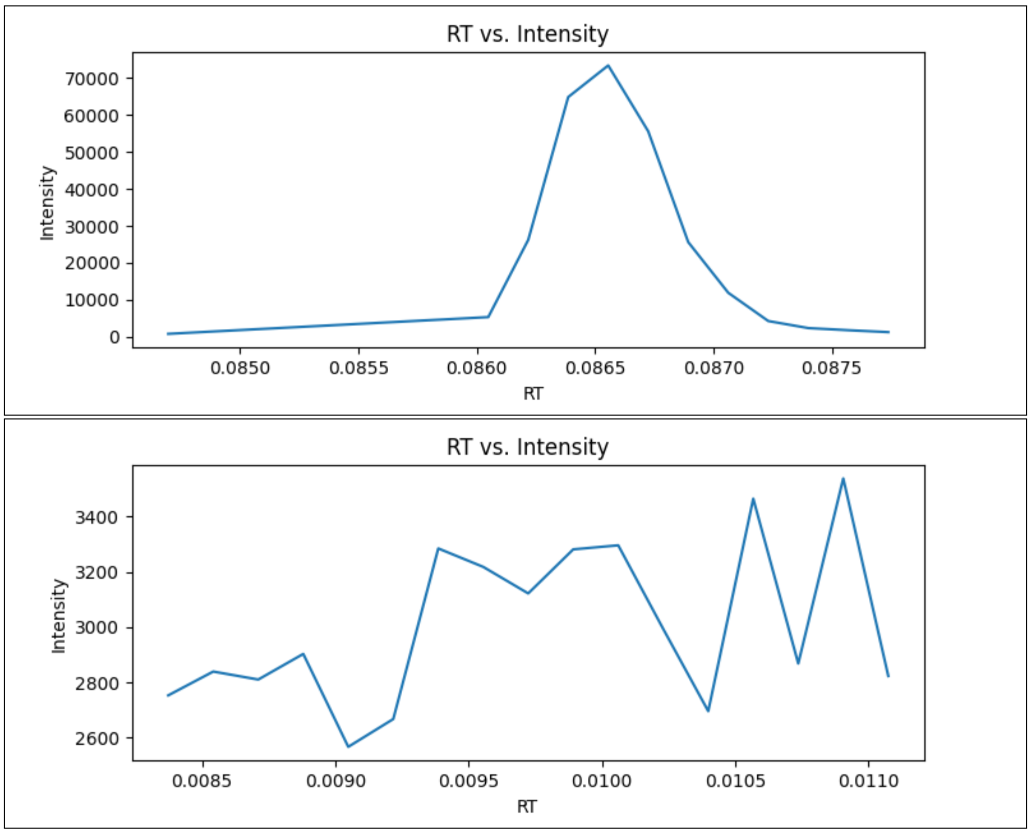

Figure 5.9: Two examples are displayed where an ROI was detected by the model but was not matched with a ground truth bounding box. The first represents a true peak undetected by the XCMS/CPC procedure used for ground truth data, while the second represents noise. By excluding the mz-axis and plotting exclusively the retention times and intensity values, a clearer visualization of the raw data's shape is achieved.

In total, there were 374 unmatched bounding boxes out of the total 151 validation images. A possible scenario could be that the model misaligned true peaks during threshold-based matching due to multiple bounding boxes. This hypothesis could be further examined by inspecting the five closest bounding boxes matching with raw data. Detected regions looking like noise suggest a misclassified peak due to its similar shape. Manual inspection showed that the data within the bounding boxes often had arbitrary shapes, likely representing noise. Furthermore, there were also empty images, indicating a lack of sufficient training. This could easily be remedied by hardcoding a condition against empty images.

6
Results

In the following chapter, the results of the project are discussed, including an examination of the significance of the results, an exploration of the limitations that might have influenced the outcomes, suggestions for potential improvements, and an exploration of future research possibilities within this particular field.

