Object Detection for Signal Separation with Different Time-Frequency Representations

by

Llewellyn Strydom

Submitted in partial fulfillment of the requirements for the degree
Master of Engineering (Computer Engineering)

in the

Department of Electrical, Electronic and Computer Engineering
Faculty of Engineering, Built Environment and Information Technology

University of Pretoria

July 2021

SUMMARY

OBJECT DETECTION FOR SIGNAL SEPARATION WITH DIFFERENT TIME-FREQUENCY REPRESENTATIONS

by

Llewellyn Strydom

Supervisor(s):	Prof. W. P. du Plessis
Department:	Electrical, Electronic and Computer Engineering
University:	University of Pretoria
Degree:	Master of Engineering (Computer Engineering)
Keywords:	Signal separation, signal classification, machine learning, object detection, joint time-frequency analysis

Signal separation refers to the task of detecting and separating multiple radio-frequency (RF) signals in a wideband scenario. This task has attracted interest from both the cognitive radio (CR) and electronic warfare (EW) communities, and has several civilian and military uses. As with so many tasks before it, machine learning (ML)-based solutions are proving to be very adept at solving this task. Specifically, several approaches based on object detection algorithms from the computer vision (CV) literature has been proposed for the signal separation task. There are two categories of ML-based object detection algorithms, namely single-shot and region-based. In this work one single-shot (YOLOv2) and one region-based (Faster R-CNN) object detection algorithm are implemented, trained and tested, to solve the signal separation task. The features used in previous signal separation research are limited to the magnitude of the short-time Fourier transform (STFT) and the related spectrogram, and in many cases it is unclear how these features are represented in ML workflows. This work provides an in-depth analysis of different time-frequency representations based on the complex-valued STFT. Results for the signal separation task are presented in terms of traditional object detection metrics for different time-frequency representations and several experiments are conducted to quantify the performance of YOLOv2 and Faster R-CNN under certain conditions.

ACKNOWLEDGEMENTS

I would like to thank Saab Grintek Defence and the University of Pretoria for their financial support. I would also like to thank the Centre for High Performance Computing (CHPC) for providing me with the necessary compute resources to have completed this work.

To Prof. Warren du Plessis: Thank you for the tremendous amount of time and effort that you have invested in me. I have yet to meet someone who is more committed to their profession, and for this I applaud you.

I would like to extend my gratitude to Mrs Helena Gous for the courteous manner in which she dealt with all admin related queries.

I am grateful for my parents, Charl and Carien, who have always encouraged me to follow my passion, and supported me in all my endeavours. I owe everything that I am and everything that I will ever be to you. I would also like to thank my sister, Anita, for her friendship and support.

Carla – my partner for life – you are the reason that I wake up with a smile every morning. I thank you for your endless love and support.

I would also like to thank Andrew Purves for proofreading this work.

Finally, praise is due to the Lord for blessing me with the multitude of opportunities that have allowed me to complete this work.

Live as if you were to die tomorrow. Learn as if you were to live forever. - Mahatma Gandhi

LIST OF ABBREVIATIONS

AMC	automatic modulation classification
AP	average precision
AWGN	additive white Gaussian noise
BFSK	binary frequency-shift keying (FSK)
CDF	cumulative distribution function
CHPC	Centre for High Performance Computing
CNN	convolutional neural network
COCO	Common Objects in Context
CPU	central processing unit
CR	cognitive radio
CV	computer vision
CW	continuous wave
CWD	Choi-Williams distribution
DARPA	Defense Advanced Research Projects Agency
DWT	discrete wavelet transform
ELINT	electronic intelligence
EMS	electromagnetic spectrum
ES	electronic support
EW	electronic warfare
FFT	fast Fourier transform
FM	frequency modulation
FSK	frequency-shift keying
GMSK	Gaussian minimum shift keying

GNU	GNU's Not Unix!
GPU	graphics processing unit
GSM	Global System for Mobile Communications
I/Q	in-phase and quadrature
ICASA	Independent Communications Authority of South Africa
IEEE	Institute of Electrical and Electronics Engineers
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoT	Internet of Things
IoU	intersection over union
ITU	International Telecommunication Union
JTFA	joint time-frequency analysis
LFM	linear frequency modulation
LPI	low probability of interception
LTE	Long-Term Evolution
mAP	mean average precision
MJD	multiband joint detection
ML	machine learning
NMS	non-maximum suppression
PDF	probability density function
PSK	phase-shift keying
QAM	quadrature amplitude modulation
R-CNN	Regions with convolutional neural network (CNN) Features
RAT	Radio Access Technology
ReLU	rectified linear unit
RF	radio-frequency
ROC	receiver operating characteristic
RoI	region of interest
RPN	Region Proposal Network
RWR	radar warning receiver
SAR	synthetic aperture radar

SDR	software-defined radio
SGD	stochastic gradient descent
SNR	signal-to-noise ratio
SPP	spatial pyramid pooling
SSD	Single Shot MultiBox Detector
STFT	short-time Fourier transform
SVM	support vector machine
TPU	tensor processing unit
VGG	Visual Geometry Group
VOC	Visual Object Classes
WVD	Wigner-Ville distribution
XML	Extensible Markup Language
YOLO	You Only Look Once

TABLE OF CONTENTS

Chapter 1 INTRODUCTION

1.1 INTRODUCTORY REMARKS

The aim of this chapter is to provide the reader with an explanation as to why this research was conducted and what questions it aims to answer. A motivation for the methods that were considered in this work is given in Section 1.2. A list of the research objectives and questions is provided in Section 1.3, with the hypothesis and methodology contained in Section 1.4. In Section 1.5, the goals of this research are stated, and the expected research outputs are summarised in Section 1.6. A summary of the layout of this document is contained at the end of this chapter, in Section 1.7.

1.2 PROBLEM STATEMENT

The purpose of this research work is to investigate the use of object detection methods to detect and localise signals in the time-frequency domain by using different input features. The primary motivation for conducting this research is introduced in Section 1.2.1, and the aspects of the problem that are not covered in the current literature are discussed in Section 1.2.2.

1.2.1 Context of the Problem

Wideband spectrum sensing requires multiple signals of varying types to be detected and localised over a wide frequency range. In this work, this task is referred to as signal separation. Signal separation cannot be performed by traditional narrowband spectrum sensing algorithms, *e.g.*, matched filtering, energy detection, and cyclostationary detection, since these algorithms only detect the presence (or absence) of a single signal [1].

Signal separation also deals with multiple signals that may overlap in both time and frequency. In such cases, a detection is only useful if a signal can be localised by accurately estimating the centre frequency, bandwidth, and start and stop times of the relevant signal. This information makes it possible to separate – or potentially counter – multiple signals.

The advances in cognitive radio (CR) on the civilian side, and electronic support (ES) on the military side, necessitate systems that can perform the signal separation task effectively. These systems are also often required to extract more advanced signal features, *e.g.*, modulation type. At the same time that these requirements are becoming more demanding, the electromagnetic spectrum (EMS) is also becoming increasingly congested, which makes the task even more challenging [2]. Accurate signal separation systems that can operate in complex environments will become more and more prevalent, and clearly already have many uses.

National and international communications authorities, like the Independent Communications Authority of South Africa (ICASA) and the International Telecommunication Union (ITU), may benefit from systems that can detect, localise and identify various signals over a wide frequency range. Such systems can be used to perform automated monitoring of the EMS in areas that are under their administration. This will allow signals that are transmitted by rogue operators, and signals that exceed their licensed bandwidth to be detected quickly.

Most existing automatic modulation classification (AMC) methods assume that a single signal has already been isolated, but in a wideband scenario this assumption is invalid [3, 4]. A signal separation algorithm that can estimate the centre frequency, bandwidth, and start-stop times of multiple signals can enable these AMC methods to operate in a wideband scenario. It has also been shown that if the estimates of the aforementioned parameters are inaccurate, then the accuracy of some AMC methods will degrade significantly, which again necessitates the need for accurate systems [5].

Methods based on computer vision algorithms, and object detection in particular, have been proposed for the signal separation task [6–10]. These algorithms are able to detect and isolate signals from time-frequency images in much the same way that a human operator looking at a spectrogram would. These methods show a lot of promise, but very little research has been conducted to determine which object detection algorithms work best and which input features are most suitable for the problem.

1.2.2 Research Gap

The limited number of object detection-based algorithms for signal separation have all used a single input feature, without considering alternatives [7–11]. All of the currently available research used either the short-time Fourier transform (STFT) or the spectrogram, and either projected the values onto a colour map or used the raw values, without considering whether their choice was optimal or affected their results in any way. In order to maintain compatibility with open-source computer vision frameworks, some researchers have repeated a single feature over three input channels, resulting in a tensor with the same dimensions as a natural image.

Different object detection algorithms have been proposed to perform signal separation tasks, but there exists no in-depth comparison of the different algorithms. Specifically no in-depth investigation has been conducted to compare the performance of both single-shot and region-based object detection algorithms for the signal separation problem.

Previous work has also not included a thorough investigation into the performance of these algorithms at different signal-to-noise ratio (SNR) levels, non-maximum suppression (NMS) thresholds and intersection over union (IoU) thresholds. In this work, all of these aspects pertaining to the signal separation problem will be investigated and the results will be analysed. This will provide greater insight into how these algorithms perform and what their strengths and weaknesses are.

Results from previous work is often difficult to validate or replicate as the data that was used is not publicly avaiable. Details pertaining to the custom datasets that were used are also scant, which also makes it impossible to generate new data with the same parameters. This work will aim to clearly describe how data was generated, so as to allow others to reproduce the results contained herein.

1.3 RESEARCH OBJECTIVE AND QUESTIONS

The primary objective of this work is to investigate the ability of deep learning-based object detection algorithms to detect and localise heterogenous radar and communications signals, when operating on different time-frequency representations. Researchers have shown that the magnitude of the STFT and spectrogram are suitable for this task, but in this work, experiments will be conducted to investigate

specifically how different representations of the complex-valued STFT affect the accuracy of these algorithms.

This objective can be broken down into the following research questions.

- How do different time-frequency representations affect the performance of object detection algorithms on the signal separation task?
- Are single-stage or two-stage object detection algorithms more suitable for solving the signal separation task using time-frequency images?
- How well do object detection algorithms perform the signal separation task when signals are partially overlapping?
- How well do object detection algorithms perform at low SNR levels?

1.4 HYPOTHESIS AND APPROACH

It has been shown that object detection algorithms, if suitably used, are capable of performing the signal separation task effectively, and it is expected that this work will reiterate this finding. Other time-frequency representations, that have not been considered before, are expected to make it possible to achieve the same accuracy as before, while somewhat reducing the complexity of preprocessing.

In order to test this hypothesis, two different object detection algorithms will be trained and tested on the signal separation task. These methods will require a large amount of data, which will be simulated using appropriate software packages. This study will not consider any real, recorded data, as the task of collecting and labelling such data is very time consuming and prone to errors. In contrast, the approach described in this work can be accurately reproduced by others, which will allow for future comparisons.

The trained object detection algorithms will be used to conduct several experiments. These experiments will be designed to highlight the strengths and weaknesses of the different algorithms, under different conditions. Different configurations and thresholds that are used during training, inference and evaluation will be tested, which includes the use of transfer learning, NMS configuration and IoU

thresholds for determining positive detections. The results will be reported for different signal types at different SNR levels.

The proposed methods are expected to detect and estimate the parameters of high-SNR signals more accurately than those of low-SNR signals. Transfer learning has been used successfully in many domains, and as such, it is expected that the use thereof will also be beneficial for the methods proposed here. NMS has been shown to improve the accuracy of several object detection algorithms, and although it is expected that the same will hold true for the signal separation task, the use of NMS will naturally mean that some highly overlapping signals will go undetected.

The accuracy of object detection algorithms is commonly compared using the mean average precision (mAP) metric. This metric has also been used to quantify the performance of object detection algorithms on the signal separation problem [8–10]. Since this metric is commonly used, it will also be used in this work. The relationship between this metric and other metrics and measures that are often used by the signal processing community, *e.g.*, false-alarm rate and receiver operating characteristic (ROC) curves, is not known and could be the basis of a future study.

1.5 RESEARCH GOALS

Prior research has been conducted to determine that object detection algorithms are suitable to perform signal separation, but the goal of this research work is to investigate the performance of these algorithms given different time-frequency representations. In this work, different object detection algorithms will also be compared, which will shed more light on which algorithms are best suited for this problem.

1.6 RESEARCH OUTPUTS

The research described in this document has led to the submission of one journal article, the details of which are as follows:

Author(s): Llewellyn Strydom and Warren P. du Plessis
Title: Object Detection for Signal Separation with Different Time-Frequency Representations

Journal: IEEE Transactions on Signal Processing

Abstract: The task of detecting and separating multiple radio-frequency (RF) signals in a wideband scenario has attracted much interest recently, especially from the CR community. Many successful approaches in this field have been based on machine learning (ML) and computer vision methods using the wideband signal spectrogram as the only input feature. YOLO and R-CNN are deep learning-based object-detection algorithms that have been used to obtain state-of-the-art results on several computer vision benchmark tests. In this work, YOLOv2 and Faster R-CNN are implemented, trained and tested, to solve the signal separation task. Previous signal separation research does not consider representations other than the magnitude of the STFT and the spectrogram. Here, specific focus is placed on investigating different time-frequency representations based on the complex-valued STFT. Results are presented in terms of traditional object-detection metrics, and a mAP of 89.0% is achieved using Faster R-CNN with STFT magnitude as input.

1.7 OVERVIEW OF THE STUDY

The content of the chapters in this study are outlined below.

Chapter 1: The context of this work is introduced and discussed, and a motivation is provided as to why this research was conducted. A set of questions, objectives and goals is also outlined which clarifies the intention of this work.

Chapter 2: An introduction to deep learning and how it relates to computer vision, and object detection in particular, is provided. This includes details on different deep neural network architectures and their significance to the computer vision community. In this chapter, different object detection algorithms are also introduced and discussed, with a focus on Faster R-CNN and YOLOv2. Background to the signal separation problem is then provided in terms of a short literature review. The use of object detection algorithms to solve the signal separation problem is highlighted and methods that have been proposed in the literature are discussed.

Chapter 3: The procedures that were used to generate data, and train both Faster R-CNN and YOLOv2 are provided. These procedures include sufficient detail to allow other researchers to reproduce the datasets used and results generated.

Chapter 4: Empirical results on the signal separation dataset are reported for Faster R-CNN and YOLOv2. Different aspects of the results are analysed and possible explanations are given for

some of the results. Examples showing different scenarios are also included.

Chapter 5: A brief conclusion and some inspiration for future work is provided.

Two addenda are provided to explain concepts and terminology that are used throughout the rest of this work. The addenda are summarised below.

Addendum A: Deep learning researchers often assume that readers of their work are acquainted with many technical terms and concepts that form part of their work. A brief description of some of these terms is given, which will ensure that the terminology used in this work is clear.

Addendum B: Similar to the case in deep learning, there are many terms that are only known to those in the computer vision and object detection community. Relevant terms that appear in this work are discussed.

Chapter 2 BACKGROUND

2.1 INTRODUCTORY REMARKS

Prior research that supports this work is introduced and discussed in this chapter. The developments that have taken place in the field of deep learning has led to state-of-the-art results in many fields, and especially in computer vision. Many of these advancements provide the foundation for the work that is presented in this document, and are introduced in Section 2.2.

In Section 2.3, object detection, a sub-field of computer vision, is introduced and different object detection algorithms based on deep learning are discussed. Towards the end of this chapter, in Section 2.4, the signal separation problem that is investigated in this work is introduced and the publications that have inspired this work are discussed. The chapter ends with a summary in Section 2.5.

2.2 DEEP LEARNING AND COMPUTER VISION

The resurgence of neural networks, and specifically convolutional neural networks (CNNs), can somewhat be attributed to the success of AlexNet [12] in the 2012 ImageNet Large Scale Visual Recognition Challenge (ILSVRC); winning by a phenomenally large margin. CNNs were first introduced by Yann LeCun in 1989 [13], and although they showed promising results, they were overshadowed in both performance and popularity by other ML algorithms during the 1990s and early 2000s. The victory by the AlexNet team in 2012 showed that CNNs can vastly outperform other algorithms, *e.g.*, support vector machines (SVMs) [14] and tree-based methods [15], given enough data and processing power. The availability of large datasets and the wide adoption of graphics processing

unit (GPU) [12], and more recently, tensor processing unit (TPU) [16] technology, means that both of these hurdles have now been crossed.

Broadly, computer vision refers to the ability of machines to analyse, understand, and manipulate images and video. Computer vision tasks include image classification [12], object detection [17], semantic segmentation [18] and image captioning [19], amongst others. Many recent advances in computer vision can be directly attributed to advances in deep learning [20–23], and hence it makes sense to discuss these developments side-by-side. For example, in 2015 computers surpassed human-level performance on a difficult image classification task for the first time by leveraging the power of deep CNNs [21].

The rest of this chapter will comprise a discussion on deep learning architectures and methods that have contributed to the advancement of computer vision.

2.2.1 Deep Neural Network Architectures

There exists certain deep network architectures that have either introduced new, seminal concepts, or achieved such remarkable results on certain benchmarks, that they have been widely adopted by ML researchers and professionals. These networks are often used as a starting point when designing new architectures, and since pretrained weights are usually available for these networks, they are often used to perform transfer learning (see Section 2.2.3).

2.2.1.1 Visual Geometry Group

The Visual Geometry Group (VGG) family of CNN-based architectures were proposed by researchers from the University of Oxford [24]. Before VGG, the first few layers of most CNN architectures had large receptive fields (typically 5×5 [13]). Instead, VGG stacks several convolutional layers with small receptive fields (3×3) at the start of the network, which leads to fewer parameters and more non-linearities in between layers. This novel change resulted in the decision function of the VGG networks being more discriminative and also made it easier to train deep networks [24].

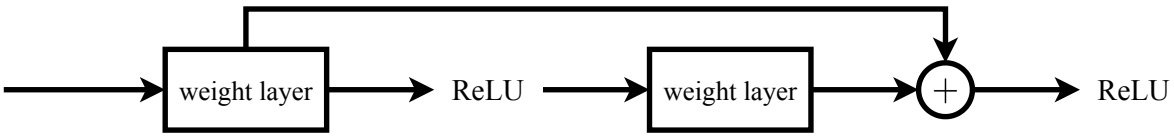


Figure 2.1. An illustration of the residual block with rectified linear unit (ReLU) activation functions. Adapted from [21], © 2015 IEEE.

VGG-16 is a network consisting of 16 weight layers that achieved a 92.7% top-5 test accuracy in the 2014 ILSVRC. R-CNN (discussed in Section 2.3.1) originally used VGG-16 as the feature extraction backbone for its own network [25].

2.2.1.2 ResNet

ResNet-152, consisting of 152 convolutional layers, won the 2016 ILSVRC with 96.4% accuracy [22]. The network is much deeper than any of the previous winners of the ILSVRC, which was made possible by the introduction of residual blocks (see Figure 2.1). Residual blocks allow extremely deep architectures to be trained efficiently by introducing identity skip connections between a layer's input and output. These blocks help to ensure that each layer learns new features, rather than simply repeating previous features. Skip connections also help resolve the vanishing gradient problem which has plagued deep neural networks in the past [22]. Other well-known variants of the ResNet family include ResNet-50 and ResNet-101.

The number of parameters that comprise the ResNet networks is smaller than most architectures that have approximately the same number of layers. For example, ResNet-18 consists of only 11.4 million parameters, while VGG-16 contains 134.7 million parameters. ResNet networks are a good option to replace the backbone of meta-algorithms, like R-CNN, as they provide a good trade-off between speed and accuracy [26].

2.2.1.3 Darknet Models

Darknet is an open-source neural network framework that was developed by Joseph Redmon, the creator of YOLO. The framework contains three pretrained networks: Darknet Reference Model, Darknet-19 and Darknet-53. These networks have all been trained for the 1 000-class ImageNet

challenge and make efficient use of common GPU architectures, which makes them very powerful compared to other networks with approximately the same number of parameters [27]. The networks are roughly based on AlexNet, but also employ ideas from GoogleNet [28], like 1×1 convolutions and batch normalisation[1]. The Darknet Reference Model is used as the backbone for YOLO Tiny, while Darknet19 is used for the original YOLO network and, Darknet-53 for YOLOv2.

2.2.2 PASCAL VOC

The PASCAL Visual Object Classes (VOC) challenge is the *de facto* benchmark for many computer vision problems; object detection algorithms in particular are often compared based on the results that they achieve on the PASCAL VOC challenge [29]. The challenge provides the computer vision and ML communities with a standard dataset of images and annotations, along with standard evaluation procedures. The challenge was organised annually (between 2005 and 2012) by the University of Oxford and includes several different object category recognition and detection tasks, including object detection and semantic segmentation.

All images contained in the PASCAL VOC dataset (since 2007) are collected from `flickr.com`, a popular photo-sharing website. Since these photos are not taken by (or selected by) researchers with a specific purpose in mind, the result is an unbiased dataset. The images contain a wide range of viewing conditions (pose, lighting, *etc.*) and images of a particular object are rare, *e.g.*, there are more images of motorcycles in street scenes, and fewer images where a motorcycle is the focus of the picture.

The PASCAL VOC dataset has labels for 20 common objects. They are: aeroplane, bicycle, bird, boat, bottle, bus, car, cat, chair, cow, diningtable [sic], dog, horse, motorbike, person, pottedplant [sic], sheep, sofa, train and tvmonitor [sic].

The PASCAL VOC detection task requires participants to predict the bounding boxes of each object of each of the 20 classes in a test image (if any), with associated real-valued confidence scores. Training and validation sets (commonly referred to as 'train' and 'val') are also provided, along with human-annotated bounding boxes and labels.

[1] Batch normalisation is explained in Addendum A.

The ground truth annotations are provided in Extensible Markup Language (XML) files and their contents are described in the development kit provided on the PASCAL VOC website [30]. The annotations include the bounding box coordinates, and also other metadata which includes a flag to indicate whether a specific object is difficult to detect. Objects that are marked as difficult are not used when evaluating results on the PASCAL VOC challenge.

This work uses the 2007 PASCAL VOC object detection dataset to verify the accuracy and correctness of custom implementations for different object detection algorithms. This is possible since results for these networks are known and published in peer-reviewed journals. The PASCAL VOC annotation format is also adopted for the signal separation dataset discussed in Chapter 3, as it allows the same codebase to be used throughout the verification and testing process.

2.2.3 Transfer Learning

Deep neural networks are notoriously difficult to train from scratch, and without the necessary resources, many attempts to do so are futile. Even in cases where an appropriately large dataset is available and convergence time is not important, starting with pretrained weights, rather than randomly initialised ones, can prove useful [31, 32]. Transfer learning is an approach where pretrained weights are used to initialise a network (or a portion of a network) before resuming training on a different dataset. Ideally features should be transferred from a similar problem, but even transferring features from distant tasks can lead to better results than using random initialisation [33].