6.1 Importance of Results

Metabolomics has become a valuable tool in the life sciences, allowing for the identification and quantification of small molecules within biological systems. The detection of chromatographic peaks from raw spectral data is an essential element of any untargeted metabolomics pipeline. However, due to the availability of various techniques and analytical tools with adjustable input parameters, a single raw dataset may yield vastly different peak lists, significantly impacting the accuracy and reliability of outputs. Obtaining valid and reproducible results is, therefore, critical. The elimination of false positive chromatographic peaks is a crucial aspect that demands attention, as they impose a significant statistical burden and increase the likelihood of incorrect identifications when comparing metabolite variations among different biological groups [6]. This project emphasises the importance of obtaining accurate results in metabolomics and the essential role of eliminating false positives in extensive datasets.

Moreover, a limited number of machine learning models claim to be more efficient and consistent than human expert evaluation, including the NeatMS model mentioned in the study by Kantz et al. [6]. Accordingly, the objective of this project is to develop a machine learning model capable of providing an expert decision base with accuracy, efficiency, and consistency for ROI detection, to be combined with other approaches for final peak qualification.

6.2 Limitations of Results

The findings of this thesis were subject to certain limitations. Handling and analysing the large quantity of raw data and quality control peaks required a level of computational power that exceeded the available resources in this project. Consequently, the model could only be trained on a selected portion of the data. Additionally, the limited resources posed a challenge to train the model with different batch sizes and an increasing number of epochs. This resulted in fewer opportunities during the tuning phase, and consequently, the model's training results and loss were impacted. These constraints highlight the need for increased access and use of high-performance computing resources, such as GPU clusters. This would allow for greater experimentation and training of the model, leading to potentially more accurate and robust outputs. It should be noted, however, that this issue is prevalent within the entire fields of machine learning and artificial intelligence [73] and is not specific to this project.

Another factor that impacted the project are peaks with arbitrary shapes, unusual retention times, or mass-to-charge values. Such data can affect the accuracy of the model while training and, hence testing. Such peaks were commonly observed in the initial and concluding scans of a file. One possible way to address this issue is to adopt an outlier methodology that addresses these atypical peaks, such as removing or flagging them while training, to enable the model to handle them efficiently. It's important to consider that determining the exact minimum and maximum values can be difficult and may vary depending on subjective factors. Thus, it can affect the accuracy of bounding box determination in the training material.

Furthermore, this project required to build domain knowledge concerning the complex and specialised topic of metabolomics and LC-MS data. Similarly, all models require domain competence and work plans frequently need to take height to bridge the gap between domain knowledge and data analytical competence to effectively achieve an accurate problem description and solution. Therefore, it was necessary to dedicate sufficient time to comprehend the subject matter as well as identify how to effectively translate it into a deep learning model. The scope of the study was thus continually adapted to accommodate the new findings and information that emerged during the course of the project.

Importantly, the final model may miss certain peaks in the raw data: The task of identifying all peaks in the data requires a delicate balance between detecting low-intensity peaks and filtering out noise. Nevertheless, the application of deep learning brings the notable advantage of a high degree of flexibility. By integrating additional training data or modifying the model's architecture, it is possible to further enhance the model's performance [9].

Lastly, since this project was conducted as part of a master's thesis spanning a single semester, the modelling process had to be time-boxed to complete all tasks within the allotted timeframe. Hence, with more time allocated to this project, superior outcomes could have been achieved. Nonetheless, the current model serves as a good foundation for future development in the field and was considered acceptable within the constraints of the study.

6.3 Potential Improvement

An important consideration would be to incorporate augmented variations of the original chromatogram as part of the training process. This could involve applying diverse transformations such as introducing random Gaussian noise or performing random shifting, which are commonly employed in object detection tasks. One example would be to use the Albumentations tool [74], a widely used Python library within deep learning. Albumentations enables swift and adaptable image augmentations and focus on performance optimisation and efficiently incorporating a wide array of image transformations. These data augmentations serve not only to virtually expand the size of the training dataset but also as a regularisation technique, which enhances the model's ability to handle minor variations in the input data and thus improving its overall robustness [75].

Another development would be to conduct code refactoring to restructure and optimise existing code. This would enhance the code's readability and reusability, while concurrently reducing its complexity and long-term maintenance costs [76]. In relation to this aspect, our model was developed specifically for the analysis of LC-MS metabolomics data. However, an extension of the model would be to apply it to other data types, such as GC-MS data. Additionally, through collaborative engagement with metabolomics experts and acquiring deeper insights into the various factors contributing to specific instances of peak misclassification, modifications such as refining the input data or optimising model parameters could help enhancing the model performance.

Lastly, training a CNN with simulated chromatograms, e.g., using generative models to achieve realistic data, would help to reduce the laborious effort of collecting and annotating chromatograms. Employing such data offers advantages, as the absence of unwanted outliers and noise in the ground truth enhances hyperparameter tuning and subsequent precision. Such as approach would also facilitate project reproducibility. However, a potential shortcoming of this approach is the possibility that the CNN may not adequately capture the true chromatogram space required to accurately detect peaks in real-world chromatograms [50].

6.4 Future Work

Based on the results and limitations of this project, several recommendations can be proposed for prospective research pursuits. First, instead of utilising the pre-existing Faster R-CNN package provided by PyTorch, a viable improvement would involve the development of a custom LC-MS data R-CNN architecture from scratch, with a specific emphasis on capturing peak shapes. This approach would enable the development of a tailored model specifically designed for the precise objective of peak detection. Furthermore, this would allow for the inclusion of negative images in the training and test set, which pertain to images that do not include any peaks or bounding boxes. The inclusion of negative images would facilitate an evaluation of the model's capacity to effectively process a wider spectrum of inputs, thereby enhancing their practical applicability in classification. Moreover, it would be beneficial to evaluate the algorithm by benchmarking it against state-of-the-art peak-picking algorithms such as XCMS and MS-DIAL with optimised parameters.

Second, it is worth noting that the optimal training set size for the model depends on the specific LC methods employed. A higher degree of peak distinctiveness and ease of human classification results in a reduced requirement for training peaks to achieve optimal performance in the model. Conversely, LC-MS approaches that involve numerous isobaric peaks eluting within a narrow retention time range or entail a substantial number of peaks occurring close to the baseline noise level would necessitate a larger set of training peaks to attain optimal performance [6]. However, with a larger training dataset, the computational time required for processing increases, and parallel computing techniques, particularly during the pre-processing of input data and the training phase of the model, would be beneficial.

Third, another process to consider and explore in further research is the normalisation of values. It is important to determine which specific values should undergo normalisation and the rationale behind this decision. For instance, Kensert et al. [50] normalised intensity values to a range of 0 to 1, thereby making the intensity factor irrelevant to their CNN model. Normalising values can help deep learning models converge faster during training, by bringing input features to a similar scale, where the optimization process becomes more efficient and stable. Additionally, normalisation improves how well deep learning models can generalise. It ensures that no single feature dominates the learning process just because of its magnitude, allowing all features to contribute equally [77].

As highlighted above, extensive pre-processing was conducted on both the raw data and the QC dataset for ROI detection using Faster R-CNN. However, post-processing measures were not implemented, which are instrumental in enhancing the ability to differentiate true signals from false ones [6]. Incorporating post-processing techniques would be highly advantageous for the model's performance, along with expert-level tuning. For example, Melnikov et al. [9] performed post-processing by removing peaks with the rt length of more than three minutes. Moreover, considering the scope of the thesis, the model did not incorporate expert-level tuning. Nonetheless, the future aspiration entails integrating a tuning layer whereby an expert can examine the classification outcomes and subsequently provide their own insights to retrain the model. Adding such a tuning layer could be useful for transfer learning, to adapt a general peak-picking algorithm to instrument-specific conditions.

Last, it would be of consideration to incorporate a second neural network within the model, tasked with the classification of specific peaks. This additional neural network would handle the identification of rt, m/z, and intensities for individual peaks. Moreover, with additional time and resources, the optimal solution would be to train specific models for different ionization modes (e.g., positive, negative) and chromatography methods (e.g., RP, HILIC) and to incorporate them into a single package capable of handling different dataset types. Thereafter, creating an automated pipeline that integrates the entire process, including the identification of ROIs, detection of chromatographic peaks, quantification of peaks, grouping or matching of peaks for batch or analysis samples, and clustering of peaks belonging to the same compound [10]. This would eliminate the need for user intervention, requiring only input data from the user.