Any pretrained network can be used as a starting point for transfer learning, but popular ones include AlexNet [12], VGG [24] and ResNet [22]. The architecture of these networks places a constraint on the final architecture of the target network, but since most problems do not require an entirely novel architecture, it is common to reuse existing network architectures to enable transfer learning.

The training process for transfer learning differs in some ways to training from scratch. Firstly, it is not necessary to retrain all the layers, and instead only the deeper layers are often fine-tuned [31]. This is because the earlier layers contain generic, low-level features that are applicable to many problems. The learning rate that is used during transfer learning is also often lower for the pretrained layers since

they are intuitively expected to be relatively good and it should thus not be necessary to make any drastic changes to them.

2.2.4 Complex Data Representations for Neural Networks

Neural networks that can natively support complex-valued data is an active field of research [34, 35]. Most deep learning frameworks (*e.g.*, PyTorch and TensorFlow) do however not support complex-valued data. This means that in fields like signal processing, where data is naturally represented by complex data types, researchers who would like to use deep learning as a tool to help them solve problems, have to use alternative representations of the data.

Automatic modulation classification is one field where researchers have been able to harness the power of deep learning by using alternative complex-valued data representations [36, 37]. The complex-valued signal data to be represented comprise the in-phase and quadrature (I/Q) components universally sampled by digital receivers. This data can be represented in either the time or frequency domain, and representing the complex values using only real numbers can be done by either considering its polar or rectangular coordinates.

These representations can be described mathematically by considering a discrete signal $\mathbf{s}$ that consists of samples with both an in-phase component s_i and a quadrature component s_q as in

$$s[n] = s_i[n] + js_q[n], \tag{2.1}$$

where j is the imaginary number $\sqrt{-1}$ and n is the index. The complex vector containing all N data samples is then given by

$$\mathbf{s} = [s[1], s[2], \ldots, s[N]]^T. \tag{2.2}$$

This complex-valued data vector $\mathbf{s} \in \mathbb{C}^N$ can then be translated into a real-valued feature vector $\mathbf{x} \in \mathbb{R}^N$, like

$$\mathbf{s} \mapsto \mathbf{x}. \tag{2.3}$$

The translation that gives the rectangular representation of the data may be written as

$$f : \mathbb{C}^N \to \mathbb{R}^{2\times N} \tag{2.4}$$

$$\mathbf{s} \mapsto \mathbf{x}^{IQ}, \tag{2.5}$$

where $\mathbf{x}^{IQ}$ is

$$\mathbf{x}^{IQ} = \begin{bmatrix} \mathbf{s}_i^T \\ \mathbf{s}_q^T \end{bmatrix}. \tag{2.6}$$

This is also sometimes referred to as the raw I/Q representation [37].

Another common representation is the polar representation which is given by

$$f : \mathbb{C}^N \to \mathbb{R}^{2\times N} \tag{2.7}$$

$$\mathbf{s} \mapsto \mathbf{x}^{A/\phi}, \tag{2.8}$$

where $\mathbf{x}^{A/\phi}$ is

$$\mathbf{x}^{A/\phi} = \begin{bmatrix} \mathbf{x}_A^T \\ \mathbf{x}_\phi^T \end{bmatrix}, \tag{2.9}$$

and the phase vector $\mathbf{x}_\phi \in \mathbb{R}^N$ and the magnitude vector $\mathbf{x}_A \in \mathbb{R}^N$ have the elements

$$x_A[n] = \sqrt{s_i[n]^2 + s_q[n]^2}, \text{ and} \tag{2.10}$$

$$x_\phi[n] = \arctan\left(\frac{s_q[n]}{s_i[n]}\right) \quad , \tag{2.11}$$

where $n = 1, 2, \ldots, N$.

Both the magnitude vector $\mathbf{x}_A$ and the phase vector $\mathbf{x}_\phi$ can also be used in isolation to create feature vectors, as in

$$f : \mathbb{C}^N \to \mathbb{R}^N \tag{2.12}$$

$$\mathbf{s} \mapsto \mathbf{x}^A \tag{2.13}$$

$$\mathbf{x}^A = \left[\mathbf{x}_A^T\right], \tag{2.14}$$

and

$$f : \mathbb{C}^N \to \mathbb{R}^N \tag{2.15}$$

$$\mathbf{s} \mapsto \mathbf{x}^\phi \tag{2.16}$$

$$\mathbf{x}^\phi = \left[\mathbf{x}_\phi^T\right]. \tag{2.17}$$

However, note that neither of the last two representations are reversible, as a complex value cannot be reconstructed given only its phase or magnitude. Depending on the algorithm being used, this property may or may not affect performance.

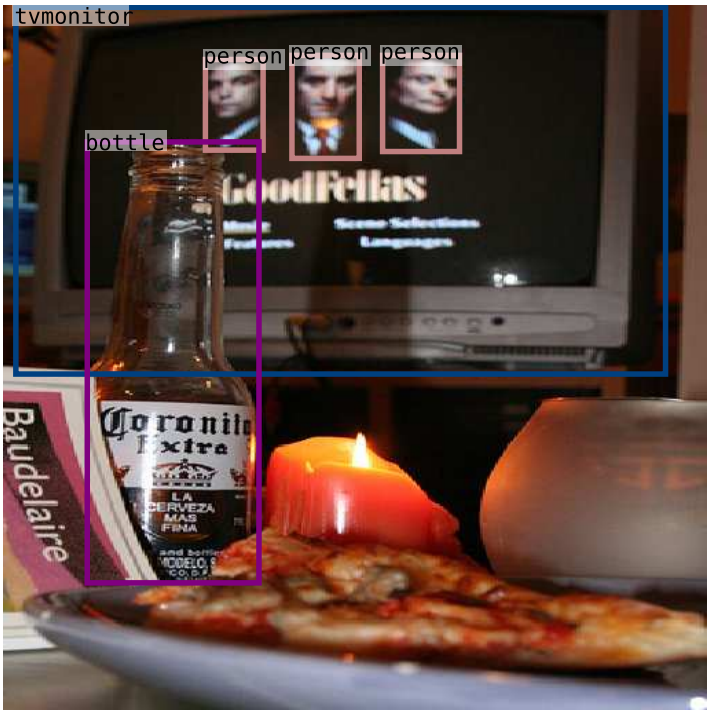


Figure 2.2. In this image (from [29]) each object is labelled according to its class and the exact location of each object is indicated by a rectangular bounding box. This example shows the ground truth annotations for this image, as labelled by a human.

2.3 OBJECT DETECTION

Object detection is the task of precisely estimating the class and location of multiple objects in an image [17]. Hence, the task of all object detection algorithms is to produce a list of semantic labels with bounding boxes, given an image. Object detection algorithms that are based on ML methods, aim to produce accurate predictions by training on a set of images that were labelled by people. An example of such a human annotated image is shown in Figure 2.2.

Object detection algorithms can broadly be divided into two categories: single-stage methods and two-stage methods. Two-stage methods, also called sparse methods, generate a list of possible regions of interest (ROIs), before classifying, and possibly refining, each region individually. In contrast, single stage methods (or dense methods) will take an image as input and produce bounding box outputs with class labels, in a single forward pass of the network without the need for any region proposals.

The R-CNN family of detectors comprise three well-known, two-stage object detectors [20, 25, 38].

The original variant of R-CNN [25], which stands for regions with CNN features, uses selective search [39] to determine possible RoIs in an image. These regions are passed through a classification network to determine whether certain objects are present in each region. An update to the R-CNN framework, called Fast R-CNN, improved the speed of the framework by eliminating the need to recalculate features for each RoI. This left the region proposal method as the bottleneck in the object detection pipeline [38]. Faster R-CNN aimed to reduce the time spent on generating region proposals by replacing selective search with a fully-convolutional network that produces region proposals. This also resulted in an improvement in the framework's accuracy [20].

YOLO [40] was the first single-stage object detection algorithm. The approach has since been refined and improved by the original author who has published his incremental improvements in the open access journal, ArXiv [23, 41]. Another single-stage object detection algorithm that rivals the latest version of YOLO is Single Shot MultiBox Detector (SSD) [42].

In this work only Faster R-CNN and YOLOv2 are considered. This decision was made as Faster R-CNN and YOLOv2 both provide a good trade-off between complexity, speed and accuracy. By investigating these two approaches, some conclusions can also be drawn about the viability of both single-stage and two-stage detectors for the signal separation task discussed in Chapter 2.4.

2.3.1 Regions with CNN Features

R-CNN is a simple and scalable two-stage object detection algorithm that achieved a mAP of 53.3% on the 2012 PASCAL VOC test set [25]. This was a 30% relative improvement on the previous best result at the time of its release in 2014. R-CNN uses selective search [39] to generate $\sim$2 000 category-independent region proposals. Each RoI is then cropped, scaled and passed through a CNN that performs feature extraction and generates a fixed-length feature vector. Category-specific linear SVMs are then used to classify each RoI as either background or belonging to one of C_k classes, based on the feature vector. The size and location of each RoI is refined by optimizing the regularised least squares objective, also known as ridge regression. The R-CNN framework is depicted in Figure 2.3.

Since R-CNN combines a CNN, multiple SVMs and ridge regression, it is not possible to train its parameters end-to-end, and instead each stage is trained and tuned separately; later research shows

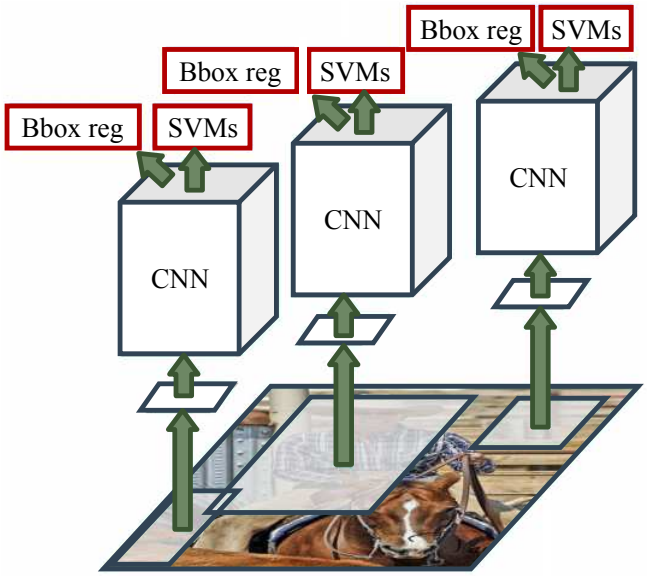


Figure 2.3. R-CNN was the state-of-the-art object detector when it was released in 2014. Redrawn from [38], © 2015 IEEE.

that this is not optimal [38]. R-CNN is also slow (inference takes approximately 10 s per image on a GPU) since each RoI has to be passed through the feature extractor individually, accounting for ∼2 000 passes through the CNN backbone.

R-CNN introduced a parameterisation of bounding boxes, which makes it possible to formulate the problem in such a way that ridge regression is possible. The same parameterisation is also used by Fast R-CNN and Faster R-CNN, and hence it is shown here. The relevant parameterisation is given by

$$t_x = \frac{x - x_{ROI}}{w_{ROI}}, \tag{2.18}$$

$$t_y = \frac{y - y_{ROI}}{h_{ROI}}, \tag{2.19}$$

$$t_w = \log\left(\frac{w}{w_{ROI}}\right), \text{ and} \tag{2.20}$$

$$t_h = \log\left(\frac{h}{h_{ROI}}\right), \tag{2.21}$$

where $t_\star$ ($\star \in x, y, w, h$) are the regression targets, and x, y, w, and h denote the centre coordinates, the width and the height of the ground truth bounding box. The location and dimensions of the RoI to be regressed is represented by $x_{ROI}, y_{ROI}, w_{ROI}$, and h_{ROI}.

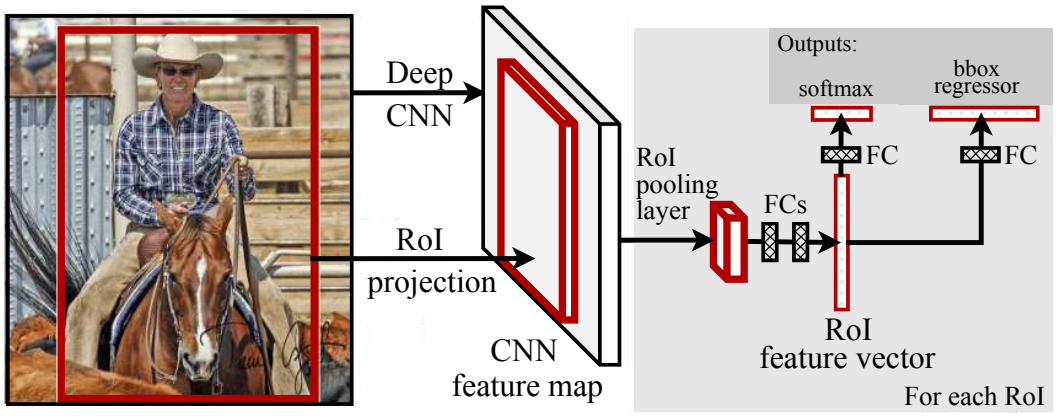


Figure 2.4. Fast R-CNN only requires one pass through the CNN feature extractor to generate features for all of the RoIs to consider. Each RoI is then processed by the RoI pooling layer before being passed through some fully-connected layers. Adapted from [38], © 2015 IEEE.

2.3.1.1 Fast R-CNN

Fast R-CNN is an updated version of R-CNN which reduces the amount of time required to test and train the network, and also results in improved detection accuracy [38]. Fast R-CNN is 213 times faster than R-CNN at test time and achieves a mAP of 68.4% on the 2012 PASCAL VOC dataset [38].

The improvement in run-time achieved by Fast R-CNN is mainly due to the fact that it performs feature extraction for all RoIs in only one forward pass. This is made possible by the RoI pooling layer which is introduced in the Fast R-CNN paper [38]. The Fast R-CNN framework is illustrated in Figure 2.6.

ROI Pool

As previously mentioned, R-CNN requires new features to be extracted for every RoI [38]. In order to extract RoI-specific features from a single global feature map, an operation that isolates local information and produces a fixed size output for any arbitrary input or RoI size is required. RoI pooling satisfies both of the aforementioned requirements and enables Fast R-CNN to calculate image-wide features only once, and to reuse those features for any number of individual RoIs.

The RoI pooling layer is similar to the spatial pyramid pooling (SPP) layer that was introduced before

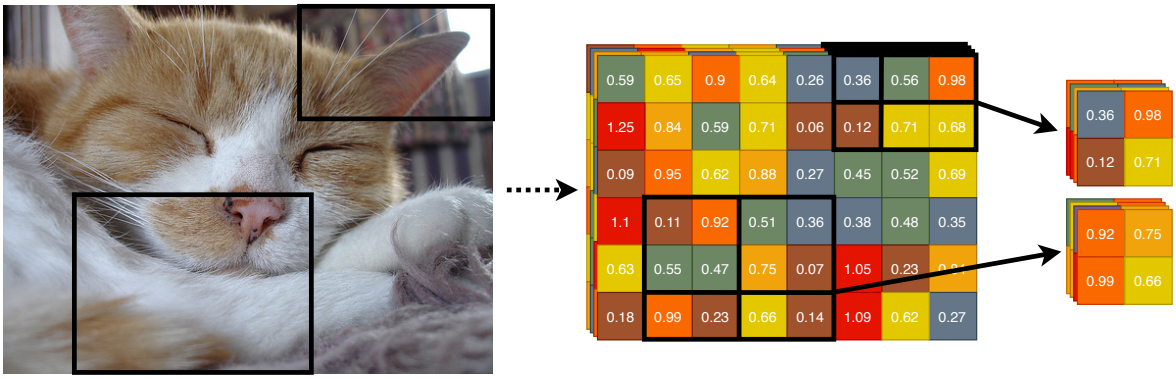


Figure 2.5. In this arbitrary example an image of a cat (from [29]) is transformed to a feature map with 4 channels. Two randomly-selected RoIs (indicated by black boxes) are extracted by RoI pool to create two new feature maps, both of size 2×2.

it, with the main (important) difference being that derivatives can be routed through the RoI pooling layer. This makes backpropagation possible, and allows Fast R-CNN to be trained end-to-end. Thus, the issue of training R-CNN in three separate stages, which is both slow and less accurate [38], is solved by the introduction of the RoI pool layer.

The operation of the RoI pooling layer is both simple and intuitive. RoI pooling divides all RoIs of arbitrary size $h \times w$ into $H \times W$ sub-windows, each of size $h/H \times w/W$ (rounded). If h is not divisible by H, or w is not divisible by W, then some sub-windows will be smaller than others. The maximum per-channel values from each of these sub-windows make up the output of the RoI pooling layer. This is similar to max pooling[2].

The values of H and W are hyperparameters that determine the ouput size of the RoI pooling layer. The default output size of the RoI pooling layer in Fast R-CNN is 7×7. An example showing the operation of the RoI pooling layer with H and W both set to 2 can be seen in Figure 2.5.

2.3.1.2 Faster R-CNN

Faster R-CNN further improves on the R-CNN framework by introducing the Region Proposal Network (RPN) to generate RoIs, thus replacing selective search [20]. Although this is the biggest difference between Fast R-CNN and Faster R-CNN, there are also other smaller differences, which help it to

[2]Max pooling is explained in Addendum A.

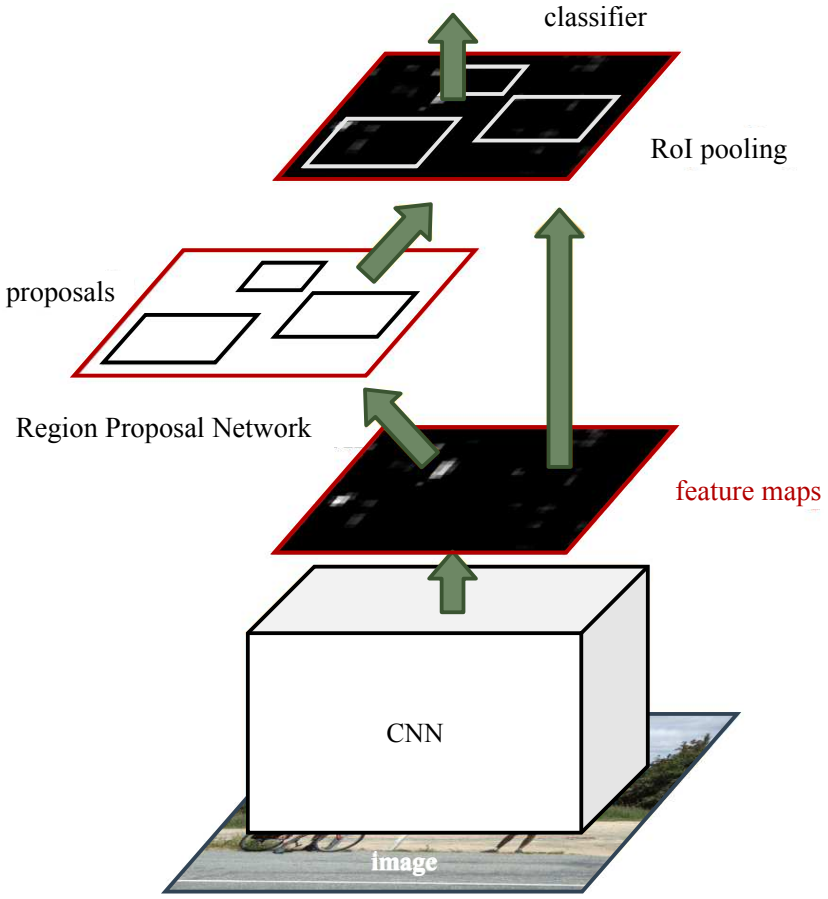


Figure 2.6. Faster R-CNN introduced the RPN to speed up training and inference. Apart from the region proposal stage, Faster R-CNN is more or less identical to Fast R-CNN. Redrawn from [20], © 2017 IEEE.

achieve a mAP of 70.4% on the 2012 PASCAL VOC object detection test set. The general framework for Faster R-CNN is shown in Figure 2.6.

Region Proposal Network

RPN is a novel neural network-based approach to genere class-agnostic region proposals. The region proposal problem is treated in much the same way as object detection, but instead of predicting multiple class labels, there are only two: object and no object. RPN predicts bounding boxes by applying bounding box deltas to anchors[3]. This allows RPN to learn a direct mapping from raw images to bounding boxes.

[3] Anchors are explained in Addendum A.

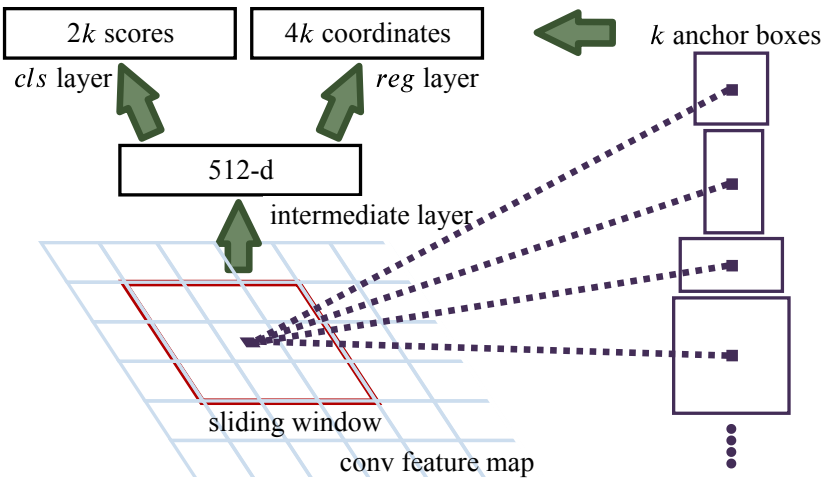


Figure 2.7. RPN replaces selective search in Faster R-CNN. Objectness scores and bounding box deltas are produced by RPN for anchors positioned at different points across the input image. Redrawn from [20], © 2017 IEEE.

RPN operates on the feature map at the end of the backbone network, which in the case of VGG-16, consists of 512 layers. A sliding window is applied to the feature map to produce classification and regression scores for k anchors at each position in the feature map. At each position, each anchor produces two scores to indicate how likely an anchor is to represent either the foreground (object) or background (no object) class. In addition to this, four regression scores (t_x, t_y, t_w, t_h) are produced that alter the size, aspect ratio and position of each of the anchors. The regression scores are applied to the anchors according to (2.18)–(2.21). The main idea behind the RPN is illustrated in Figure 2.7.