7 Conclusion

In this project, the potential for enhancing and automating the LC-MS pre-processing pipeline was explored: The specific objective of this thesis was to create a deep learning model to process raw data and generate identified regions of interest, representing potential chromatographic peaks, along with their corresponding coordinates. The developed model exhibited a certain degree of proficiency in generating ROIs, characterised by an average precision of 0.652/value and an average recall of 0.725/value during training/validation. This pilot application demonstrates high potential to identify ROIs with no reliance on predetermined parameters. Moreover, there is a strong conviction that allocating more time and resources to further refine these models could yield substantial improvements. However, several indicated avenues for model improvement could not be pursued due to time limitations. Furthermore, during validation, it was discovered that in addition to discovering several noise regions, the model also discovered several actual peaks that were not reported in the ground truth data.

This study constitutes an important advancement towards the development of a highly valuable tool that can automate LC-MS metabolomics data processing with precision, efficacy, and consistency without depending exclusively on valuable expert resources. Although the obtained results are of high value, there have also been several practical limitations, the need for computing power as well as the presence of arbitrarily shaped peaks. That should be taken into account in future work.

To conclude, the findings of this thesis hold potential implications for open science and academic contributions. The utilisation of deep learning models represents a noteworthy initial step towards automating the pipeline and addressing a crucial bottleneck, namely, the processing of raw data [9]. This advancement has yielded promising outcomes with substantial potential.

Bibliography

[1] G.J. Patti, O. Yanes, and G. Siuzdak. Metabolomics: the apogee of the omics trilogy. *Nature Reviews Molecular Cell Biology*, 13(4):263–269, March 2012. `doi:10.1038/nrm3314`.

[2] K. Pirttilä, D. Balgoma, J. Rainer, C. Pettersson, M. Hedeland, and C. Brunius. Comprehensive peak characterization (cpc) in untargeted lc-ms analysis. *Metabolites*, 12(2):137, 2022. `doi:10.3390/metabo12020137`.

[3] B. Zhou, J.F. Xiao, L. Tuli, and H.W. Ressom. Lc-ms-based metabolomics. *Mol Biosyst*, 8(2):470–481, 2012. `doi:10.1039/c1mb05350g`.

[4] J.C. Lindon, J.K. Nicholson, and E. Holmes. *The Handbook of Metabonomics and Metabolomics*. Elsevier, Amsterdam, The Netherlands, 2007.

[5] G. Gürdeniz, M. Kristensen, T. Skov, and L.O. Dragsted. The effect of lc-ms data preprocessing methods on the selection of plasma biomarkers in fed vs. fasted rats. *Metabolites*, 2(1):77–99, 2012. `doi:10.1021/ac051437y`.

[6] E.D. Kantz, S. Tiwari, J.D. Watrous, S. Cheng, and M. Jain. Deep neural networks for classification of lc-ms spectral peaks. *Analytical Chemistry*, 91(19):12407–12413, 2019. `doi:10.1021/acs.analchem.9b02983`.

[7] R. Tautenhahn, C. Böttcher, and S. Neumann. Highly sensitive feature detection for high resolution lc/ms. *BMC Bioinformatics*, 9(1), 2008. `doi:10.1186/1471-2105-9-504`.

[8] G. Libiseller, M. Dvorzak, U. Kleb, E. Gander, T. Eisenberg, F. Madeo, S. Neumann, G. Trausinger, F. Sinner, T. Pieber, and C. Magnes. Ipo: a tool for automated optimization of xcms parameters. *BMC Bioinformatics*, 16:118, 2015. `doi:10.1186/s12859-015-0562-8`.

[9] A.D. Melnikov, Y.P. Tsentalovich, and V.V. Yanshole. Deep learning for the precise peak detection in high-resolution lc-ms data. *Analytical Chemistry*, 92(7):588–592, 2020. `doi:10.1021/acs.analchem.9b04811`.

[10] Y. Gloaguen, J.A. Kirwan, and D. Beule. Deep learning-assisted peak curation for large-scale lc-ms metabolomics. *Analytical Chemistry*, 94(12):4930–4937, 2022. `doi:10.1021/acs.analchem.1c02220`.

[11] S. Castillo, P. Gopalacharyulu, L. Yetukuri, and M. Orešič. Algorithms and tools for the preprocessing of lc–ms metabolomics data. *Chemometrics and*

Intelligent Laboratory Systems, 108(1):23–32, 2011. doi:https://doi.org/10.1016/j.chemolab.2011.03.010.

[12] C.A. Smith, E.J. Want, G. O'Maille, R. Abagyan, and G. Siuzdak. Xcms: Processing mass spectrometry data for metabolite profiling using nonlinear peak alignment, matching, and identification. *Analytical Chemistry*, 78(3):779–787, 2006. doi:10.1021/ac051437y.

[13] M. Katajamaa and M. Orešič. Data processing for mass spectrometry-based metabolomics. *Journal of Chromatography A*, 1158(1):318–328, 2007. doi:https://doi.org/10.1016/j.chroma.2007.04.021.

[14] M. Hilario, A. Kalousis, C. Pellegrini, and M. Müller. Processing and classification of protein mass spectra. *Mass spectrometry reviews*, 25(3):409–449, 2006. doi:https://doi.org/10.1002/mas.20072.

[15] E. Rampler, Y.E. Abiead, H. Schoeny, M. Rusz, F. Hildebrand, V. Fitz, and G. Koellensperger. Recurrent topics in mass spectrometry-based metabolomics and lipidomics—standardization, coverage, and throughput. *Analytical Chemistry*, 93(1):519–545, 2021. doi:10.1021/acs.analchem.0c04698.

[16] C.A. Hastings, S.M. Norton, and S. Roy. New algorithms for processing and peak detection in liquid chromatography/mass spectrometry data. *Rapid Communications in Mass Spectrometry*, 16(5):462–467, 2002. doi:https://doi.org/10.1002/rcm.600.

[17] X. Li, E.C. Yi, C.J. Kemp, H. Zhang, and R. Aebersold. A software suite for the generation and comparison of peptide arrays from sets of data collected by liquid chromatography-mass spectrometry*s. *Molecular and Cellular Proteomics*, 4(9):1328–1340, 2005. doi:https://doi.org/10.1074/mcp.M500141-MCP200.

[18] W. Wang, H. Zhou, H. Lin, S. Roy, T.A. Shaler, L.R. Hill, S. Norton, P. Kumar, M. Anderle, and C.H. Becker. Quantification of proteins and metabolites by mass spectrometry without isotopic labeling or spiked standards. *Molecular and Cellular Proteomics*, 75(18):4818–4826, 2003. doi:10.1021/ac026468x.

[19] M. Bellew, M. Coram, M. Fitzgibbon, M. Igra, T. Randolph, P. Wang, D. May, J. Eng, R. Fang, C. Lin, J. Chen, D. Goodlett, J. Whiteaker, A. Paulovich, and M. McIntosh. A suite of algorithms for the comprehensive analysis of complex protein mixtures using high-resolution lc-ms. *Bioinformatics*, 22(15):1902–1909, 2006. doi:10.1093/bioinformatics/btl276.

[20] D. Radulovic, S. Jelveh, S. Ryu, T.G. Hamilton, E. Foss, Y. Mao, and A. Emili. Informatics platform for global proteomic profiling and biomarker discovery using liquid chromatography-tandem mass spectrometry. *Molecular and Cellular Proteomics*, 3(10):984–997, 2004. doi:10.1074/mcp.M400061-MCP200.

[21] K.C. Leptos, D.A. Sarracino, J.D. Jaffe, B. Krastins, and G.M. Church. Mapquant: open-source software for large-scale protein quantification. *Proteomics*, 6(6):1770–1782, 2006. doi:10.1002/pmic.200500201.

[22] S. Walczak and N. Cerpa. *Artificial Neural Networks*. Academic Press, New York, third edition edition, 2003.

[23] IBM. What is a neural network? URL: https://www.ibm.com/topics/neural-networks.

[24] R. Yamashita, M. Nishio, R.K.G. Do, and K. Togashi. Convolutional neural networks: An overview and application in radiology - insights into imaging. *SpringerOpen*, 9(4):611–629, 2018. doi:https://doi.org/10.1007/s13244-018-0639-9.

[25] M. Mishra. Convolutional neural networks, explained, 2020. URL: https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939.

[26] V. Gulshan, L. Peng, M. Coram, M. Stumpe, D. Wu, A. Narayanaswamy, S. Venugopalan, K. Widner, T. Madams, J. Cuadros, R. Kim, R. Raman, P. Nelson, J.L. Mega, and D.R. Webster. Development and validation of a deep learning algorithm for detection of diabetic retinopathy in retinal fundus photographs. *JAMA*, 316(22):2402–2410, 2016. doi:10.1001/jama.2016.17216.