ROI Align

Although RoI pool works well for object detection, researchers found that the coarse features that it produces are inadequate for segmentation tasks. Mask R-CNN [43] consequently introduced a new pooling layer, RoI align, which removes the quantisation steps that are associated with RoI pool.

RoI pool performs two quantisation steps; first when RoIs are mapped to feature maps, and again when performing the pooling operation. RoI align is similar to RoI pool, but does not quantise the locations of the sub-windows, and uses a sampling technique to determine the correct values of the points inside of these sub-windows. Specifically, bilinear interpolation is used to calculate features at a fixed number of evenly-spaced points within each sub-window. The maximum value from each

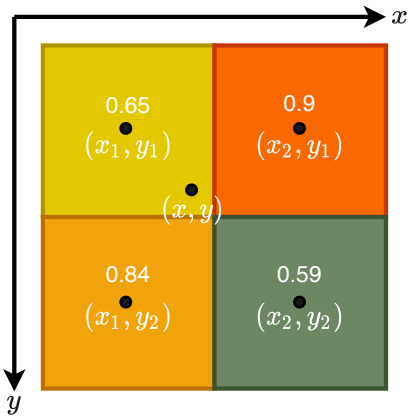


Figure 2.8. The bilinear interpolation step for RoI align. The value at (x, y) can be calculated as 0.74 by using (2.22).

sub-window is then reported to the output in order to condense the amount of information that is passed through the network. This is similar to other pooling functions.

The equation for performing bilinear interpolation is [44]

$$f(x,y) \approx \frac{y_2 - y}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(x_1, y_1) + \frac{x - x_1}{x_2 - x_1} f(x_2, y_1)\right) + \frac{y - y_1}{y_2 - y_1}\left(\frac{x_2 - x}{x_2 - x_1} f(x_1, y_2) + \frac{x - x_1}{x_2 - x_1} f(x_2, y_2)\right), \tag{2.22}$$

where (x, y) is the point of interest, (x_1, y_1) is the location of the nearest cell to the top-left of that point, and (x_2, y_2) is the nearest cell to the bottom-right. A visual example of the bilinear interpolation step for RoI align is shown in Figure 2.8

An example of RoI align is shown in Figure 2.9. In this example, only one RoI is shown and the output size of the RoI align layer is set to 2×2. Bilinear interpolation is performed on four equally spaced points within each sub-window, of which only the maximum value is reported to the output.

The features that are produced by the RoI align layer are accurate enough to predict the exact pixel location of certain features, which is why it is used for instance segmentation tasks [43]. But, the improved accuracy of RoI align also means that the overall accuracy of the Faster R-CNN framework can be improved by replacing the RoI pooling layer with RoI align.

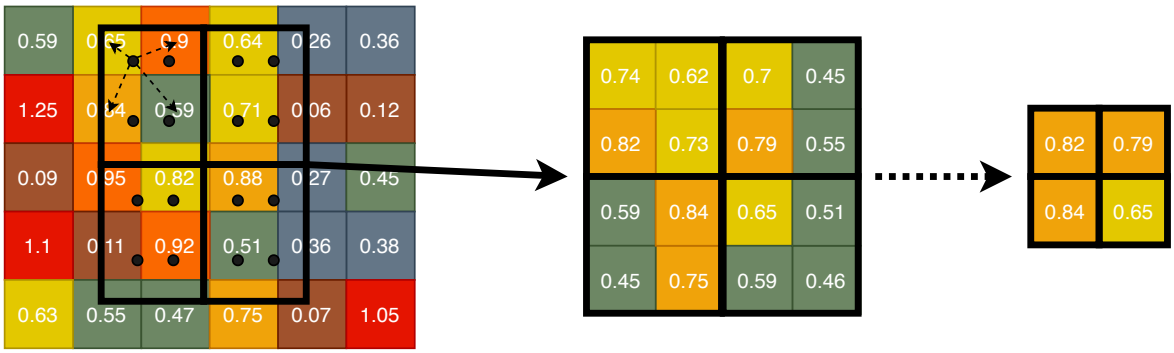


Figure 2.9. RoI align uses bilinear interpolation to sample points within a specified area without performing quantisation which results in a fixed-size feature map that maintains spatial features. Note that the top leftmost point that is also annotated with four arrows, corresponds to the example in Figure 2.8.

Loss Function

Training Faster R-CNN requires both the RPN and the detection head (Fast R-CNN) to be trained. This means that the loss function comprises two components, which can each be broken down further. At a high level, the loss for Faster R-CNN is given by

$$L_{FRCN}(p_i, t_i) = L_{RPN}(p_i, t_i) + L_{DET}(p_i, t_i). \tag{2.23}$$

The multi-task loss for RPN can be further broken down into a classification and a regression component respectively, as in

$$L_{RPN}(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^*) + \lambda \frac{1}{N_{reg}} \sum_i p_i^* L_{reg}(t_i, t_i^*), \tag{2.24}$$

where p_i is the predicted probability that the i-th anchor contains an object. The ground truth label p_i^* is 1 if the anchor is associated with a ground truth object, and 0 if it is deemed to be part of the background. The parameter t_i represents the four parameterised coordinates of the predicted bounding box, as discussed earlier [(2.18)–(2.21)], while t_i^* is the ground truth for these values. The two terms in the loss function are normalised by the mini-batch size N_{cls} and the total number of anchor locations N_{reg}, and weighed by λ which is set to 10 in the original implementation, so as to approximately have equal weightings for both parts [38].

In order to determine whether an anchor is associated with a ground truth object, the IoU between all anchors and all ground truth objects has to be calculated. The anchor with the greatest IoU with

each ground truth object, as well as all anchors that have an IoU greater than 0.7 with any object are labeled as foreground. Anchors for which the IoU with all objects is less than 0.3 are treated as not containing an object, and hence regression targets for these anchors are ignored. Anchors that have an IoU between 0.3 and 0.7 with any object are ignored entirely during training, and do not contribute to the loss function.

The classification loss L_{cls} for the RPN is the negative log-likelihood loss, as given by

$$L_{cls}(p,u) = -\log(p_u), \tag{2.25}$$

for true class u. The regression loss L_{reg} is the smooth L_1 loss, and is only calculated for RoIs that are associated with ground truth objects. This gives

$$L_{reg}\left(t^u, v\right) = \sum_{i \in x,y,w,h} \text{smooth}_{L_1}\left(t_i^u - v_i\right) \tag{2.26}$$

where

$$\text{smooth}_{L_1}(x) = \begin{cases} 0.5x^2 & \text{if } |x| < 1 \\ |x| - 0.5 & \text{otherwise.} \end{cases} \tag{2.27}$$

The loss that is associated with the detection head (Fast R-CNN) L_{DET} comprises the same terms as L_{RPN}, but instead of relating anchors to ground truth bounding boxes, predicted RoIs are related to ground truth boxes.

If the loss function for Faster R-CNN is calculated for all RoIs, then the background class will dominate the loss function. It is thus advised to sample a selection of background and foreground RoIs during training. For the RPN, 256 anchors in each image are sampled, where the positive and negative anchors have a ratio of up to $1:1$. Similarly, for the Fast R-CNN detection head, 64 RoIs are sampled from each image. Up to 25% of these RoIs have an IoU of at least 0.5 with a ground-truth bounding box, while the remaining RoIs are sampled from object proposals that have a maximum IoU with ground-truth bounding boxes in the interval $[0.1, 0.5)$. The lower bound of 0.1 on background RoIs is a form of hard negative mining, which ensures that the network learns from difficult examples [38].

Custom Implementation

A custom PyTorch [45] implementation of the original Faster R-CNN framework is used throughout this work. This allows full control over all aspects of the algorithm, and in order to ensure reproducibility, the source code is made available on GitHub [46].

Table 2.1. Accuracy (in % mAP) of different implementations of the Faster R-CNN framework.

Author(s)	Framework	Pooling	Backbone	PASCAL VOC	MS COCO
Girshick *et al.* [38]	Caffe [47]	RoI Pool	VGG-16	73.2	41.5
He *et al.* [22]			ResNet-101	76.4	48.4
Lin *et al.* [48]			ResNet-101 FPN	—	59.1
He *et al.* [43]		RoI Align	ResNet-101 FPN	—	59.6
Strydom [49]	PyTorch	RoI Pool	VGG-16	71.7	—
			ResNet-101	76.2	—
		RoI Align	ResNet-101	77.1	—

In order to verify that the custom implementation is correct, results on the 2007 PASCAL VOC test set are compared and reported in Table 2.1. The results for some implementations are only reported for Microsoft Common Objects in Context (COCO) (another popular computer vision dataset/competition), and are included here only for reference. The Microsoft COCO competition uses several different definitions of the mAP metric, but the results that are reported in Table 2.1 use the same mAP metric as PASCAL VOC, and as described in Section 2.3.3. The mAP score for common object detection algorithms on Microsoft COCO are evidently lower than for PASCAL VOC, because there are 80 classes compared to 20 for PASCAL VOC.

All experiments were conducted by replicating the training steps from the relevant original publications as closely as possible. In addition to the original implementation, results are also included for ResNet-101 and RoI align. The accuracy of the custom implementation is slightly worse than the previously reported results, but are deemed to be within a satisfactory margin so as to accept that the implementation is correct.

2.3.2 You Only Look Once

Detection systems prior to YOLO would repurpose classifiers to perform detection by applying the model to an image at multiple locations and scales [40]. YOLO introduced the idea of applying a single neural network to the full image by considering object detection as a regression problem of

sorts, and hence, became the first single-stage object detection method.

The basic premise behind YOLO is to divide each image into $S \times S$ cells and to predict B bounding boxes, along with objectness scores and class probabilities for each cell. If the objectness score for a bounding box is above some threshold, then the class with the highest probability is reported. If the objectness score is below the threshold, then that box is ignored, and assumed to belong to the background class.

Since YOLO shares all of the calculation for each image, it has the ability to reason globally about an image when making predictions [40]. This means that predictions are naturally informed by global context in the image.

YOLO is extremely fast, because it only requires a single forward pass of the neural network. This makes YOLO more than 1000 times faster than R-CNN and up to 100 times faster than Fast R-CNN [40].

YOLO's biggest drawback is its failure to predict small objects accurately [50]. This is one of the biggest facets on which YOLOv2 and YOLOv3 aim to improve.

2.3.2.1 YOLOv2

YOLOv2 proposed several novel improvements to the original YOLO implementation, which makes it faster and easier to train [41]. The biggest change from YOLO to YOLOv2 is the introduction of anchors. This also improves accuracy on small objects [41].

Like the original YOLO algorithm, YOLOv2 also splits each image into $S \times S$ cells, but centres B bounding box priors (or anchors) at each of the cells. If the centre of an object falls within a cell, the bounding box (centred at that same cell) with the maximum IoU with that object is deemed responsible for detecting that object during training. This is shown in Figure 2.10.

YOLOv2 produces four regression coordinates, an objectness score and K class probability scores for each bounding box (there are $S \times S \times B$ bounding boxes in total). The bounding boxes that are

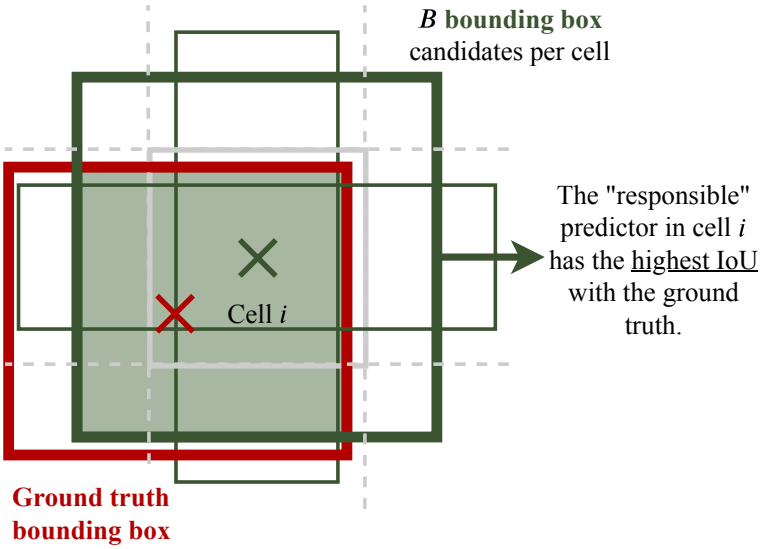


Figure 2.10. Bounding box candidates for one cell in the YOLO architecture is shown. The thick green box represents the anchor that has the greatest IoU with the red, ground truth bounding box, and is thus deemed responsible for that object. Redrawn from [51] with permission.

responsible for objects will have complete targets during training, while the remaining boxes will only be penalised based on their objectness scores, not their associated regression coordinates or class probabilities.

The regression coordinates that are produced by the network for a single bounding box are defined by a tuple of four values, (t_x, t_y, t_w, t_h). If the cell at which the bounding box is centred has coordinates (c_x, c_y), and the anchor is of width p_w and height p_h, then

$$b_x = \sigma(t_x) + c_x, \tag{2.28}$$

$$b_y = \sigma(t_y) + c_y, \tag{2.29}$$

$$b_w = p_w e^{t_w}, \text{and} \tag{2.30}$$

$$b_h = p_h e^{t_h}, \tag{2.31}$$

where $b_\star$ $(\star \in x, y, w, h)$ correspond to the coordinates of the predicted bounding box. These coordinates are visualised in Figure 2.11.

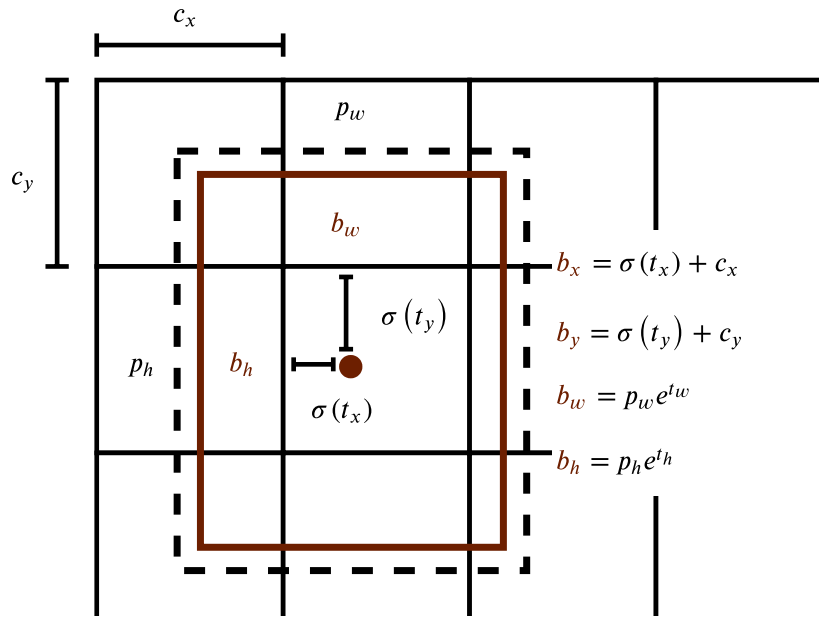


Figure 2.11. In YOLO, bounding boxes are parameterised with respect to anchors according to the following set of rules. The dotted box represents the anchor, while the ground truth object is represented by the maroon box. Redrawn from [41], © 2015 IEEE.

The objectness score t_o is a confidence score that predicts the likelihood that a specific bounding box prior contains an object. The target for the objectness score during training is

$$\sigma(t_o) = \Pr(\text{object}) \times \text{IoU(b, object)}. \tag{2.32}$$

For each bounding box prior, a probability score that an object belongs to class $C_i, i = 1, \ldots, K$ is generated. The probability score represents a conditional probability, which depends on whether an object is contained within that bounding box. The posterior probability that an object from class C_i is contained within a given bounding box prior is given by

$$\Pr(C_i) = \Pr(C_i|\text{object}) \times \Pr(\text{object}), \tag{2.33}$$

where $\Pr(\text{object}) \approx \sigma(t_o)$ from earlier.

Loss Function

The loss function for YOLOv2 is

$$\begin{aligned} L =& \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[(x_i - \hat{x}_i)^2 + (y_i - \hat{y}_i)^2 \right] \\ &+ \lambda_{\text{coord}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left[\left(\sqrt{w_i} - \sqrt{\hat{w}_i} \right)^2 + \left(\sqrt{h_i} - \sqrt{\hat{h}_i} \right)^2 \right] \\ &+ \lambda_{\text{obj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \left(C_i - \hat{C}_i \right)^2 \\ &+ \lambda_{\text{noobj}} \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{noobj}} \left(C_i - \hat{C}_i \right)^2 \\ &+ \sum_{i=0}^{S^2} \sum_{j=0}^{B} \mathbb{1}_{ij}^{\text{obj}} \sum_{c \in \text{ classes}} \left(p_i(c) - \hat{p}_i(c) \right)^2 . \end{aligned} \tag{2.34}$$

This looks immensely complex at first sight, but once it is broken down, it becomes a lot easier to understand. Note that the first three terms, as well as the last term, in (2.34) are only applied to anchor boxes that are responsible for detecting an object.

The first term is applied to correct the centre position of each predicted bounding box. This term is simply the sum of the squared errors for the predicted x and y coordinates with respect to the ground truth coordinates. The second term is similar, but corrects the width and the height of the predicted bounding boxes. The square root of the width and the height was used because these errors should have a larger impact on small bounding boxes than on big ones [41].

The next two terms in (2.34) are identical, except for the fact that they are weighted differently and that the first one is applied only to anchors that are responsible for objects, while the second is applied to all anchors that are not responsible for objects. These squared error terms are used for learning the objectness scores associated with each anchor, *i.e.* how likely they are to contain objects.

The last term of the YOLO loss function is the error associated with class predictions. Traditionally this would be a cross entropy loss, but YOLOv2 uses the mean squared error loss.

Table 2.2. Accuracy (in % mAP) of different implementations of the YOLOv2 algorithm.

Author(s)	Framework	Backbone	PASCAL VOC
Redmon and Farhadi [41]	Darknet	Darknet53	76.8
Strydom [49]	PyTorch		71.4

Custom Implementation

A custom PyTorch implementation of YOLOv2 supported this work, and is available online [52]. The custom implementation aims to replicate all the details outlined in the YOLOv2 paper and seen in the original code base. Interestingly, the official code base includes some minor differences from the published paper. The irregularity that had the biggest impact on the accuracy of the custom implementation, is that an extra term is added to the loss function for the first 12 800 iterations. This added term forces all anchors that are not responsible for objects to retain their original dimensions.

The accuracy of the implementation that accompanies this work is slightly lower than that of the original implementation, and although the reason for this is unknown, it was deemed good enough to use as is. The accuracy for the custom YOLOv2 algorithm on the 2007 PASCAL VOC challenge is given in Table 2.2.

2.3.2.2 YOLOv3

YOLOv3 is based on YOLOv2 and most of the proposed changes were inferred from other publications that came after YOLOv2, and are not entirely new or novel [23]. The changes did however increase the accuracy of the YOLO framework quite substantially, but also decreased the speed at which YOLO can be trained and at which it performs inference.

2.3.3 Metrics

IoU (also known as the Jaccard index) is a statistic that is widely used in object detection to measure the similarity between a prediction and an associated ground truth label [17]. In object detection, the

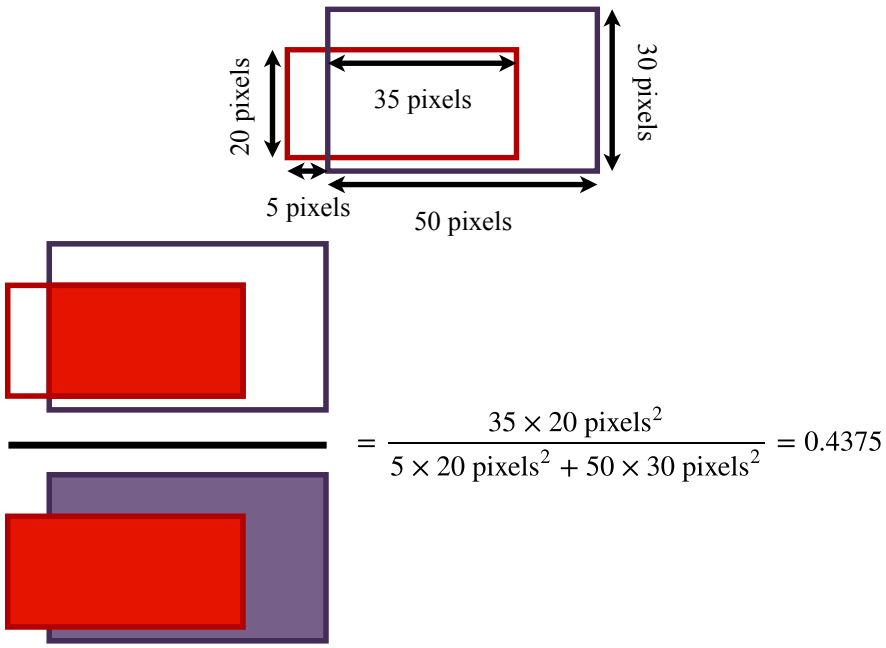


Figure 2.12. An example of calculating the IoU between a predicted and a ground truth bounding box for object detection. Note that the IoU is not altered if the predicted bounding box and the ground truth bounding box are swapped.

IoU is calculated as the area of overlap between a predicted and ground truth bounding box, divided by the area of union between these boxes, as shown in Figure 2.12. The equation for calculating the IoU is

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|} \tag{2.35}$$

$$= \frac{|A \cap B|}{|A| + |B| - |A \cap B|}, \tag{2.36}$$

where A and B are the predicted and ground truth predictions respectively. The IoU between two bounding boxes falls in the range $[0,1]$.

The IoU provides a metric for how good a single detection is, but in order to quantify the accuracy of an object detector over an entire dataset, another metric has to be used. This is where average precision (AP) and mAP comes in useful. The AP is calculated as the area under the precision-recall graph, which is obtained by varying the threshold of a certain detector. The mAP is the average per-class AP over all classes.

The precision P is the ratio between the number of true positives and the total number of predicted

positives, and indicates how likely any predicted positive is to be correct. Recall R is defined as the ratio between the number of true positives and the total number of ground truth positives in the set, and indicates how likely a ground truth positive is to be classified as such. These metrics can be calculated as

$$P = \frac{T_p}{T_p + F_p}, \text{ and} \tag{2.37}$$

$$R = \frac{T_p}{T_p + F_n}, \tag{2.38}$$

where T_p is the true positive rate, and F_p and F_n are the false positive and negative rates respectively.

The AP, and thus mAP, of a detector will be influenced by how true positives are defined. True positives are defined as bounding box predictions that have an IoU greater than some threshold with any ground truth bounding box of the same class. Note that the value of this threshold will have a large impact on the mAP. In the PASCAL VOC detection competition, an IoU threshold of 0.5 is used to define true positives when calculating the mAP of a detector [29]. In contrast, the Microsoft COCO detection competition calculates the mAP for 10 different IoU thresholds between 0.5 and 0.95 (spaced at intervals of 0.05) and calculates the average of these values. The metric for the Microsoft COCO competition should thus technically be called the average mAP, or written out, the average mean average precision. There has been some controversy regarding which metric (VOC or COCO) is more useful [23], especially since humans even find it difficult to distinguish between bounding boxes with an IoU of 0.3 and 0.5 [53]. However, for now, both metrics are used and widely accepted. This report will only use the PASCAL VOC metric (AP@0.5).

2.4 SIGNAL SEPARATION

Wideband spectrum sensing requires multiple signals of varying types to be detected and localised over a wide frequency range. In this work, this task is referred to as signal separation. Signal separation cannot be performed by traditional narrowband spectrum sensing algorithms, *e.g.*, matched filtering, energy detection, and cyclostationary detection, since these algorithms only detect the presence (or absence) of a single signal [1].

Multiple signals that may be overlapping in both time and frequency are considered for the signal separation task, and hence the centre frequency, bandwidth, and start and stop times have to be

estimated separately for each signal. Traditional wideband spectrum sensing approaches, including multiband joint detection (MJD), wavelet-based approaches and filter bank spectrum sensing, often fail under these conditions.

Recently a limited number of algorithms have been proposed to solve the task of signal separation using object detection and other computer vision algorithms. These algorithms are all based on extracting information from the wideband STFT and the related spectrogram [6–10].

2.4.1 Methods for Signal Separation

In the past, energy detection-based methods have been used to detect and localise signals from spectrograms due to their effectiveness when information about the signals is not known [54]. It has been proposed that energy detection can be combined with an image processing approach to perform signal separation [7]. This approach applies an energy detector to each STFT segment, in order to produce a binary spectrogram. Image processing techniques are then used to remove noise and draw bounding boxes around signals in the spectrogram. This technique works well for signals that are separated in time and frequency, but fails when signals are close to one another in the time-frequency plane.

In order to overcome the shortcomings of energy detection-based methods, it has been proposed to repurpose object detection algorithms for the task of signal separation [6]. The researchers who proposed the idea argued that, compared to naïve energy thresholding-based signal detection schemes, their approach could achieve higher levels of contextual understanding, which would lead to improved sensitivity performance. The main idea behind the approach was to use a modified version of YOLO to detect and localise signals in a spectrogram. The method was used to enter the Defense Advanced Research Projects Agency (DARPA) Battle of the ModRecs competition, and although their paper indicates that they believed that they would have outperformed traditional signal detection methods, the results for the competition were unfortunately never released.

In a 2018 paper, the concept of “eventness” was proposed for audio event detection [55]. This was seen as an analogue to “objectness” from computer vision, and the key observation behind the idea was that audio events reveal themselves as 2-dimensional time-frequency patterns with specific textures and

geometric structures in spectrograms. The idea that time-frequency patterns are analogous to objects occurring in natural images, is the same whether the signals are audio signals or RF signals. This was the first time that audio event detection had been approached from a vision-inspired angle. They used the RPN from Faster R-CNN to construct their model, which they showed could detect different audio events ranging from sirens to human voices [55]. The accuracy of their model was comparable to a state-of-the-art baseline model that is also based on a neural network [55].

Another application where object detection methods have been used to separate signals – this time RF – is in the suppression of radio frequency interference in synthetic aperture radar (SAR) echo signals [56]. The problem is posed as an underdetermined blind source separation problem, and an SSD model is trained to detect, localise and identify interference in the time-frequency domain. This method uses the STFT to produce a time-frequency graph which is fed to the detection network. They show, using simulated data, that this method can effectively increase the signal-to-interference-noise ratio of contaminated SAR echoes and improve the peak sidelobe ratio after pulse compression [56].

Automatic modulation classification for multiple signals is a problem that has not received much attention, as the assumption is often made that signals will be separated before being classified [36]. A novel approach has however been proposed to combine the processes of detecting, separating and classifying modulated signals [8]. This approach is based on the popular SSD object detection framework for detecting signals and they consider several digital modulation types including M-phase-shift keying (PSK), M-frequency-shift keying (FSK) and M-quadrature amplitude modulation (QAM). They use the spectrogram to represent the time-frequency spectrum, and show mathematically that the time-frequency characteristics of M-FSK signals is such that they can be identified from the spectrogram. Meanwhile, for M-PSK and M-QAM signals, the differences in the spectrogram are insufficient to identify the signal modulation, and hence if these signals are detected, eye and vector diagrams are constructed for these signals, which are then classified by a separate classification network [8].

Object detection methods may incur some unnecessary overhead, and hence a method that is inspired by object detection algorithms, but that predicts signal centrelines, rather than bounding boxes, has been proposed [9]. The method has been shown to be both more accurate and faster than object detection algorithms [9], but this has not been confirmed by other researchers. The method uses CNNs and is loosely based on SSD, but abandons the need for candidate anchors, which makes it faster.

The network is also able to perform signal classification for a subset of signal types (2-FSK, 4-FSK, PSK/QAM, resident noise, Morse and speech). This indicates that this approach is also only able to classify signals that have distinct time-frequency characteristics.

Radio Access Technology (RAT) classification is used to detect and classify RATs that coexist in the same band, to facilitate spectrum sharing [11]. Object detection has recently been proposed to perform this task [11]. This method can identify critical parameters of RAT technologies, such as frame duration, inter-frame duration, centre frequency, and signal bandwidth by using YOLO to extract features from spectrograms. The method was tested on real Wi-Fi and Long-Term Evolution (LTE) transmissions, and was shown to be very accurate.

The task of Internet of Things (IoT) discovery is almost identical to RAT classification, but focuses specifically on IoT devices. It should then come as no surprise that an object detection-based approach has been proposed for IoT discovery [10]. This approach also uses YOLO and is used primarily to detect and classify LoRaWAN transmission modes [10]. Preliminary results were also reported for Zigbee, Bluetooth and Wi-Fi signals. This approach projects spectrograms onto colour maps, as they would be viewed by humans on computer screens, and feed these images into the YOLO detection network.

2.4.2 Joint Time-Frequency Analysis

Signal separation requires that signals be precisely located in the time and frequency domains. A common time-frequency representation of signals is the STFT. The STFT is a Fourier-based transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. The STFT can be computed by dividing a time signal into equal length segments and computing the Fourier transform of each segment separately. The discrete STFT can be expressed as

$$\text{STFT}\{x[n]\}(m,f) \equiv \mathbf{STFT} \tag{2.39}$$

$$\mathbf{STFT} = \sum_{n=-\infty}^{\infty} x[n]w[n-m]e^{-j2\pi fn} \tag{2.40}$$

for a signal x and window w, where n is the sample index. The secondary index m is used to refer to each window of the STFT, which has different frequency components that are indexed by f. Here, the Hanning window is used [57].

The spectrogram is often regarded as a visual representation of the frequency spectrum of a signal over time. The spectrogram and the STFT are in fact very closely related, as the spectrogram is calculated by taking the squared magnitude of the STFT as in

$$\text{spectrogram}\{x[n]\}(m,f) \equiv |\mathbf{STFT}|^2 . \tag{2.41}$$

The Wigner-Ville distribution (WVD) is a quadratic transform that is used for joint time-frequency analysis (JTFA). This transform has higher spectral resolution than the STFT, while maintaining the same temporal resolution, but suffers greatly from cross-terms when multiple signals or signals with multiple components are to be considered. The Choi-Williams distribution (CWD) is based on the WVD, but reduces the number and extent of the cross terms. The WVD and CWD have been used for low probability of interception (LPI) radar pulse classification [58], and also mode identification [59].

Quadratic transforms are not considered in this work, since cross-terms may complicate the wideband signal separation task. Scale transforms, *e.g.*, the discrete wavelet transform (DWT), are also not considered since extracting time and frequency information from these transforms is a non-trivial, non-linear task [60].

2.5 SUMMARY

Developments in deep learning that provided the foundation for the work that is presented in this document were introduced. Specifically, the deep learning architectures that form the backbone of the deep learning-based object detection algorithms were discussed.

Complex values are often used in signal processing, and in this chapter the current state of complex neural networks were discussed. Neural networks that support complex-valued weights and inputs are not mature enough to warrant their widespread use, and instead different real-valued representations of complex values for neural networks were introduced. These real-valued representations will be used in the rest of this work.

In this chapter, different object detection paradigms (single-stage and two-stage) were described, and examples of each were provided. Faster R-CNN and YOLOv2 are two popular object detection

algorithms, each representing one of the paradigms. Details on how they work and how they are trained were provided and discussed in detail. Custom implementations of both of these algorithms were also introduced and tests were conducted to ensure that their accuracy is comparable to the original implementations.

The most commonly used metric for object detection is mAP. A description of this metric and how it is calculated was provided. This metric will be used throughout the rest of the work, and having a good grasp of the concept is important in order to understand the results that are presented later in this work.

The signal separation task that is considered throughout the rest of this work was introduced. The task is a specific case of wideband spectrum sensing, where possibly overlapping signals have to be detected and localised in the time-frequency domain.

Methods based on object detection algorithms have been proposed for the task of signal separation and have been shown to produce promising results. The algorithms and features that these methods use were discussed. An introduction to different time-frequency transforms was provided with specific emphasis on the STFT, which will be used for the signal separation task in this work.

In Chapter 3, the dataset that was used throughout this work is introduced, along with details about how the different object detection algorithms were trained.

Chapter 3 IMPLEMENTATION AND EXPERIMENTAL PROCEDURE

3.1 INTRODUCTORY REMARKS

This chapter includes information pertaining to how the experiments that led to the results contained in Chapter 4 were conducted. This includes a description of the dataset that was used and the motivation behind certain design choices relating to the dataset. The steps and parameters that were used to train the object detection algorithms are also described in this chapter to allow other researchers to reproduce the results that were obtained as part of this work.

The work described in this chapter forms part of a submitted journal paper [49].

3.2 DATASET GENERATION

Training and testing signal separation methods require the collection, or generation, of realistic data. Since this work utilises deep learning techniques that require vast amounts of data, all of the required data is generated synthetically using appropriate software packages and simulation tools, rather than recording real signals. The tools that were used include GNU Radio 3.8.1.0 and Python 3.6 with NumPy 1.17.0, matplotlib 3.1.1 and Pillow 6.1.0.

3.2.1 Narrowband Signal Generation

Three different signal types were generated so as to represent a variety of realistic communication and radar signals. This also makes it possible to investigate whether the object detection methods can discriminate between different signal types. The signal modulations that were used, in alphabetical order, are

- frequency modulation (FM),
- Gaussian minimum shift keying (GMSK), and
- linear frequency modulation (LFM) – both continuous wave (CW) and pulsed.

In the discussion that follows, $S(t)$ refers to the modulated signal and $m(t)$ is the message signal, which is given by $A_m \cos(2\pi f_m t)$ for analogue modulations. A_c and A_m refer to the respective magnitudes of the carrier and the message, and similarly, f_c and f_m represent the frequency of the carrier and the message respectively. The variable T is used to refer to the symbol duration for digital modulations, which is equal to the inverse of the symbol rate R_b.

3.2.1.1 Frequency Modulation

FM is used to encode information in a carrier signal by varying the instantaneous frequency of the signal. The equation for a FM signal is given by

$$S(t) = A_c \cos\left(2\pi \int_0^t f(\tau) d\tau\right) \tag{3.1}$$

$$= A_c \cos\left(2\pi \int_0^t [f_c + f_\Delta m(\tau)]\, d\tau\right) \tag{3.2}$$

$$= A_c \cos\left(2\pi f_c t + 2\pi f_\Delta \int_0^t m(\tau) d\tau\right) \tag{3.3}$$

$$= A_c \cos\left(2\pi f_c t + \frac{A_m f_\Delta}{f_m} \sin(2\pi f_m t)\right), \tag{3.4}$$

where $f(\tau)$ is the instantaneous frequency of the oscillator and f_Δ is the maximum absolute frequency deviation from f_c, when $m(t)$ is limited to $[-1; 1]$.

In this work, four different recordings are used as input when generating FM signals to ensure variety and to make the simulations more realistic. These four files are simply referred to as #1, #2, #3 and #4,

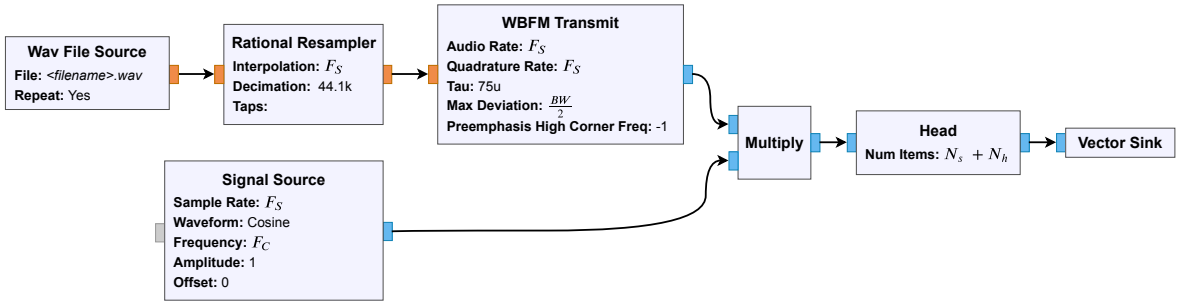


Figure 3.1. The GNU Radio Companion flowchart equivalent of the blocks that were used to simulate FM signals.

and contain rock music, pop music, classical music and human speech, respectively.

In 2020, FM signals are still used in some countries around the world for high fidelity radio broadcasts. The advent of CR, as well as rogue, amateur radio operators has increased the likelihood that these type of signals may overlap, and as such it is realistic to expect scenarios where some method will be required to detect and separate these signals.

GNU Radio contains a block for generating FM signals, and this block was used to ensure that the simulations are correct. A GNU Radio flowchart depicting the blocks that were used to generate FM signals is shown in Figure 3.1.

3.2.1.2 Gaussian Minimum Shift Keying

Digital modulation schemes are used for the analogue transmission of digital data, *i.e.* bits. GMSK is a special case of a broader class of modulation schemes, called FSK. In FSK signaling, bits are represented by signals of different frequencies. Binary FSK (BFSK) refers to FSK with two distinct levels represented by the two signals $S_0(t)$ or $S_1(t)$. These signals are transmitted to represent either a binary 0 or 1, with

$$S_0(t) = A_c \cos\left(2\pi\left(f_c - \Delta f\right)t\right), \quad 0 < t \leq T, \text{ and} \tag{3.5}$$

$$S_1(t) = A_c \cos\left(2\pi\left(f_c + \Delta f\right)t\right), \quad 0 < t \leq T. \tag{3.6}$$

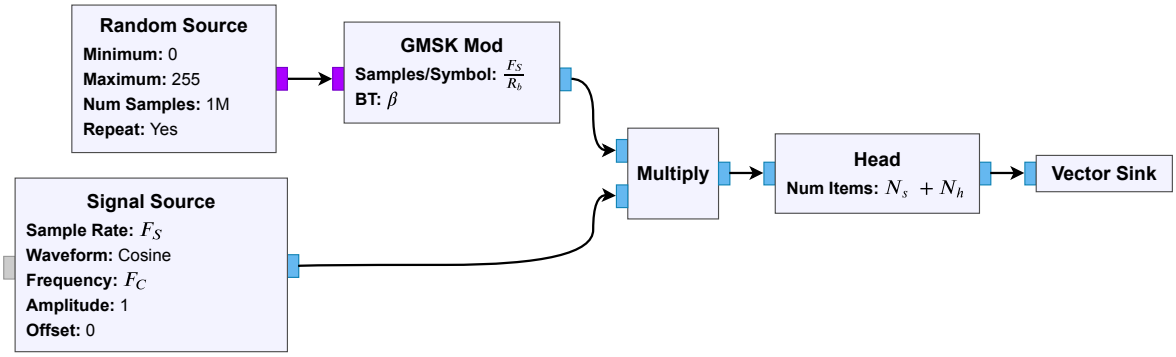


Figure 3.2. The GNU Radio Companion flowchart equivalent of the blocks that were used to simulate GMSK signals.

GMSK is more spectrally efficient than BFSK as there are no abrupt phase changes at the bit-transmission instants, as is characteristic of general FSK. This is achieved by setting $\Delta f = 0.25/T$, or equivalently $\Delta f = 0.25R_b$. A Gaussian pulse shaping filter is also used at the transmitter to further reduce the signal bandwidth.

GMSK is the digital modulation scheme preferred by the Global System for Mobile Communications (GSM) standard, and is thus used for cellular communications worldwide. There are several licensed frequency bands at which cellular communications operate, and as the number of cellphone users and the demand for high speed communication links increase, so do the frequencies at which these technologies have to operate.

These requirements have given rise to communication standards and systems that operate at very high frequencies that have previously been assigned for use by radar systems. The result is that GSM, and GSM-like signals, are overlapping with radar signals more frequently than before. This overlap, which occurs both in time and in frequency, means that modern military systems like electronic intelligence (ELINT) receivers and radar warning receivers (RWRs), are required to separate these signals before performing further processing.

GNU Radio makes it easy to generate GMSK signals by including a block that performs GMSK modulation. The other blocks that are used to generate GMSK signals representing pseudo-random data is shown in Figure 3.2.

3.2.1.3 Linear Frequency Modulation

In an LFM pulse, the instantaneous frequency (*i.e.* the rate of change of phase) increases or decreases linearly over the duration of the pulse. An LFM pulse can be represented mathematically as

$$f(t) = A_c \cos(\theta(t)) \tag{3.7}$$

$$= A_c \cos\left(2\pi\left\{f_c - \frac{BW}{2}\right\}t + \left\{\frac{BW}{\tau}\right\}\pi t^2 + \theta_0\right), \tag{3.8}$$

where A_c is the maximum pulse amplitude, f_c is the centre carrier frequency and BW is the total signal bandwidth.

The LFM pulse can be well understood by inspecting the derivative of the phase with respect to time

$$\partial\theta(t)/\partial t = 2\pi\left\{f_c - \frac{BW}{2}\right\} + 2\pi\left\{\frac{BW}{\tau}\right\}t, \tag{3.9}$$

which can be shown to vary linearly from $2\pi\left(f_c - \frac{BW}{2}\right)$ to $2\pi\left(f_c + \frac{BW}{2}\right)$ over the duration of the pulse.

LFM pulses are commonly used in radar applications as they have good Doppler tolerance, are easy to generate, and when pulse compression is used, they provide better range resolution than an unmodulated pulse of the same duration [61]. Mention was made earlier as to the fact that it is becoming more common for communications signals and radar signals to overlap, and hence LFM signals are included in this dataset. An LFM radar can operate in either a CW or pulsed fashion. In this work both CW and pulsed LFM signals with a duty cycle of 50% are considered.

Figures 3.3 and 3.4 show the fundamental blocks that can be used to generate LFM pulses in GNU Radio. This is by no means the only way to generate LFM pulses in GNU Radio, and does not necessarily offer an accurate represention of how these signals are generated in real radar systems.

3.2.2 Wideband Signal Generation

The aforementioned signals (FM, GMSK and LFM) are all generated one-by-one before they are scaled, added together and have the desired amount of Gaussian noise added to them. This is done to simulate scenarios in which multiple signals occupy the same part of the spectrum while having fine

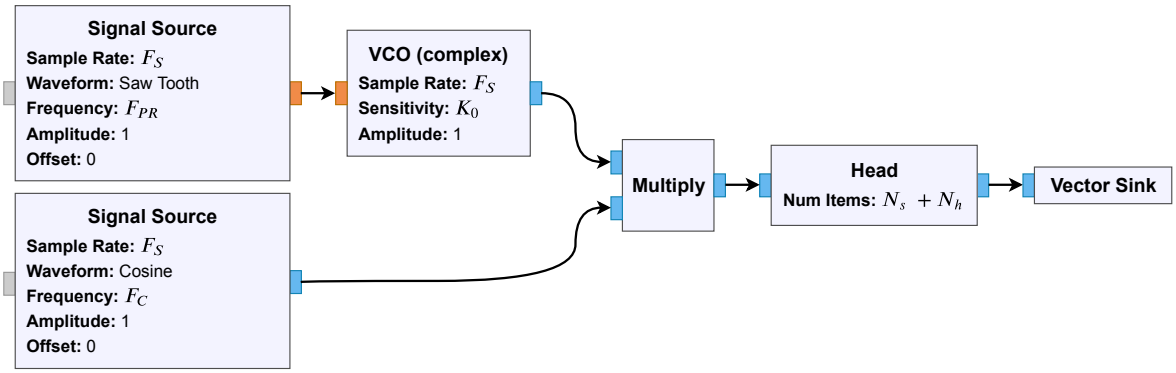


Figure 3.3. The GNU Radio Companion flowchart equivalent of the blocks that were used to simulate continuous wave LFM signals.

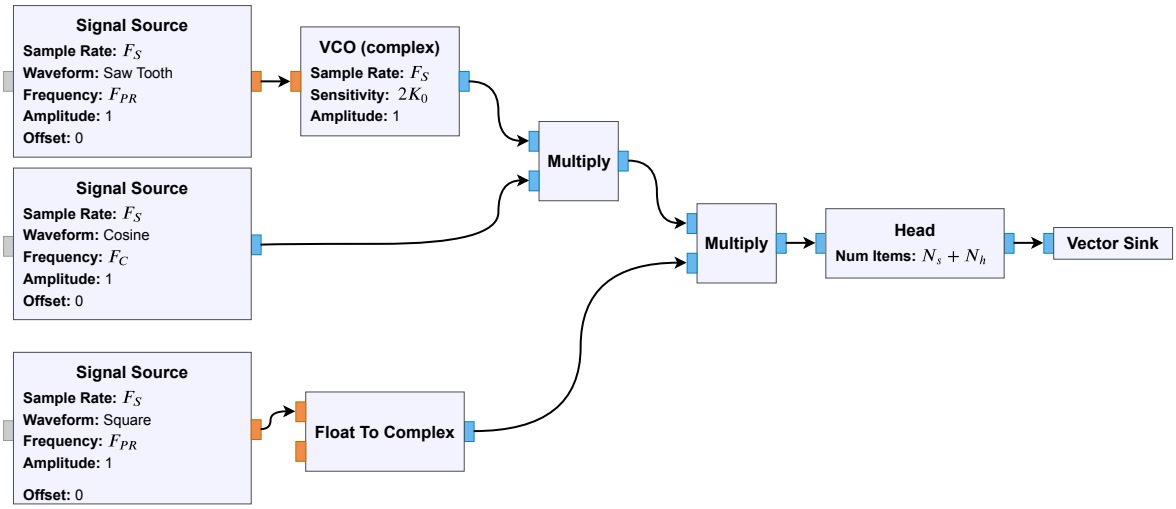


Figure 3.4. The GNU Radio Companion flowchart equivalent of the blocks that were used to simulate pulsed LFM signals.

control over parameters such as the bandwidth and SNR of each signal. Below is a step-by-step guide to how the signals are generated and combined.