[27] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, and S. Thrun. Dermatologist-level classification of skin cancer with deep neural networks. *Nature*, 542(7639):115–118, 2017. doi:10.1038/nature21056.

[28] S. Saha. A comprehensive guide to convolutional neural networks — the eli5 way, 2018. URL: https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5-way-3bd2b1164a53.

[29] M. Isaksson. Four common types of neural network layers, 2020. URL: https://towardsdatascience.com/four-common-types-of-neural-network-layers-c0d3bb2a966c.

[30] R. Khandelwal. Convolutional neural network: Feature map and filter visualization, 2020. URL: https://towardsdatascience.com/convolutional-neural-network-feature-map-and-filter-visualization-f75012a5a49c.

[31] Dharmaraj. Zero-padding in convolutional neural networks, 2021. URL: https://medium.com/@draj0718/zero-padding-in-convolutional-neural-networks-bf1410438e99.

[32] K. Nyuytiymbiy. Parameters and hyperparameters in machine learning and deep learning, 2020. URL: https://towardsdatascience.com/parameters-and-hyperparameters-aa609601a9ac.

[33] R. Gandhi. R-cnn, fast r-cnn, faster r-cnn, yolo — object detection algorithms, 2018. URL: https://towardsdatascience.com/r-cnn-fast-r-cnn-faster-r-cnn-yolo-object-detection-algorithms-36d53571365e.

[34] Infolks Group. How bounding box enables object detection?, 2019. URL: https://infolksgroup.medium.com/how-bounding-box-enables-object-detection-999b3059974e.

[35] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '14, pages 580–587, USA, 2014. IEEE Computer Society. `doi:10.1109/CVPR.2014.81`.

[36] B. Liu, W. Zhao, and Q. Sun. Study of object detection based on faster r-cnn. In *2017 Chinese Automation Congress (CAC)*, pages 6233–6236, 2017. `doi:10.1109/CAC.2017.8243900`.

[37] S. Ren, K. He, R. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press.

[38] F. Zhuang, Z. Qi, K. Duan, D. Xi, Y. Zhu, H. Zhu, H. Xiong, and Q. He. A comprehensive survey on transfer learning. *Computing Research Repository*, abs/1911.02685, 2019. URL: `http://arxiv.org/abs/1911.02685`, `arXiv:1911.02685`.

[39] R. Girshick. Fast r-cnn. In *2015 IEEE International Conference on Computer Vision (ICCV)*, pages 1440–1448, 2015. `doi:10.1109/ICCV.2015.169`.

[40] R. Padilla, S.L. Netto, and E.A.B. da Silva. A survey on performance metrics for object-detection algorithms. In *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)*, pages 237–242, 2020. `doi:10.1109/IWSSIP48289.2020.9145130`.

[41] M. Everingham, S.M. Ali Eslami, L. Van Gool, C.K.I. Williams, J. Winn, and A. Zisserman. The PASCAL visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, 2014. `doi:10.1007/s11263-014-0733-5`.

[42] O.D. Myers, S.J. Sumner, S. Li, S. Barnes, and X. Du. Detailed investigation and comparison of the XCMS and MZmine 2 chromatogram construction and chromatographic peak detection methods for preprocessing mass spectrometry metabolomics data. *Analytical Chemistry*, 89(17):8689–8695, August 2017.

[43] O. Owen, S. Sumner, S. Li, S. Barnes, and X. Du. One step forward for reducing false positive and false negative compound identifications from mass spectrometry metabolomics data: New algorithms for constructing extracted ion chromatograms and detecting chromatographic peaks. *Analytical Chemistry*, 89(17):8696–8703, 2017. `doi:10.1021/acs.analchem.7b00947`.

[44] J. Coble and C.G. Fraga. Comparative evaluation of preprocessing freeware on chromatography/mass spectrometry data for signature discovery. *Journal of Chromatography A*, 1358:155–164, 2014. `doi:https://doi.org/10.1016/j.chroma.2014.06.100`.