1. Generate $N_s + N_h$ I/Q samples of any of the 3 signal types mentioned above using GNU Radio.
2. Discard the first N_s samples to ensure that the transmitter has started up properly.
3. Calculate the power P of the signal, and divide each sample by $\sqrt{P}$. This ensures that the signal has unity power.
4. Multiply each sample by $10^{\frac{SNR}{20}}$ to obtain the desired SNR.

Table 3.1. Signal parameters used to generate the signal separation dataset. Values indicated by $[x_{min}, x_{max}]$ are drawn from a uniform distribution, while those indicated as $\{x_0, x_1, x_2\}$ are randomly chosen with equal probability.

	FM	**GMSK**	**LFM**
F_C (MHz)	$[-4.75, 4.75]$	$[-4.5, 4.5]$	$[-4.5, 4.5]$
Bandwidth (kHz)	$[100, 400]$	$[100, 1\,000]$	$\pm[100, 1\,000]$
CW	True	True	{True, False}
Input File	{#1, #2, #3, #4}	0, 1	0, 1
PRF (Hz)	–	–	$[32, 256]$
β	–	$\{0.3, 0.5\}$	–
SNR	$[-10, 20]^\dagger$	$[-10, 20]^\dagger$	$[-10, 20]^\dagger$

†: The SNR levels that are used are all 5 dB apart for a total of seven levels.

5. Add the generated signal to the combined signal array, which is a complex-valued array initialized to N_h samples of additive white Gaussian noise (AWGN) with unity power.
6. Repeat steps 1 to 5 between 1 and 10 times.

The number of signals that are generated for each sample is randomly, and uniformly, selected to fall between one (inclusive) and ten (exclusive) and the three signal types are all equally likely to be selected for inclusion. The signal parameters and their possible values and distributions are shown in Table 3.1.

3.2.3 Time-Frequency Features and Representations

In the past, only the magnitude of the STFT and the spectrogram have been considered for object detection-based signal separation [6, 8–11]. In most cases, the spectrogram has been represented by an RGB image, which is obtained by projecting the values of the spectrogram onto a colour map. In other cases, the spectrogram has been replicated over three channels, to make it possible to use with pipelines that support RGB images [56]. No work has been done to investigate how these features affect signal separation performance, or which is the best.

In this work, nine different input features based on the STFT are considered. Specifically, the features are based on different representations of the complex-valued STFT. The features can all be represented as three-dimensional tensors[1] of dimensions $C \times 512 \times 512$, where C is the number of input channels. The number of input channels varies for the different representations, while 512×512 is the chosen spatial extent of the STFT or spectrogram. The features that are considered are listed below.

RGB spectrogram: A three-channel visual representation of the spectrogram is generated by first calculating (2.41), then applying the base-10 logarithm to improve the dynamic range, before scaling the values to the range $[0,1]$ and projecting it onto a colour map. The *viridis* colour map is used as it is perceptually uniform [62], and it is shown in Figure 3.5. A colour map is perceptually uniform if its derivative in perceptual space, with respect to the data, is a constant value. Naturally, this feature has three input channels: red, green and blue. Examples of the RGB spectrograms are provided in Figure 3.6.

$|\mathbf{STFT}|$**:** The magnitude of the STFT is a simple feature that can be represented by a single input channel[2]. The value of this feature will be higher in regions where a signal is present than in regions only containing noise.

$\angle(\mathbf{STFT})$**:** The argument[3] of the STFT is a feature that is often ignored, and which may or may not contain enough information for the object detection methods to detect and localise the signals. This feature only requires a single input channel.

$|\mathbf{STFT}|, \angle(\mathbf{STFT})$**:** Combining the phase of the complex-valued STFT with its magnitude potentially adds more information which the different object detection algorithms could exploit to detect and differentiate between different signals. This feature is represented by two input channels.

$\Re\{\mathbf{STFT}\}, \Im\{\mathbf{STFT}\}$**:** The real and imaginary components of the STFT give enough information to fully reconstruct a signal and hence the information can also provide the necessary features to discern between different signals. This feature also requires two input channels.

$|\mathbf{STFT}|^2$**:** The spectrogram is commonly used to visualise signals in the time-frequency domain. Only one input channel is required to represent this feature.

$|\mathbf{STFT}|^2, \angle(\mathbf{STFT})$**:** Similar to how the argument of the STFT was added to its magnitude to create a feature, here its added to the spectrogram to create a feature with two input channels.

$10\log\left(|\mathbf{STFT}|^2\right)$**:** The decibel scale[4] is often used in signal processing to improve the dynamic

[1] Tensors are explained in Addendum A

[2] Input channels are explained in Addendum A.

[3] Note $\angle(\cdot) = \arctan2(\Im\{\cdot\}/\Re\{\cdot\})$.

[4] Note $\log(\cdot) = \log_{10}(\cdot)$.

Figure 3.5. A perceptually uniform colourmap, *viridis*, is included in the matplotlib library for Python.

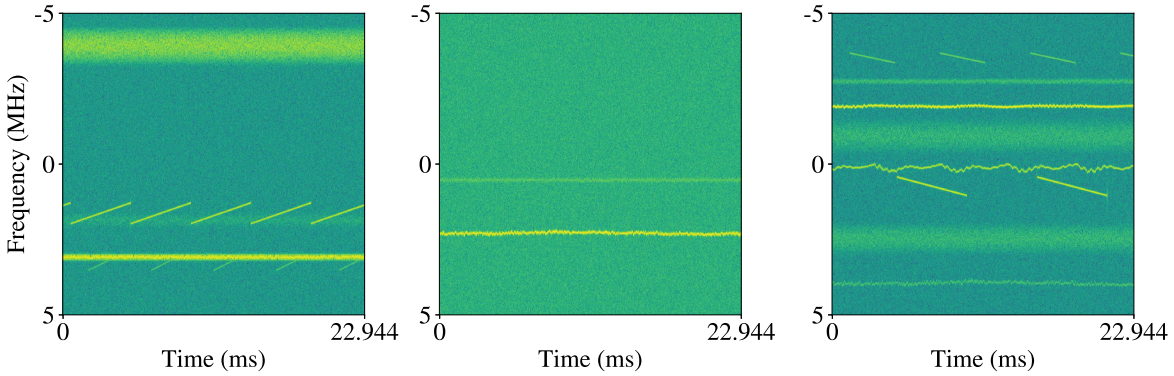


Figure 3.6. The individual signals are often clearly visible upon inspection of the spectrograms of the composite signals. However, some overlapping and low-SNR signals may be difficult for humans to discern.

range of a feature, and here it is applied to the spectrogram for the same reason. This feature also only requires a single input channel.

$10\log\left(|\mathbf{STFT}|^2\right), \angle(\mathbf{STFT})$ **:** As with two of the other features, the argument of the STFT is added as a feature to give a better representation of the data in the complex plane. This feature requires two input channels.

The sampling frequency F_S was chosen as 10 MSPS (or MHz), as this is well within the capabilities of the majority of low-cost software-defined radios (SDRs), and this was fixed for all simulations. In order for the generated features to be of a reasonable size (that is similar to natural images, and small enough to fit into memory when passed through a deep CNN), the decision was made to collect enough samples to create square features that are 512 values wide and 512 values high. This means that each fast Fourier transform (FFT) which is calculated contains 512 samples (this is a power of two, which speeds up calculation) and 512 such FFTs are calculated over time. It was decided to use chunks of the signal that overlap slightly to reduce artifacts at boundaries, and thus an overlap of 64 samples was chosen.

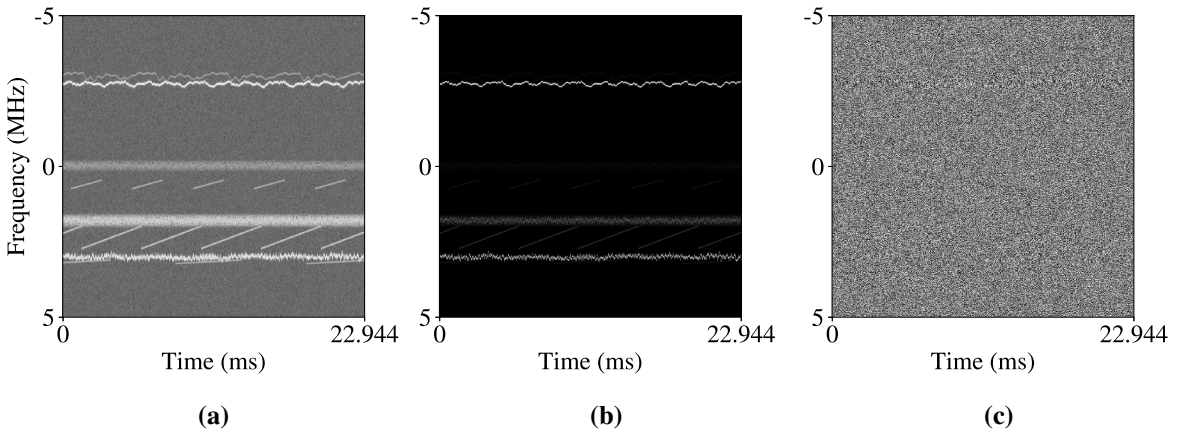


Figure 3.7. The images show three different time-frequency representations, (a) $10\log\left(|\mathbf{STFT}|^2\right)$, (b) $|\mathbf{STFT}|^2$, and (c) $\angle(\mathbf{STFT})$, of the same signal, each mapped to a black and white colour map.

An example of the RGB spectrograms are shown in Figure 3.6. To the human eye, signals appear as yellow regions in these images. It is however important to remember that computers cannot "see" colours, and are unaware of the concept that we know as "yellow", and hence a neural network will look for patterns in the red, green, and blue channels in a way that is very different to humans.

More feature representations are shown in Figure 3.7. In this case the features are projected onto a black and white colour map to remind the reader that these features only comprise a single value channel. The features in Figures 3.7(a)–(c) are all representative of the same signal. It is very clear that the feature in Figure 3.7(a) has better dynamic range than the others, and this is to be expected as the feature is projected onto the decibel scale. In Figure 3.7(b) many of the weaker signals are either invisible or barely visible, and yet neural networks may still detect the subtle patterns. The feature shown in Figure 3.7(c), the argument of the STFT, appears to contain very little information. The only signal that is partially observable to the human eye in this image, is the FM signal at the top of the image.

3.2.4 Normalisation

The only preprocessing that is done is to normalise the inputs to have approximately zero mean and unit variance. This is done by calculating the per-channel mean and variance for the training dataset, and then using these values both during training and inference. Normalisation can be done according

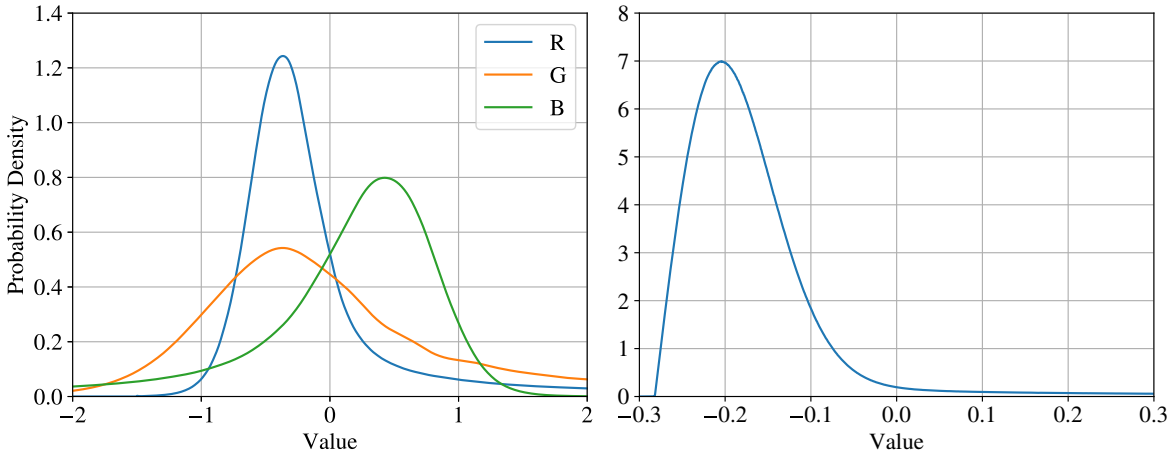
Table 3.2. Mean and standard deviation of different time-frequency representations as calculated for the signal separation training set.

Input Features	**Mean** (μ)	**Std. Dev.** (σ)
RGB spectrogram	$\begin{bmatrix} 0.174 \\ 0.634 \\ 0.505 \end{bmatrix}$	$\begin{bmatrix} 0.105 \\ 0.068 \\ 0.071 \end{bmatrix}$
$\begin{bmatrix} \lvert\mathbf{STFT}\rvert \end{bmatrix}$	$\begin{bmatrix} 0.141 \end{bmatrix}$	$\begin{bmatrix} 0.469 \end{bmatrix}$
$\begin{bmatrix} \angle(\mathbf{STFT}) \end{bmatrix}$	$\begin{bmatrix} 0.0 \end{bmatrix}$	$\begin{bmatrix} 1.814 \end{bmatrix}$
$\begin{bmatrix} \lvert\mathbf{STFT}\rvert \\ \angle(\mathbf{STFT}) \end{bmatrix}$	$\begin{bmatrix} 0.141 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 0.469 \\ 1.814 \end{bmatrix}$
$\begin{bmatrix} \Re\{\mathbf{STFT}\} \\ \Im\{\mathbf{STFT}\} \end{bmatrix}$	$\begin{bmatrix} 0.0 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 0.348 \\ 0.348 \end{bmatrix}$
$\begin{bmatrix} \lvert\mathbf{STFT}\rvert^2 \end{bmatrix}$	$\begin{bmatrix} 0.312 \end{bmatrix}$	$\begin{bmatrix} 2.953 \end{bmatrix}$
$\begin{bmatrix} \lvert\mathbf{STFT}\rvert^2 \\ \angle(\mathbf{STFT}) \end{bmatrix}$	$\begin{bmatrix} 0.312 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 2.953 \\ 1.814 \end{bmatrix}$
$\begin{bmatrix} 10\log\left(\lvert\mathbf{STFT}\rvert^2\right) \end{bmatrix}$	$\begin{bmatrix} -25.52 \end{bmatrix}$	$\begin{bmatrix} 8.55 \end{bmatrix}$
$\begin{bmatrix} 10\log\left(\lvert\mathbf{STFT}\rvert^2\right) \\ \angle(\mathbf{STFT}) \end{bmatrix}$	$\begin{bmatrix} -25.52 \\ 0.0 \end{bmatrix}$	$\begin{bmatrix} 8.55 \\ 1.814 \end{bmatrix}$

to

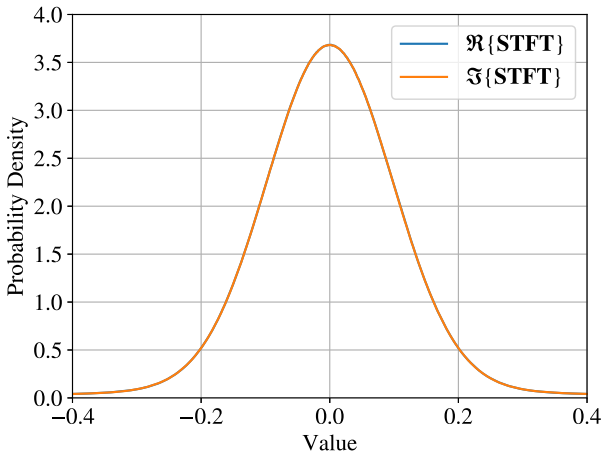
$$\hat{x} = \frac{x - \mu}{\sigma}, \tag{3.10}$$

where μ is the population mean and σ the population standard deviation. The values of $\hat{x}$ will have approximately zero mean and unity variance, but will still be distributed according to the same family of distributions as x. The values of μ and σ for the different feature representations are given in Table 3.2.

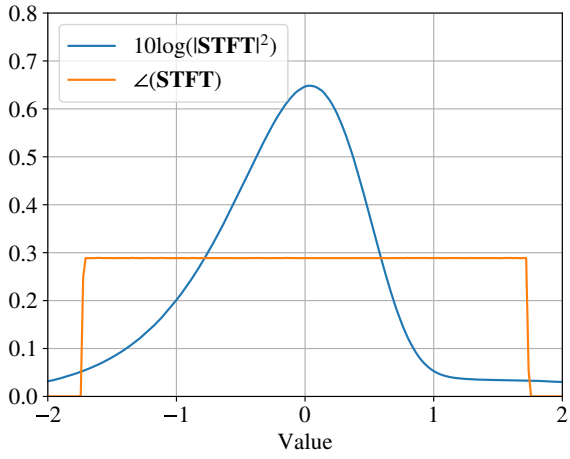


(a) RGB spectrogram

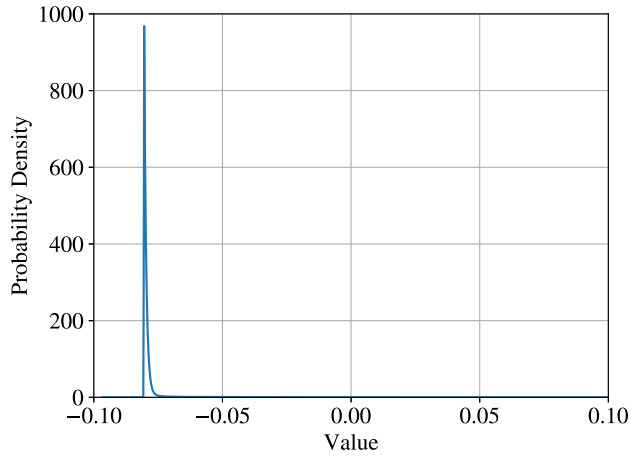
(b) $|\mathbf{STFT}|$



(c) $\Re\{\mathbf{STFT}\}, \Im\{\mathbf{STFT}\}$



(d) $10\log\left(|\mathbf{STFT}|^2\right), \angle(\mathbf{STFT})$



(e) $|\mathbf{STFT}|^2$

Figure 3.8. The probability density functions (PDFs) for the different time-frequency representations of the signal separation dataset highlight their differences.

Features that are normalised to have zero mean and unit variance do not necessarily follow a normal distribution. This is very clear from Figure 3.8. Although most of the features loosely represent a Gaussian curve, there are two that stand out: $\angle(\mathbf{STFT})$ and $|\mathbf{STFT}|^2$.

The argument of the STFT represents, almost perfectly, a uniform distribution. This may point to the fact that values are randomly distributed and do not follow any pattern. It is known that the uniform distribution maximises entropy [63], and since least information is expressed as maximum entropy, it would make sense that this feature does not convey much information. This hypothesis is proved to be correct in Chapter 4.

The squared magnitude of the STFT is the second feature that is in no way representative of a normal distribution. This feature seemingly has a very long tail, which means that although most values are centred around a point to the left of zero, a small number of really large values influence the mean enough to move it to zero. In a case such as this, the standard deviation is not very informative as it gives no real indication of how spread out the average values are.

These two examples clearly highlight the issue of merely relying on the mean and standard deviation to normalise a feature. However, to keep the scope of this work reasonable, and while adhering to the best practices of training neural networks [64], all features are normalised according to (3.10).

3.2.5 Annotations

In order to train and test the object detection algorithms, ground-truth bounding box annotations have to be generated for each signal. These annotations are saved in XML files, and follow the same format as the PASCAL VOC annotations [29], which makes it easy to integrate with existing object detection pipelines.

FM and GMSK signals are continuous and therefore each signal is associated with a single bounding box which spans the entire width of the signal. By contrast, several bounding boxes are generated for each LFM signal. These bounding boxes correspond to the LFM pulses, and for LFM pulses that increase in frequency, they start when the frequency is at a minimum and end when the pulse reaches its maximum frequency. For decreasing LFM pulses, the opposite is true.

Table 3.3. Summary of signal separation dataset.

	training		test	
	signals	**annotations**	**signals**	**annotations**
FM	2 356	5 052	2 332	4 987
GMSK	2 375	5 071	2 370	5 039
LFM	2 337	20 138	2 339	20 140
Total	3 000	30 261	3 000	30 166

The way signals are annotated in the time domain (x-axis) is described above, but the ground-truth bounding box annotations also need to describe the frequency extents (*i.e.* bandwidths) of the signals. This is done by storing the approximate minimum and maximum frequency of each signal.

FM signals have a minimum frequency of $f_c - f_\Delta$ and a maximum frequency of $f_c + f_\Delta$, since the modulated message is normalised to the range $[-1; 1]$. It was decided to use the 99% bandwidth for GMSK signals, which means that the frequency ranges from approximately $f_c - 1.2/T$ to $f_c + 1.2/T$ [65]. Alternatively the null-to-null bandwidth or a different percentage bandwidth could have been used. Determining the frequency extent for LFM signals is trivial as the frequency ranges from $f_c - BW/2$ to $f_c + BW/2$.

As is good practice [14], both a training and a test set are generated, each containing 3 000 spectrograms. The total number of composite signals containing each signal type, as well as the total number of signal annotations is shown in Table 3.3.

3.3 OBJECT DETECTION TRAINING AND CONFIGURATION

The training procedure arguably constitutes the most important part of state-of-the-art deep neural networks. This includes how and which training samples are used and which optimisation algorithm is used to tune the network weights. Other hyperparameters and settings that contribute towards training are also described in this section.

3.3.1 Transfer Learning

Transfer learning is usually applied to problems where the source and target problems use the same input features, *e.g.*, natural RGB images. In this work, there is a requirement to transfer pretrained network weights that were trained on input features with three channels to input features with one or two channels. There are no examples in the current body of literature that indicate that this has been done before.

This leaves two options for performing transfer learning. The first is to simply ignore the weights corresponding to the third (and possibly second) input channel when it will not be used, and the second is to add a single untrained layer with 1×1 convolutions at the start of the network. The second solution will have the effect of projecting the input feature from one or two dimensions to three.

In this work, the first solution is used, because of its simplicity. The results show that this choice does improve the accuracy of the Faster R-CNN framework, which indicates that this solution has merit. A thorough study will have to be conducted to determine whether the alternative method improves accuracy, and whether there are other potential solutions to this problem.

3.3.2 Anchor Generation

Faster R-CNN (or more specifically, the RPN) and YOLO (since version 2) both use anchor boxes[5] [20, 23, 41]. The methods that are proposed for generating anchors in the original Faster R-CNN and YOLOv2 papers are very different.

Faster R-CNN uses three sizes corresponding to box areas of 128^2, 256^2, and 512^2 pixels, and three different aspect ratios ($1:1$, $1:2$, and $2:1$) to generate a total of nine anchors [20]. An ablation study was conducted to determine how the choice of anchors affects detection performance, and using fewer scales or aspect ratios, has a negative impact on accuracy [20].

[5] Anchor boxes are explained in Addendum B

YOLOv2 proposes the use of the k-means algorithm to determine good anchors [41]. The distance metric that is used for the k-means algorithm is

$$d(\text{box, centroid}) = 1 - \text{IoU(box, centroid)}. \tag{3.11}$$

The reason for using this metric, and not the usual Euclidean distance, is that larger boxes should not necessarily generate larger errors than smaller boxes [41].

In Figure 3.9, the normalised sizes for all ground truth bounding boxes from the signal separation training dataset is shown. It is clear from this figure that the signals of interest are representative of a variety of different scales. The aspect ratio of most of the signals is such that they are much wider than they are tall, something which is not necessarily true when considering natural images for object detection.

In this work, the k-means algorithm originally proposed in the YOLOv2 paper is used to generate all anchors for both Faster R-CNN and YOLOv2. This means that anchors for the signal separation problem are more representative of the data than they would have been if the original choices from the PASCAL VOC implementations were used. The same number of anchors is used as in the original implementations, *i.e.* nine for Faster R-CNN and five for YOLOv2.

The calculated anchors are also shown in Figure 3.9. On the right hand edge of this figure, we can see a cluster of blue dots, which represents all FM and GMSK signals. These signals all cover the entire width (time dimension) of the time-frequency representations that are generated. Two anchors for both RPN and for YOLOv2 can also be seen towards the right hand side of the graph, and these anchors are expected to be responsible for these signals.

The rest of the anchors – seven for RPN and three for YOLOv2 – are seen further to the left of the graph, and they are all expected to represent LFM signals. The main intuition behind having this many anchors for one signal type, is that the LFM pulses come in many different shapes and sizes. There are also more LFM pulses than there are FM and GMSK signals. It is unknown whether this biases the networks to be more accurate for LFM signals, and an ablation study will need to be conducted to determine the effects of this.

Each ground truth bounding box has a single anchor associated with it, which maximises the IoU between the ground truth and any of the anchors. The distribution of IoUs between ground truth

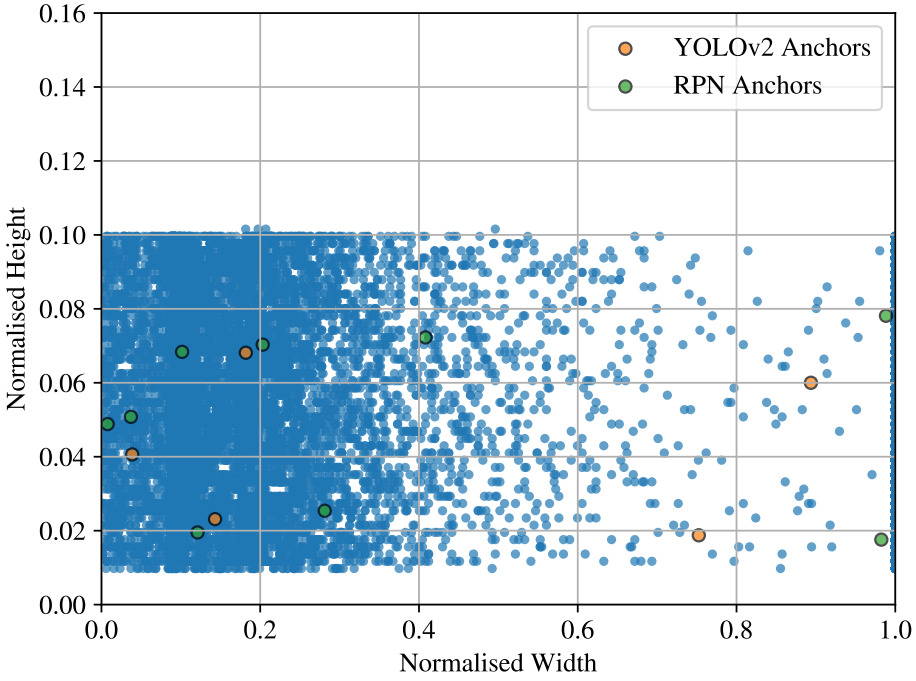


Figure 3.9. The RPN from Faster R-CNN and YOLOv2 both utilise anchors which should be as similar as possible to the objects (or in this case signals) that they represent. Here, nine anchors for RPN and five for YOLOv2 are overlaid on top of all the signals that are present in the signal separation train dataset. The anchors and the signals are normalised by the height and the width of the feature vectors that are used to represent them, *i.e.* 512×512.

bounding boxes and their associated anchors is shown in Figures 3.10 and 3.11. This shows that there are no corresponding anchors that have an IoU of more than 0.5 for almost 40% of all the signals in the training set. This can be improved by using more anchors, or possibly by using a weighted k-means algorithm to calculate the anchors. For Faster R-CNN, which uses more anchors, only 15% of all signals are matched to anchors with an IoU of less than 0.5.

3.3.3 Faster R-CNN

In this work, a ResNet-101 backbone is used for Faster R-CNN, as it has been shown to be more accurate than the original VGG-16 backbone [22]. Another deviation from the original Faster R-CNN implementation is that RoI align is used instead of RoI pool, as it also improves accuracy [43].

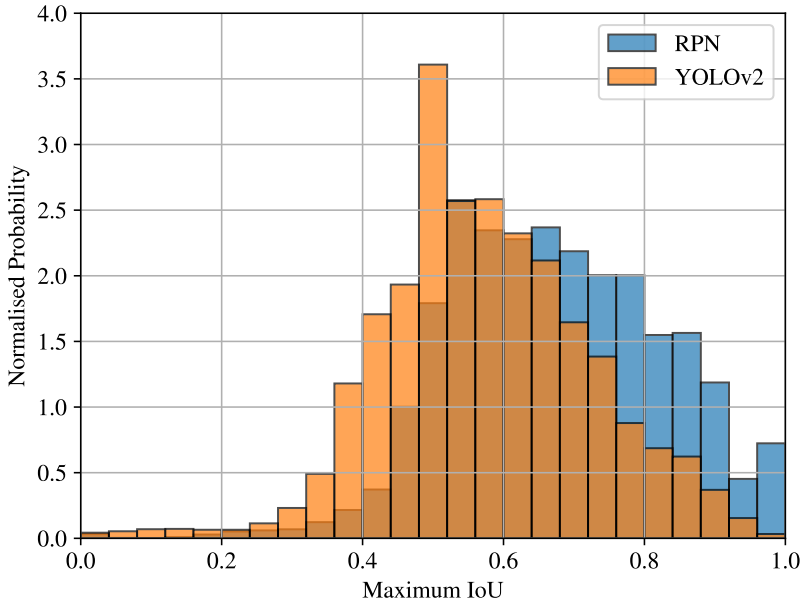


Figure 3.10. The PDF of the maximum IoU of all the signals in the signal-generation training set with any of the calculated anchors for both RPN and YOLOv2.

The training strategy is similar to the strategy proposed in the original Faster R-CNN paper, but as advised in later work [22], the network is trained end-to-end using the approximate joint method. Training was performed on the training dataset for 18 full epochs with a batch size of 1, for a total of 54 000 iterations. If all RoIs are considered during training, then background RoIs will dominate the loss and hence RoIs are sampled. This means that during training only 256 RoIs count towards the RPN loss and only 64 predictions are sampled for the detection head. Stochastic gradient descent (SGD) with momentum and a learning rate of 10^{-3} is used. The learning rate is decayed by a factor of 10 after 36 000 iterations and again after 48 000 iterations.

The backbone network that was used for transfer learning was pre-trained on the ImageNet dataset. This network is available as part of the torchvision library for PyTorch [66]. The common practice of freezing the first few weight layers is followed [22]. The corresponding batch normalisation statistics are also frozen.

The network did not converge under these settings when considering the input feature that combined the spectrogram with the phase of the STFT. This was remedied by unfreezing all the layers. A plausible reason for the failure of the network to converge for this feature is that the range and shape of

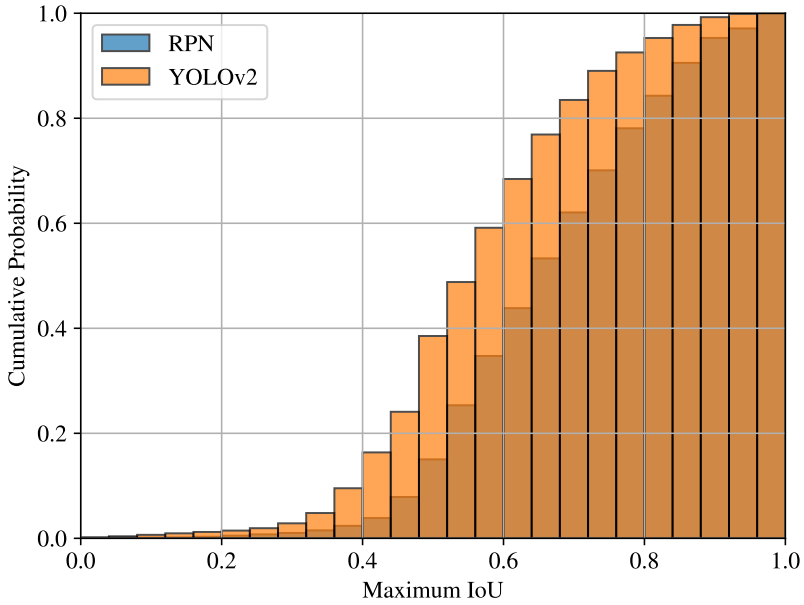


Figure 3.11. The cumulative distribution function (CDF) of the maximum IoU of all the signals in the signal-generation training set with any of calculated anchors for both RPN and YOLOv2.

the distribution is very different for the two feature channels; the value representing the phase of the STFT is often 10 times larger than the value representing the magnitude of the spectrogram. This can be seen in Figures 3.8(d) and (e).

All models were trained on the Centre for High Performance Computing (CHPC)'s Lengau compute cluster. Eight CPU cores and a single Nvidia V100 GPU were used for each training run, and training a single model took approximately 2 hours.

3.3.4 YOLOv2

The training strategy that is proposed for YOLOv2 in the original publication is used here as is with only a few minor changes. A mini-batch size of 32 is used while training on the training dataset for 120 epochs, which amounts to approximately 11 000 iterations. The learning rate is started at 10^{-2}, and is multiplied by 10 after 100 iterations and then divided by 10 after 8 000 and 10 000 iterations. The optimisation algorithm that is used is SGD with momentum.

The loss function uses the same weightings as originally proposed in the YOLOv2 paper [40]. These are $\lambda_{coord} = 5$, $\lambda_{obj} = 1$ and $\lambda_{noobj} = 0.5$.

The backbone for the YOLOv2 network is Darknet-53. The official implementation of YOLOv2 uses a pretrained version of Darknet-53, which was trained on a 448×448 variant of the ImageNet dataset. Experiments on the signal separation dataset showed that performing transfer learning led to worse results, and as a consequence, no transfer learning was used to obtain the results discussed in Chapter 4.

The CHPC's Lengau cluster was again used for training, and as with Faster R-CNN, eight CPU cores and a single Nvidia V100 GPU were used. To train a single YOLOv2 model on the signal separation dataset took approximately an hour.

3.4 SUMMARY

This chapter provided a thorough description of the data that were generated in order to perform the experiments for which the results are reported in the following chapter. This included a description of the different signal types that are included, and how they were combined and annotated to form a dataset that can be used to train object detection algorithms to solve the signal separation task.

The training procedures for both Faster R-CNN and YOLOv2 were highlighted by describing the steps and the parameters that were used to train these algorithms for the signal separation problem. The procedure that was outlined here will allow other researchers who would like to reproduce this work to do so.

In Chapter 4 the results for different experiments based on the signal separation problem are reported and discussed.

Chapter 4 RESULTS

4.1 INTRODUCTORY REMARKS

In this section, results are presented for nine different time-frequency representations based on the STFT, including the spectrogram. Results are also reported and compared for both Faster R-CNN and YOLOv2. These results can be used to answer the research questions that were outlined in Chapter 1.

4.2 EMPIRICAL RESULTS

In all experiments below where the input features are not explicitly stated, the RGB spectrogram was used. This was done because this feature generally performed well and because it represents the original object detection problem with natural images the most accurately.

4.2.1 Training Loss

The loss functions for both Faster R-CNN and YOLOv2, as trained with the RGB spectrogram as input, are shown in Figure 4.1. These figures provide some information about the training process. Firstly, they shows clearly that both methods were able to converge, as the loss decreases less and less as training progresses. The results also show that neither of the methods were suffering from overfitting, as the loss for the validation set is approximately equal to the loss on the training set.

Figures 4.1(a) and (c) also show the effect of the learning rate scheduler that was used. For Faster R-CNN, the learning rate was first reduced by a factor of 10 after 36 000 iterations and again after

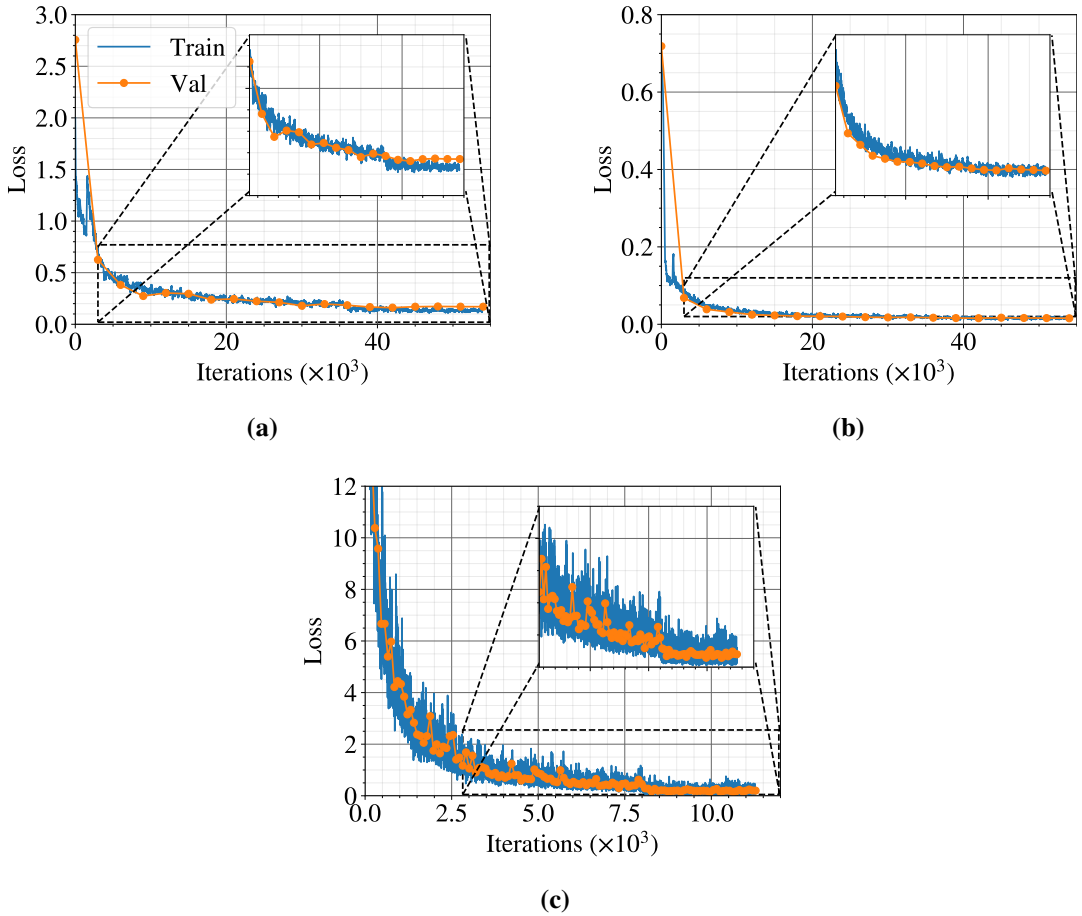


Figure 4.1. The plots of the loss functions during training for both Faster R-CNN [(a) L_{RPN} and (b) L_{DET}], and (c) YOLOv2 show that the models have converged successfully.

48 000 iterations. The drop in the loss function is very clear in the first instance, but less so in the second case. Similarly for YOLOv2, the learning rate was first reduced after 8 000 iterations, resulting in a sharp decrease in the loss function soon thereafter. The learning rate is again decreased after 10 000 iterations, but no effect is seen in the plot of the loss function.

The loss functions for other features and representations are not included, as they are visually very similar and add no further information to the plots that are already included.

4.2.2 Non-Maximum Suppression

NMS will discard all predictions from the same class that overlap one another by more than the NMS threshold. A lower NMS threshold will thus result in more predictions being discarded, while in contrast, an NMS threshold of 1.0 will result in no detections being discarded.

In both the Faster R-CNN and YOLOv2 papers, no mention is made as to how the NMS threshold was determined, but it is safe to assume that a grid search was performed using the validation set, and the value that maximised the mAP was then subsequently used. This is also the approach that is used in this work.

Figure 4.2 shows clearly that the NMS threshold affects the accuracy of the object detection methods that were implemented. In the case where NMS is not used (*i.e.* an NMS threshold of 1.0), the accuracy of Faster R-CNN is significantly worse than that of YOLOv2. The reason for this can be attributed to the fact that YOLOv2 naturally limits the number of detections per cell, while Faster R-CNN produces significantly more predictions. If the inference time has to be reduced as much as possible – as in real-time applications – it could be a good idea to use YOLOv2 without NMS, as it still produces reasonable results, while reducing the inference time.

The optimal NMS threshold for Faster R-CNN is 0.5, where the mAP reaches 89.3%, while for YOLOv2, an NMS threshold of 0.6 maximises the mAP (88.5%). These values (0.5 and 0.6) are used for all the experiments in this work, even though it is noted that other values may produce better results in specific cases, *i.e.* this experiment will have to be conducted for every possible input feature if optimum performance is to be ensured.

In Figure 4.3, examples of NMS applied to the predictions of both Faster R-CNN and YOLOv2 can be seen. Examples of all three signal types being detected multiple times can be seen in Figures 4.3(a) and 4.3(c). No duplicate detections are present in Figures 4.3(b) and 4.3(d), which shows that the NMS algorithm is performing satisfactorily.

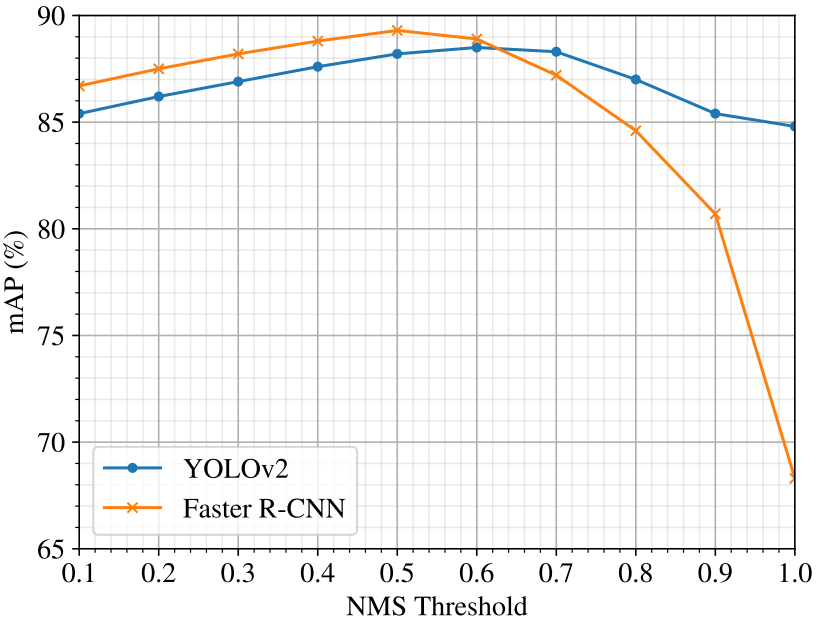


Figure 4.2. The NMS threshold determines by how much two predictions from the same class should overlap before one is discarded. This plot shows the effect that different NMS thresholds have on the accuracy of both Faster R-CNN and YOLOv2, with specific reference to the signal separation dataset.

4.2.3 Transfer Learning

Transfer learning has been shown to improve accuracy on several deep learning tasks. It is shown in Table 4.1 that Faster R-CNN benefits greatly from transfer learning, and it improves mAP by almost 10 points. The experiments show that on the signal separation task, YOLOv2 however does not benefit from transfer learning and the accuracy drops by approximately 5 points when transfer learning is used.

The results shown in Table 4.1 informed the decision to use different strategies for training Faster R-CNN and YOLOv2. In order to maximise the accuracy of both approaches, Faster R-CNN was trained with transfer learning, while YOLOv2 was trained without it. Following on the practice from the ResNet paper, all layers (including batch normalisation[1]) before the fourth block of residual connections (this is commonly known as the `conv4` layer [22]) are frozen when doing transfer learning.

[1] Batch normalisation is explained in Addendum A.