[45] Z. Li, Y. Lu, Y. Guo, H. Cao, Q. Wang, and W. Shui. Comprehensive evaluation of untargeted metabolomics data processing software in feature detection,

quantification and discriminating marker selection. *Analytica Chimica Acta*, 1029:50–57, 2018. `doi:https://doi.org/10.1016/j.aca.2018.05.001`.

[46] H. Tsugawa, T. Cajka, T. Kind, Y. Ma, B. Higgins, K. Ikeda, M. Kanazawa, J. VanderGheynst, O. Fiehn, and M. Arita. Ms-dial: Data independent ms/ms deconvolution for comprehensive metabolome analysis. *Nature Methods*, 12:523–526, 2015. `doi:10.1038/nmeth.3393`.

[47] O. Fiehn. Ms- dial (documentation), 2022. URL: `http://prime.psc.riken.jp/compms/msdial/main.html`.

[48] F. Chollet et al. Keras: The python deep learning library. Astrophysics Source Code Library, record ascl:1806.022, jun 2018. `arXiv:1806.022`.

[49] M. Abadi, P. Agarwal, A.and Barham, E. Brevdo, Z. Chen, C. Citro, G.S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mane, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, U. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viegas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and Z. Zheng. Tensorflow: Large-scale machine learning on heterogeneous distributed systems, 2016. `arXiv:1603.04467`.

[50] A. Kensert, E. Bosten, G. Collaerts, K. Efthymiadis, P. Van Broeck, G. Desmet, and D. Cabooter. Convolutional neural network for automated peak detection in reversed-phase liquid chromatography. *Journal of Chromatography A*, 1672:463005, 2022. `doi:10.1016/j.chroma.2022.463005`.

[51] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[52] M.C. Chambers, B. Maclean, R. Burke, D. Amodei, D.L. Ruderman, S. Neumann, L. Gatto, B. Fischer, B. Pratt, J. Egertson, K. Hoff, D. Kessner, N. Tasman, N. Shulman, B. Frewen, T.A. Baker, . Brusniak, C. Paulse, D. Creasy, L. Flashner, K. Kani, C. Moulding, S.L. Seymour, L.M. Nuwaysir, B. Lefebvre, F. Kuhlmann, J. Roark, P. Rainer, S. Detlev, T. Hemenway, A. Huhmer, J. Langridge, B. Connolly, T. Chadick, K. Holly, J. Eckels, E.W. Deutsch, R.L. Moritz, J.E. Katz, D.B. Agus, M. MacCoss, D.L. Tabb, and P. Mallick. A cross-platform toolkit for mass spectrometry and proteomics. *Nature Biotechnology*, 30(10):918–920, 2012. `doi:10.1038/nbt.2377`.

[53] European Union. General data protection regulation, 2022. URL: `https://gdpr-info.eu/`.

[54] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. *PyTorch: An Imperative Style, High-Performance Deep Learning Library.* Curran Associates Inc., Red Hook, NY, USA, 2019.

[55] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016. `doi:10.1109/CVPR.2016.90`.

[56] T.Y. Lin, P. Dollar, R.B. Girshick, K. He, B. Hariharan, and S.J. Belongie. Feature pyramid networks for object detection. *Computing Research Repository*, abs/1612.03144, 2016. URL: `http://arxiv.org/abs/1612.03144`, `arXiv:1612.03144`.

[57] Y. Li, S. Xie, X. Chen, P. Dollar, K. He, and R.B. Girshick. Benchmarking detection transfer learning with vision transformers. *Computing Research Repository*, abs/2111.11429, 2021. `doi:https://doi.org/10.48550/arxiv.2111.11429`.

[58] C. Kim, S. Kim, J. Kim, D. Lee, and S. Kim. Automated learning rate scheduler for large-batch training. *Computing Research Repository*, abs/2107.05855, 2021. URL: `https://arxiv.org/abs/2107.05855`.

[59] J.H Hosang, Benenson R., and Schiele. B. Learning non-maximum suppression. *Computing Research Repository*, abs/1705.02950, 2017. URL: `http://arxiv.org/abs/1705.02950`.

[60] I. Zafar, G. Tzanidou, R. Burton, N. Patel, and L. Araujo. *Hands-On Convolutional Neural Networks with TensorFlow: Solve Computer Vision Problems with Modeling in TensorFlow and Python.* Packt Publishing, 2018.