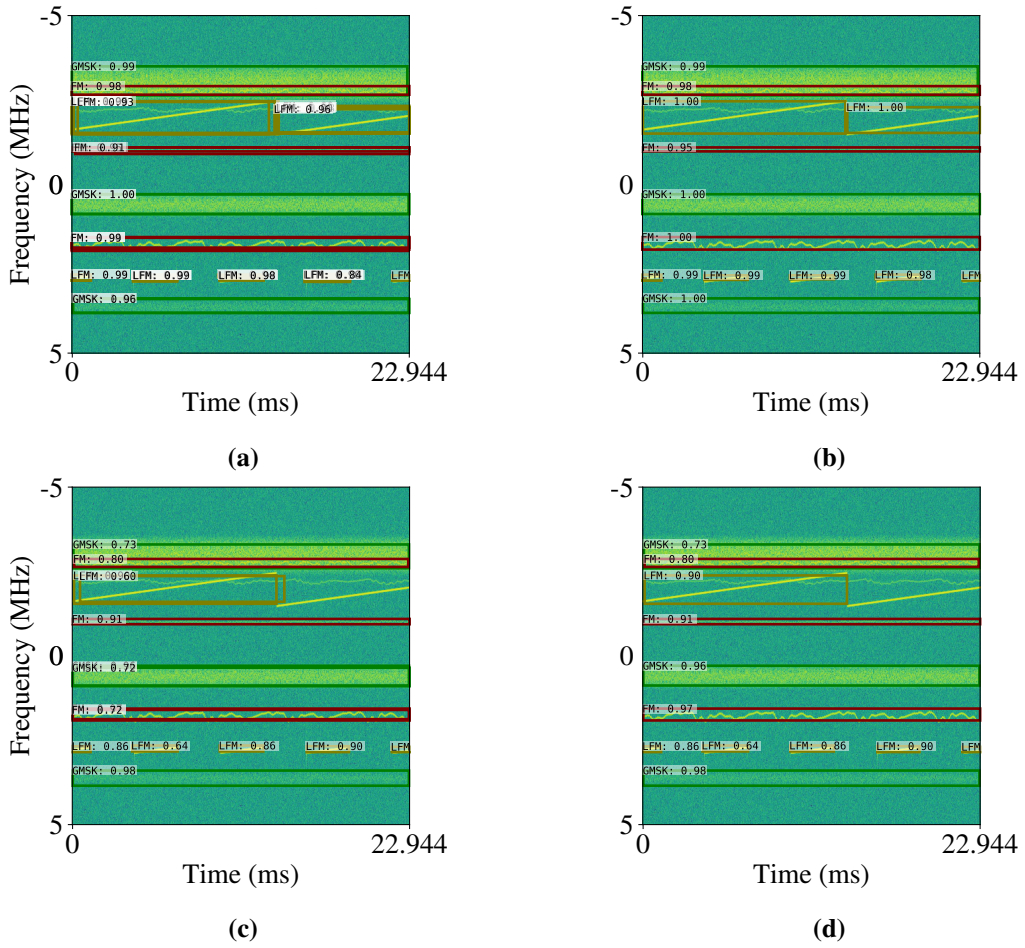


Figure 4.3. These examples show all the detections produced by (a) Faster R-CNN and (c) YOLOv2, and how NMS successfully removes duplicate detections for both (b) Faster R-CNN and (d) YOLOv2.

Kaiming uniform initialisation was used when training networks from scratch, as it has been shown to produce good results [21]. No weights were frozen when training from scratch.

4.2.4 Different Time-Frequency Representations

The different time-frequency representations introduced in Section 3.2.3 are all included in Table 4.2. Here, the accuracy that was achieved by both Faster R-CNN and YOLOv2 can be seen for all of these representations.

The average mAP for features that are log-transformed (this includes the RGB spectrogram) is 88.9%, when considering both Faster R-CNN and YOLOv2. In comparison, the average for features that

Table 4.1. Accuracy (in % mAP) for the different methods when training from randomised weights versus when using transfer learning.

	YOLOv2		**Faster R-CNN**	
	Scratch	***Transfer***	***Scratch***	***Transfer***
RGB spectrogram	**88.5**	83.5	75.1	**89.3**
$\left[10\log\left(\lvert\mathbf{STFT}\rvert^2\right)\right]$	**88.6**	83.2	79.1	**89.3**

are not log-transformed is 85.5%. This indicates that for the task of signal separation, features that are log-transformed perform better than those that are not. The log transformation is often used to make it easier to inspect both very small and very large values simultaneously. Since signals with SNR ranging from -10 dB to 20 dB are considered, it makes sense that log-transformed features will perform better.

The initial hypothesis that complex phase information may improve accuracy is disproven. The phase does not improve the accuracy of any of the features, nor are the object detection algorithms able to learn meaningful representations from the phase as an independent feature.

This can possibly be attributed to the fact that the phases are distributed randomly for certain signal types and that hence, they contain no information. The distribution of the normalised phase is also different than any of the other distributions that were considered. This can be seen in Figure 3.8(d), where the argument of the STFT is distributed uniformly between $[-1.73, 1.73]$. The range of these values is significantly different to some of the other features, especially $|\mathbf{STFT}|$, which makes it very difficult to learn good filters.

The fact that the phase wraps around at π and $-\pi$ might also influence the performance of the phase as an extra feature. Either way, the results show that, on average, adding the complex phase as an extra feature channel reduces the mAP score by 1.4 points. Faster R-CNN and YOLOv2 trained only on $\angle(\mathbf{STFT})$ achieve mAP scores of 47.3% and 28.7% respectively. This is considerably lower than the accuracy for other features.

The results in Table 4.2 show that both Faster R-CNN and YOLOv2 are relatively robust to different

Table 4.2. Accuracy (in % mAP) of the different methods for different input features.

Input Features	YOLOv2	Faster R-CNN
RGB spectrogram	88.5	**89.3**
$\left[\lvert\mathbf{STFT}\rvert\right]$	88.2	89.0
$\left[\angle(\mathbf{STFT})\right]$	47.3	28.7
$\begin{bmatrix}\lvert\mathbf{STFT}\rvert \\ \angle(\mathbf{STFT})\end{bmatrix}$	88.6	87.0
$\begin{bmatrix}\Re\{\mathbf{STFT}\} \\ \Im\{\mathbf{STFT}\}\end{bmatrix}$	87.2	88.2
$\left[\lvert\mathbf{STFT}\rvert^2\right]$	79.0	87.2
$\begin{bmatrix}\lvert\mathbf{STFT}\rvert^2 \\ \angle(\mathbf{STFT})\end{bmatrix}$	73.6	86.2
$\left[10\log\left(\lvert\mathbf{STFT}\rvert^2\right)\right]$	88.6	**89.3**
$\begin{bmatrix}10\log\left(\lvert\mathbf{STFT}\rvert^2\right) \\ \angle(\mathbf{STFT})\end{bmatrix}$	**88.8**	89.0

input features based on the STFT and the related spectrogram. This characteristic is probably boosted by the fact that all input features are normalised to have approximately zero mean and unit variance.

4.2.5 Intercept-over-Union

An IoU of 0.5 with the ground truth is widely accepted as good enough to be deemed a true positive [29], but it is interesting to see how results are affected if this threshold is either increased or decreased. A lower threshold will mean that more predictions are deemed correct, which will increase the reported accuracy, while the opposite also is also true.

In Figure 4.4, results are shown for a range of IoU thresholds. Initially, accuracy only decreases gradually as the threshold is increased from 0.1 to 0.5, but as the threshold approaches 1.0, accuracy

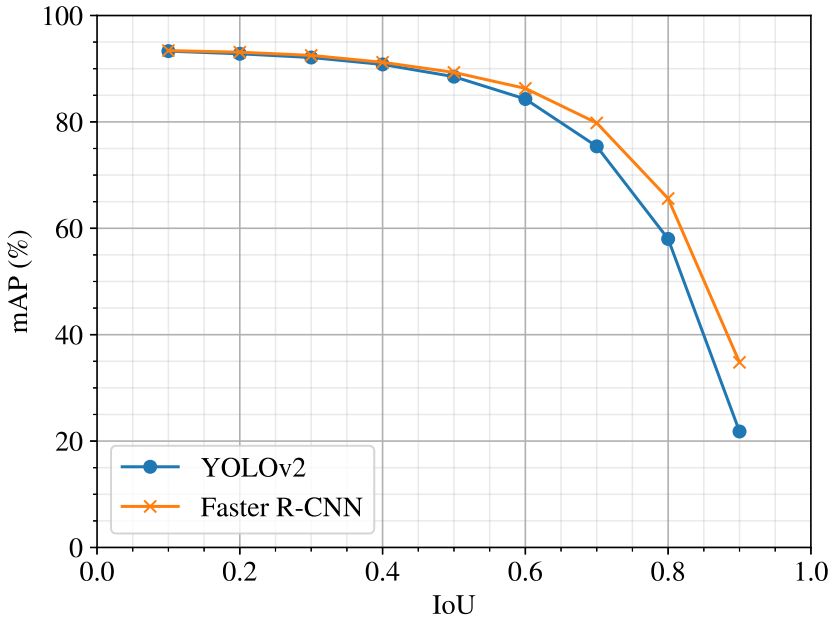


Figure 4.4. The IoU threshold that is used to determine when a prediction is correct has a great impact on the reported accuracy. Faster R-CNN outperforms YOLOv2 by a large margin when a high threshold is used.

decreases more quickly. This indicates that although most predictions are good enough to be considered correct when using a threshold of 0.5, very few predictions are perfect, which would correspond to a high accuracy at a threshold close to 1.0.

At lower thresholds, it is noticeable that Faster R-CNN and YOLOv2 perform approximately equally well, but as the threshold is increased, Faster R-CNN starts to outperform YOLOv2 significantly. This indicates that although the detection performance of both methods is similar, Faster R-CNN is able to localise signals more accurately than YOLOv2.

4.2.6 Signal-to-Noise Ratio

Tables 4.3–4.6 show that both methods perform well, even when the SNR is below 0 dB. Results are calculated by only considering bounding box annotations corresponding to the applicable SNRs. In addition, all detections that overlap with signals where the SNR is not equal to the value for which the results are currently being calculated, are discarded. This means that all false positives that do not overlap with any signals are considered when calculating the precision for each SNR. The results

Table 4.3. Accuracy (in % mAP) of Faster R-CNN for the signal separation dataset at different SNRs with RGB spectrogram as input.

	SNR				
Class	-10 dB	-5 dB	0 dB	20 dB	Overall
FM	74.7	82.1	85.3	90.7	88.2
GMSK	68.8	80.1	84.0	94.8	89.0
LFM	80.3	83.7	86.8	92.5	90.7
Mean	**74.6**	**82.0**	**85.4**	**92.7**	**89.3**

at each SNR are thus considered a lower bound, and this is also the reason why the overall mAP is greater than the arithmetic mean of the results across different SNRs. PASCAL VOC uses a similar approach to calculate the accuracy on objects that are deemed difficult to detect.

Signals with a higher SNR are expected to be easier to detect and localise than signals with a lower SNR. This is confirmed by the results in Tables 4.3–4.6. In most cases the accuracy – for both Faster R-CNN and YOLOv2 – is monotonic with respect to the SNR for both of the feature representations that are considered.

It is evident from Tables 4.3 and 4.5 that the accuracy for the three different signal types that are considered is similar for high SNR, but diverges for low SNR. The accuracy of GMSK suffers especially at low SNR, but this is to be expected, as Faster R-CNN operates without global context. Without context, a low-SNR GMSK signal looks a lot like noise. In contrast, a low-SNR LFM signal is still distinguishable, as its shape in the time-frequency domain makes it stand out from the noise, and hence its accuracy is not degraded as much at low SNR.

The YOLO-based method, for which the results are tabulated in Tables 4.4 and 4.6, performs well over all three signal types, but has the worst accuracy for FM signals. This can potentially be attributed to the fact that the FM signals that were generated have smaller bandwidths than the other signals. Hence, if the predicted bounding box is slightly offset from the true bounding box in the frequency plane, the IoU will decrease significantly. This is also supported by the plot in Figure 4.4, which shows that when the IoU threshold for true positives is relaxed, YOLOv2 matches the accuracy of Faster R-CNN.

Table 4.4. Accuracy (in % mAP) of YOLOv2 for the signal separation dataset at different SNRs with RGB spectrogram as input.

	SNR				
Class	-10 dB	-5 dB	0 dB	20 dB	Overall
FM	72.3	76.4	85.3	81.6	85.2
GMSK	72.9	83.4	84.0	95.3	89.1
LFM	80.5	83.4	86.8	90.5	91.2
Mean	**75.2**	**81.1**	**84.6**	**89.1**	**88.5**

Table 4.5. Accuracy (in % mAP) of Faster R-CNN for the signal separation dataset at different SNRs with the STFT magnitude as input.

	SNR				
Class	-10 dB	-5 dB	0 dB	20 dB	Overall
FM	68.8	78.2	85.7	90.2	86.9
GMSK	69.3	82.0	86.8	94.0	90.1
LFM	78.5	82.4	84.8	91.7	90.0
Mean	**72.2**	**80.9**	**85.8**	**92.0**	**89.0**

4.2.7 Precision-Recall

It is clear from Figure 4.5 that both methods are limited by their ability to positively detect all instances (*i.e.* recall), and that they excel at correctly classifying positive detections (*i.e.* precision). If two signals overlap in time and frequency and they have an IoU of greater than the NMS threshold, then one of the signals will always go undetected since NMS is applied to all bounding boxes. To alleviate this problem and improve recall, NMS can be removed or the NMS threshold can be increased. Soft NMS [67] can also be investigated as an alternative solution.

The YOLO-based method has the further restriction that it can only produce one bounding box per anchor in each cell. Thus, if two signals span approximately the same amount of space in time and

Table 4.6. Accuracy (in % mAP) of YOLOv2 for the signal separation dataset at different SNRs with the STFT magnitude as input.

	SNR				
Class	-10 dB	-5 dB	0 dB	20 dB	Overall
FM	71.1	76.5	81.0	82.3	85.3
GMSK	74.1	84.0	87.3	95.3	88.9
LFM	77.5	80.9	82.7	91.1	90.2
Mean	**74.2**	**80.5**	**83.7**	**89.5**	**88.2**

Table 4.7. Timing information for the different object detection methods on different GPUs.

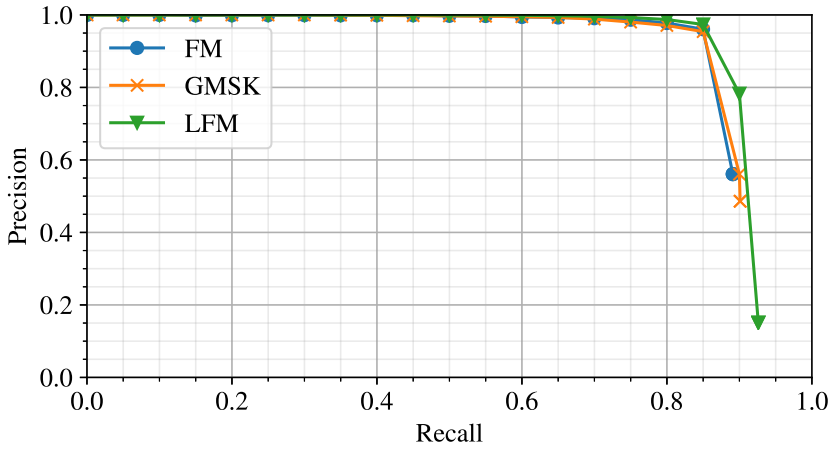
	YOLOv2	**Faster R-CNN**
Nvidia GTX1060Ti	40 ms	115 ms
Nvidia V100	29 ms	37 ms

frequency, *i.e.* they are both centred in the same cell and they most closely match the same anchor, then one of the signals will go undetected. This problem can be remedied by generating more anchors per cell, or by dividing each image into more, smaller cells.

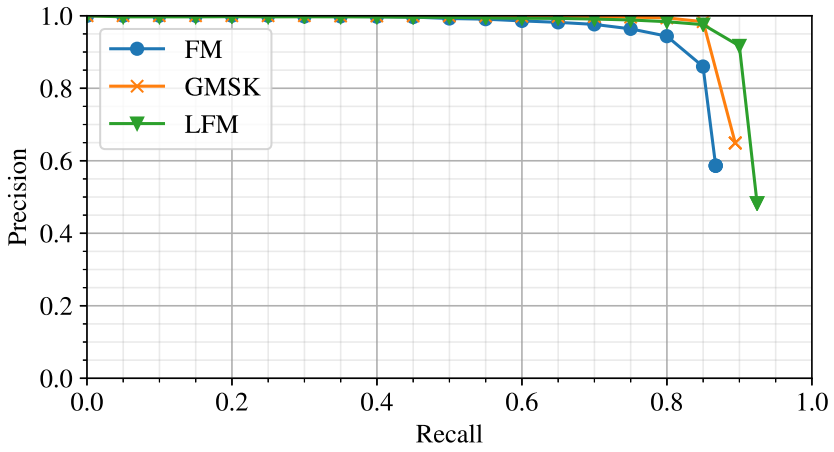
4.2.8 Timing Information

The proposed methods can benefit greatly from the highly efficient, parallel processors found in modern GPUs, as shown in Table 4.7. The speed-up gained from upgrading from the consumer grade Nvidia GTX1060Ti to the enterprise-class Nvidia V100 is greater for Faster R-CNN ($\times$3.11) than for YOLOv2 ($\times$1.38). This can be attributed to the fact that the convolutional backbone, which is highly parallelisable, is larger for Faster R-CNN than for YOLOv2, and the fact that YOLOv2 is faster overall means that a greater proportion of the time is spent on parts that are not parallelisable, *e.g.*, NMS.

The results in Table 4.7 were achieved with a batch size of 1, but by using a larger batch size, *e.g.*, 8,



(a)



(b)

Figure 4.5. The precision-recall graphs for both (a) Faster R-CNN and (b) YOLOv2 indicate that these methods can achieve excellent precision for a wide range of recall values, but both methods fail to achieve perfect recall.

the throughput can be increased at the expense of some latency. This may be acceptable in some cases, but here the worst case is rather assumed.

4.2.9 Examples

Figures 4.6 and 4.7 show examples that were hand-picked to highlight some of the strengths and weaknesses of the considered approaches. The two figures show the same signals, but show the different detections for Faster R-CNN and YOLOv2 respectively.

Figures 4.6(a) and 4.7(a) present a highly complex wideband spectrum with several different signals. Faster R-CNN is able to correctly detect and separate all 16 signals present in this example. YOLOv2 only fails to detect two of the LFM signals in this example, and both of them are highly occluded by a stronger FM signal.

In Figures 4.6(b) and 4.7(b), two relatively low-SNR signals are presented. Faster R-CNN correctly predicts both signals, but also incorrectly detects an LFM pulse. The region inside the predicted bounding box looks like an LFM pulse (the frequency decreases linearly), but by inspecting the entire image, it is clear that this region forms part of an FM signal. This is interesting, because it highlights the fact that Faster R-CNN performs predictions locally without considering global information.

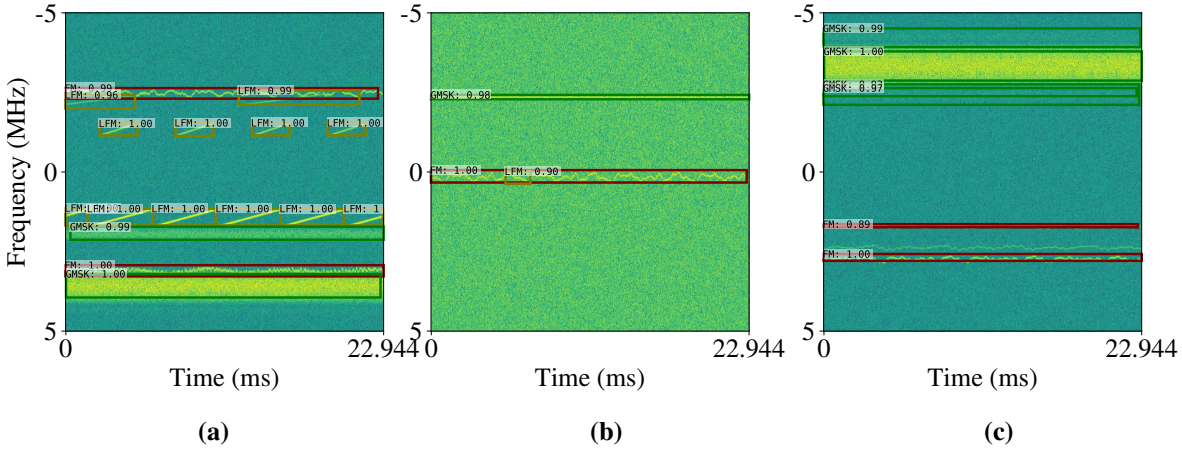
Figure 4.6(c) shows an example where Faster R-CNN confuses the sidelobe of a strong GMSK signal with an actual signal. This can again be contributed to the inability of Faster R-CNN to incorporate global context.

The idea of incorporating global context into the Faster R-CNN framework has been considered, but did not improve the accuracy when applied to natural images [22]. A study will have to be conducted to determine if global context can improve the accuracy of Faster R-CNN, when applied to the signal separation problem.

The detections in Figure 4.7 are generally very good. It is however clear that YOLOv2 is less confident in its predictions than Faster R-CNN. The recall for YOLOv2 is also not perfect, as is evident by the two LFM pulses that go undetected in Figure 4.6(a).

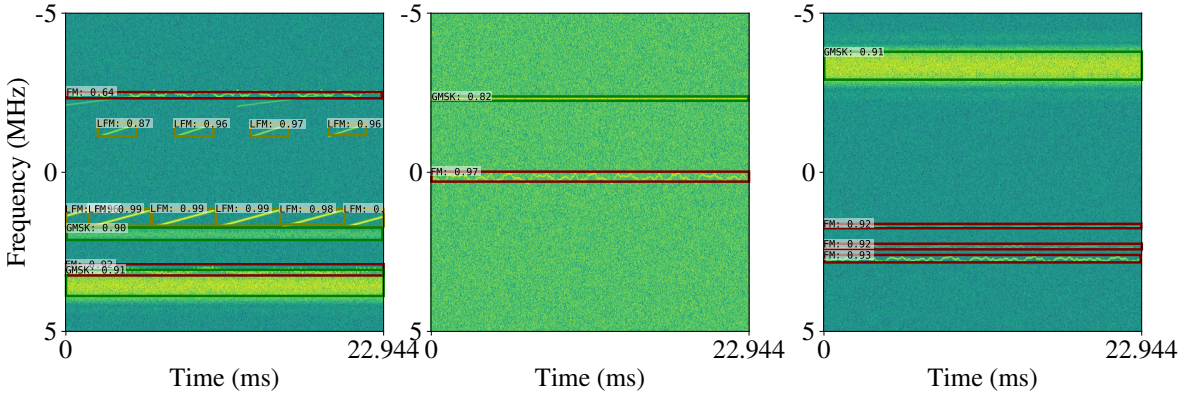
Figures 4.8–4.11 show the detections corresponding to the first four and the last four examples from the test set for Faster R-CNN and YOLOv2 respectively. The RGB spectrogram was considered for Figures 4.8 and 4.10, while the STFT magnitude was considered for Figures 4.9 and 4.11. These examples are included to showcase the average-case detection performance of the respective approaches, as they were not hand-picked.

The examples that use the STFT magnitude are again projected onto a black and white colour map to remind the reader that this feature only requires a single channel. In these examples, it is also apparent that it is difficult to perceive weak signals when a logarithmic scale is not used.



(a) **(b)** **(c)**

Figure 4.6. Example detections for Faster R-CNN showing all detections with confidence greater than 0.9.



(a) **(b)** **(c)**

Figure 4.7. Example detections using YOLOv2 showing all detections with confidence greater than 0.6.

4.3 SUMMARY

The results that are reported in this chapter show that several different time-frequency representations of wideband signals are suitable to be used by object detection algorithms to perform signal separation. The methods that were proposed produce similar results for a range of features, but it is shown that complex phase information is not a suitable feature for the problem and that better results are achieved when the magnitude of certain features are log-transformed, *i.e.* projected onto a logarithmic scale.