[61] V. Privalov. Hardware exploration on google colab and kaggle platforms, 2019. URL: `https://vovaprivalov.medium.com/hardware-exploration-on-google-colab-and-kaggle-platforms-576bf51c54e`.

[62] R Core Team. *R: A Language and Environment for Statistical Computing.* R Foundation for Statistical Computing, Vienna, Austria, 2021. URL: `https://www.R-project.org/`.

[63] S. van der Walt, S.C. Colbert, and G. Varoquaux. The numpy array: A structure for efficient numerical computation. *Computing in Science and Engineering*, 13(2):22–30, 2011. `doi:10.1109/MCSE.2011.37`.

[64] W. McKinney. Data structures for statistical computing in python. In S. van der Walt and J. Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 56–61, 2010. `doi:10.25080/Majora-92bf1922-00a`.

[65] J.D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science and Engineering*, 9(3):90–95, 2007. `doi:10.1109/MCSE.2007.55`.

[66] A.A. Goloborodko, L.I. Levitsky, M.V. Ivanov, and M.V. Gorshkov. Pyteomics—a python framework for exploratory data analysis and rapid software prototyping in proteomics. *Journal of the American Society for Mass Spectrometry*, 24(2):301–304, 2013. `doi:10.1007/s13361-012-0516-6`.

[67] L.I. Levitsky, J.A. Klein, M.V. Ivanov, and M.V. Gorshkov. Pyteomics 4.0: Five years of development of a python proteomics framework. *Journal of*

Proteome Research, 18(2):709–714, 2019. PMID: 30576148. doi:10.1021/acs.jproteome.8b00717.

[68] E.R. Davies and M. Turk. *Advanced Methods and Deep Learning in Computer Vision*. Computer Vision and Pattern Recognition. Elsevier Science, 2021. URL: https://books.google.se/books?id=ZqYsEAAAQBAJ.

[69] K.M. Ting. *Confusion Matrix*, pages 209, 781. Springer US, Boston, MA, 2010. doi:10.1007/978-0-387-30164-8_157.

[70] S.H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I.D. Reid, and S. Savarese. Generalized intersection over union: A metric and A loss for bounding box regression. *Computing Research Repository*, abs/1902.09630, 2019. URL: http://arxiv.org/abs/1902.09630.

[71] z.c Lipton, c. Elkan, and B. Narayanaswamy. Thresholding classifiers to maximize f1 score, 2014. arXiv:1402.1892.

[72] N. Borgsmüller, Y. Gloaguen, T. Opialla, E. Blanc, E. Sicard, A.L. Royer, B. Le Bizec, S. Durand, C. Migné, M. Pétéra, E. Pujos-Guillot, F. Giacomoni, Y. Guitton, D. Beule, and J. Kirwan. Wipp: Workflow for improved peak picking for gas chromatography-mass spectrometry (gc-ms) data. *Metabolites*, 9(9), 2019. URL: https://www.mdpi.com/2218-1989/9/9/171, doi:10.3390/metabo9090171.

[73] A.J. Lohn and M. Musser. Ai and compute: How much longer can computing power drive artificial intelligence progress, January 2022. doi:10.51593/2021CA009.

[74] A. Buslaev, A. Parinov, E. Khvedchenya, A. Iglovikov, and A. Kalinin. Albumentations: fast and flexible image augmentations. *ArXiv e-prints*, 2018. arXiv:1809.06839.

[75] L. Perez and J. Wang. The effectiveness of data augmentation in image classification using deep learning. *Computing Research Repository*, abs/1712.04621, 2017. URL: http://arxiv.org/abs/1712.04621.

[76] C. Abid and M.and do Nascimento Ferreira T.and Dig D. Alizadeh, V.and Kessentini. 30 years of software refactoring research: A systematic literature review. *Computing Research Repository*, abs/2007.02194, 2020. URL: https://arxiv.org/abs/2007.02194.

[77] J.and Zhou Y. Huang, L.and Qin, F. Zhu, L. Liu, and L. Shao. Normalization techniques in training dnns: Methodology, analysis and application. *Computing Research Repository*, abs/2009.12836, 2020. URL: https://arxiv.org/abs/2009.12836.