The robustness of the methods to the different representations indicate that features other than the spectrogram should be considered for the signal separation task. Also, since the other representations

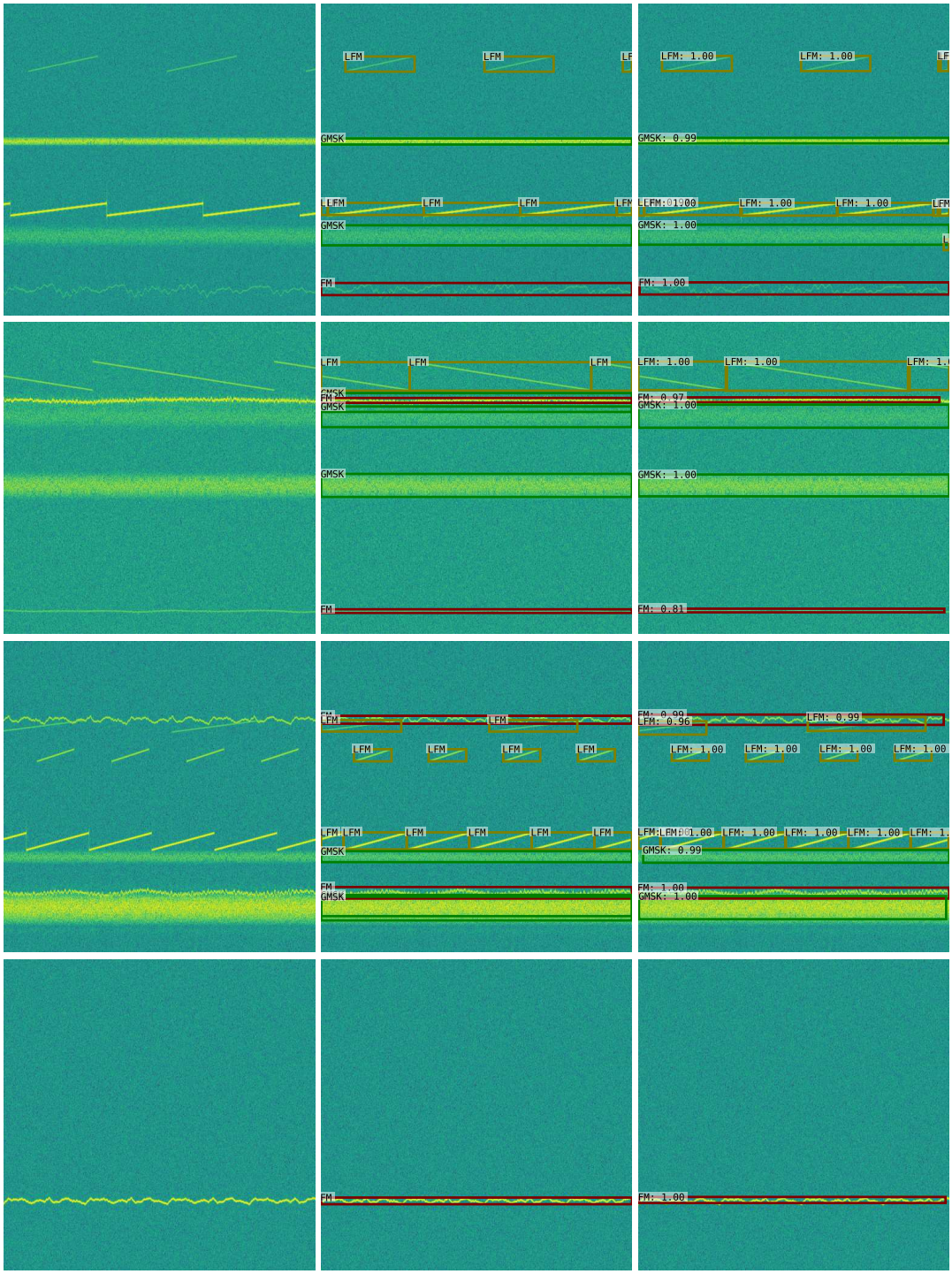


Figure 4.8. Example detections made by Faster R-CNN with the RGB spectrogram as input are shown in the final column. The first column shows no bounding boxes, allowing the reader to inspect the signal without any occlusions. The second column shows the ground truth bounding boxes which are used to evaluate the detections.

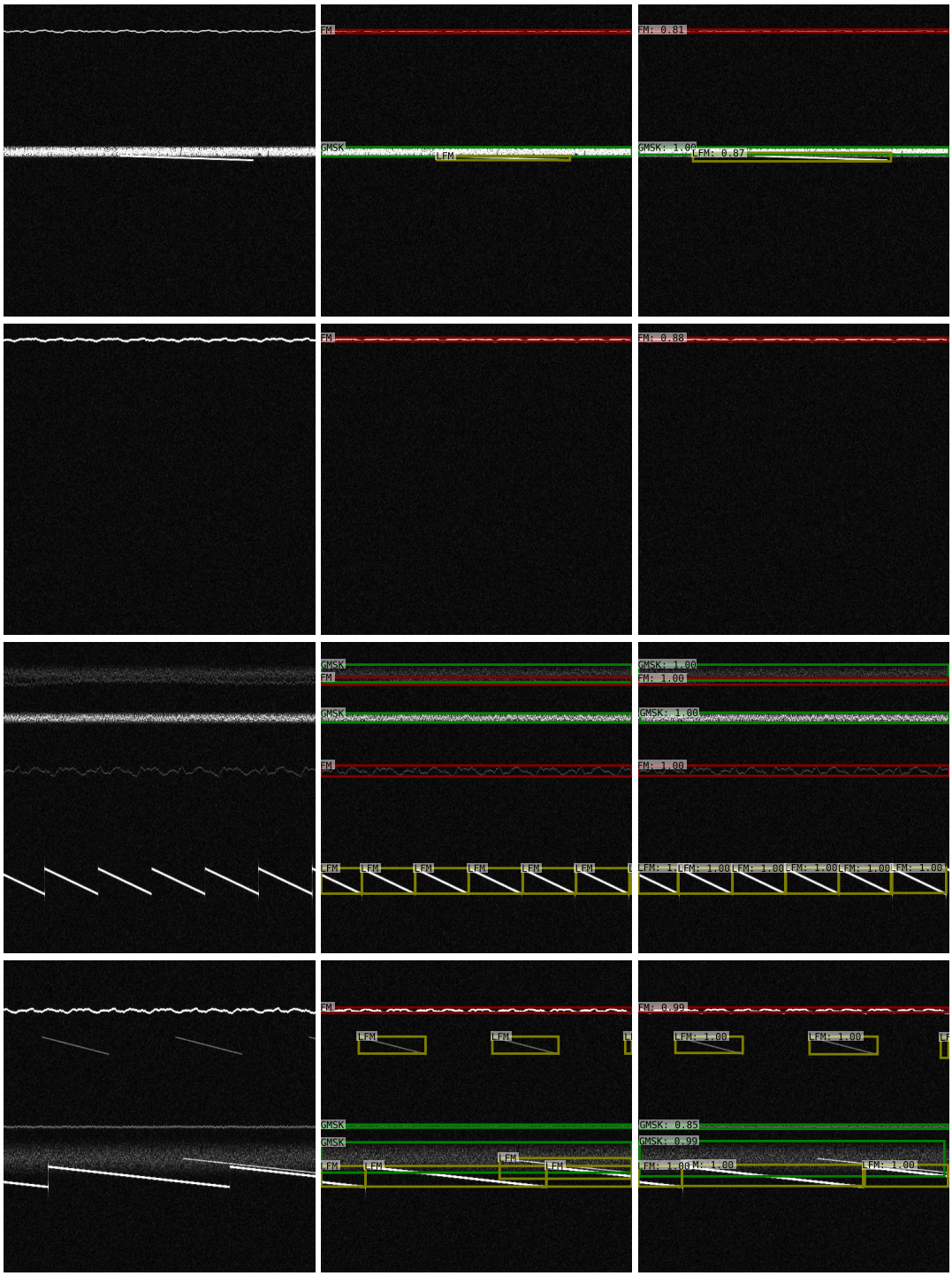


Figure 4.9. Example detections made by Faster R-CNN with the STFT magnitude as input are shown in the final column. The first column shows no bounding boxes, allowing the reader to inspect the signal without any occlusions. The second column shows the ground truth bounding boxes which are used to evaluate the detections. The images in this example are not projected onto an RGB colour map, but rather shown in grayscale, to highlight the fact that the feature that was used here is represented by a single input channel.

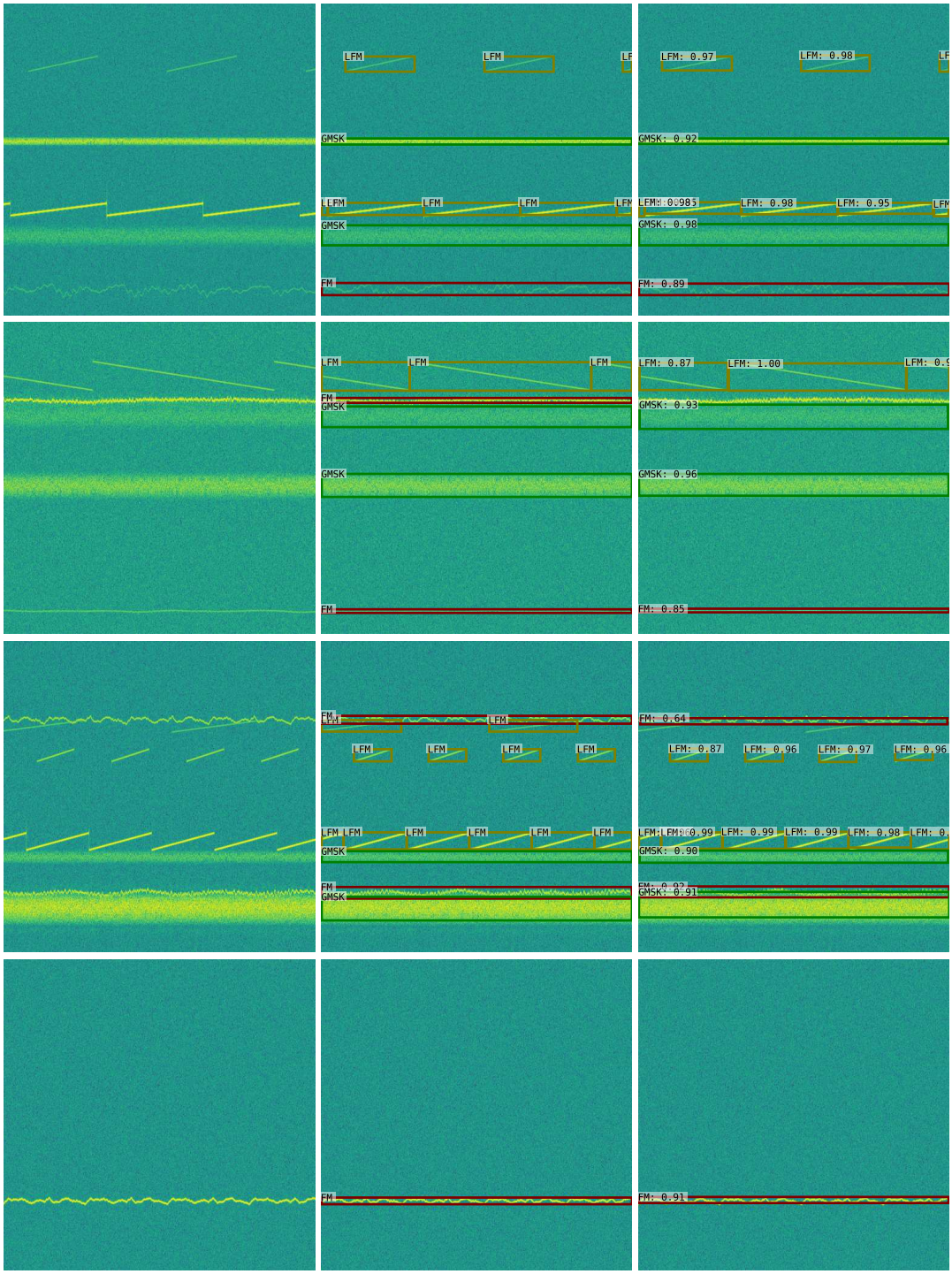


Figure 4.10. Example detections made by YOLOv2 with the RGB spectrogram as input are shown in the final column. The first column shows no bounding boxes, allowing the reader to inspect the signal without any occlusions. The second column shows the ground truth bounding boxes which are used to evaluate the detections.

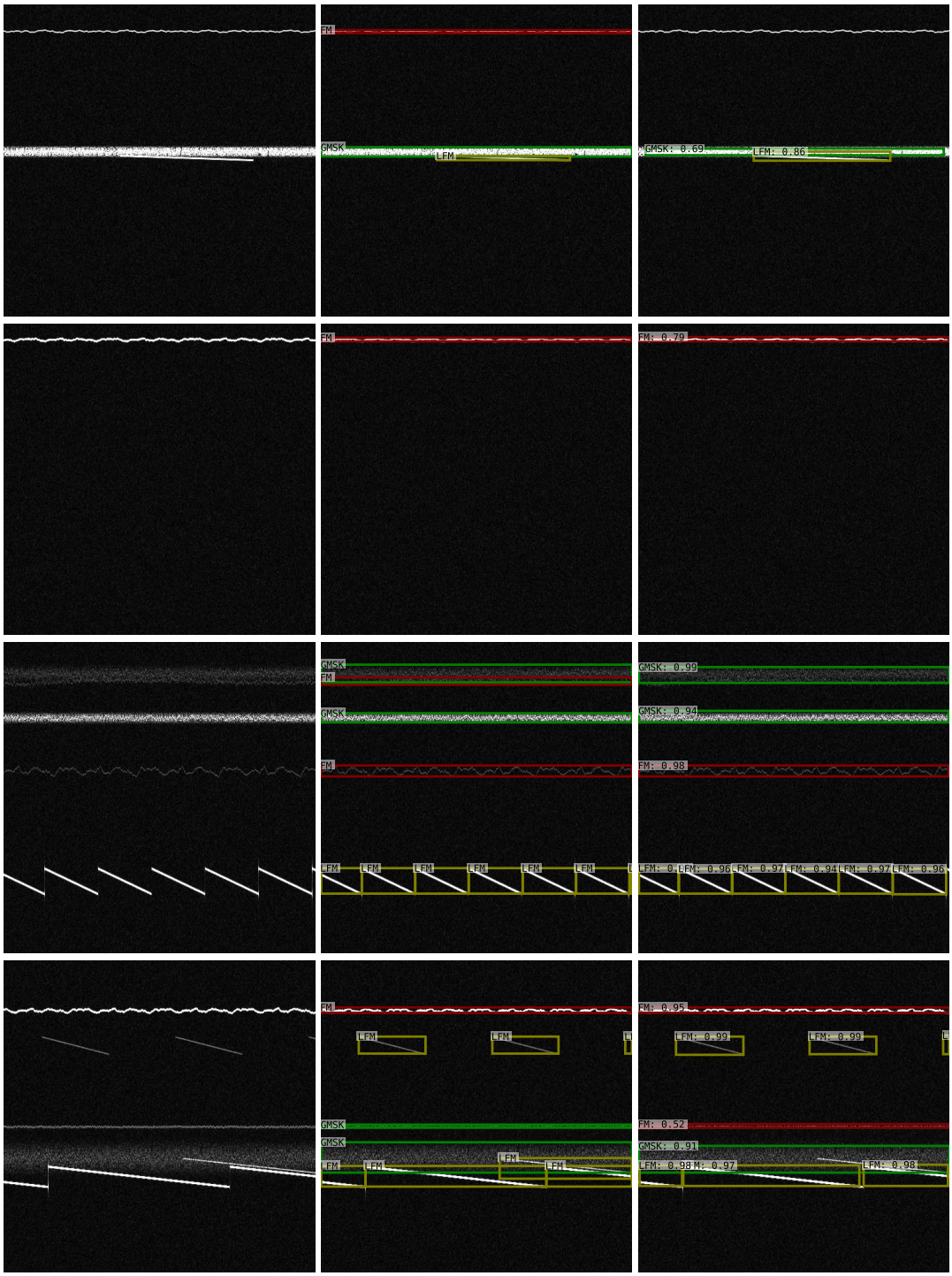


Figure 4.11. Example detections made by YOLOv2 with the STFT magnitude as input are shown in the final column. The first column shows no bounding boxes, allowing the reader to inspect the signal without any occlusions. The second column shows the ground truth bounding boxes which are used to evaluate the detections. The images in this example are not projected onto an RGB colour map, but rather shown in grayscale, to highlight the fact that the feature that was used here is represented by a single input channel.

all require fewer input channels than the RGB spectrogram, the total number of parameters in the input layer to the CNN is reduced. The alternative representations also do not require any image processing, nor that values be mapped to a colour map, which could reduce the inference time of these methods.

Different experiments showed that Faster R-CNN is able to localise signals better than YOLOv2 could, which resulted in higher accuracy scores. YOLOv2 was however shown to be faster and more accurate when NMS was not applied, which allows for an even faster algorithm at the expense of accuracy.

In Chapter 5, a conclusion will be drawn, which ties the results that were reported in this chapter to the research questions that were posed at the start of this work. A list of possible ideas that may inspire future work is also included in the next chapter.

Chapter 5 CONCLUSIONS

5.1 GENERAL COMMENTS

Object detection algorithms, applied to time-frequency representations of wideband signals, can successfully perform signal separation. The results in the previous chapter show that this is true for both Faster R-CNN and YOLOv2. Eight of the nine features that were tested performed well, with the phase of the STFT the only exception.

A thorough understanding of the respective object detection algorithms – Faster R-CNN and YOLOv2 – will allow a researcher or engineer to optimise these methods for specific scenarios. For example, understanding that YOLO struggles with predicting groups of small objects, or how a low NMS threshold will degrade the performance of any object detection method when considering highly occluded signals, will allow for goof design decisions to be made. The findings in the previous chapter aim to provide researcher with the information required to make these decisions.

In terms of the research questions given in Chapter 1, the results are summarised below.

Different time-frequency representations: The object detection algorithms were able to detect, localise and classify all three signal types accurately, using a variety of different time-frequency representations. The phase information from the STFT did not improve detection accuracy, and models trained on only the phase information did not produce meaningful results. The time-frequency representation that led to the highest average accuracy across both methods was the log-transformed spectrogram, which only requires a single input channel, thus also resulting in fewer parameters for each of the networks.

Single-stage vs two-stage object detection: Overall, the Faster R-CNN approach is more accurate than the YOLOv2 approach. The average mAP over all input features is 88.2% for Faster R-CNN, and 85.3% for YOLOv2.[1] This indicates that – as is the case in traditional object detection – two-stage object detection methods are more accurate than single-stage methods.

Overlapping signals: None of the previous works that used object detection for signal detection and localisation included any examples of signals that were overlapping in both time and frequency in their publications. That made the accuracy with which the object detection algorithms could detect overlapping signals in this work all the more impressive. More work should be done to quantify the performance of overlapping signals, but this work has shown that object detection can perform accurate signal separation even when signals are highly occluded.

Signal-to-noise ratio: As hypothesised, the detection performance degraded at lower SNR levels, but even at -10 dB, the detectors were able to successfully localise the majority of the signals. This shows that the proposed machine learning-based methods are able to perform pattern recognition, even in the presence of noise, which leaves the door open to applying these methods in real-world applications, where noise is an omnipresent phenomenon.

5.2 FUTURE WORK

The research that has been described in this work is relatively limited in its scope, as the main goal was to prove that object detection algorithms can be used to detect multiple overlapping signals at different SNR levels, and that it can be done using a variety of time-frequency representations. A list of recommendations for future research is given below.

Novel Object Detection Methods: Since the commencement of this work, several new object detection methods have been proposed. They include EfficientDet [68], DETR [69], YOLOv4 [70] and YOLOv5 [71]. These works are all promising as they all claim state-of-the-art results under some specific conditions, albeit using different metrics. DETR is especially interesting as it introduces a new object detection paradigm that can neither be classified as one-stage or two-stage, but is instead based on recent advances in new neural network architectures known as transformers. Transformers use attention mechanisms which allow neural networks to focus

[1] The only feature that is excluded in this calculation is the phase of the STFT, *i.e.* $\angle(\mathbf{STFT})$, as this result is an outlier.

their calculations on specific regions of the input data, and this could potentially work very well for the signal separation problem discussed in this work.

Complex-Valued Neural Networks: Complex-valued neural networks are currently being actively researched and some neural network frameworks are also planning on supporting them in the future. Since complex values are a more natural way of representing RF signals, neural networks with complex valued weights could potentially be better suited to the problem of signal separation than current approaches.

Automatic Modulation Classification: Systems that can automatically identify the modulation type of a signal have received much attention, and has application in CR amongst others. Most approaches that have been proposed for the automatic modulation classification task, assume that a single signal has been extracted and transposed to baseband. In 2020, the first system that can detect and classify multiple signals in a wideband scenario was proposed [9]. This system is however a cascade of sub-systems that detect the signals of interest, transpose them to baseband and then classify each signal individually. If the object detection methods that were investigated in this work can be adapted to discriminate between several different, potentially similar, modulation types, then it would be the first system to perform automatic modulation classification of multiple signals in a single-shot manner.

Recording Real Signals: Real signals will have to be recorded to test the true performance of object detection algorithms on the signal separation problem in real-world scenarios. This can be done by using an SDR, and if the algorithms are implemented on an integrated processing platform, then these experiments can also confirm the viability of these algorithms to operate in real time.

Detection Metrics: Work remains to be done to quantify the performance of the detector in terms of classical signal processing metrics. This includes measuring the constant false alarm rate and performing an ROC-style sensitivity analysis. These metrics can also be used to compare the performance of object detection methods to traditional wideband sensing methods, even if they only work in simple scenarios.

References

[1] H. Sun, A. Nallanathan, C. Wang, and Y. Chen, "Wideband spectrum sensing for cognitive radio networks: a survey," *IEEE Wireless Communications*, vol. 20, no. 2, pp. 74–81, Apr. 2013.

[2] B. Manz, "The spectrum management challenge," *The Journal of Electronic Defence*, vol. 36, no. 8, pp. 28–35, Aug. 2013.

[3] O. Dobre, A. Abdi, Y. Bar-Ness, and W. Su, "Survey of automatic modulation classification techniques: classical approaches and new trends," *IET Communications*, vol. 1, no. 2, pp. 137–156, Apr. 2007.

[4] A. Nandi and Z. Zhu, *Automatic Modulation Classification: Principles, Algorithms and Applications*, First ed. Wiley, 2015.

[5] S. C. Hauser, W. C. Headley, and A. J. Michaels, "Signal detection effects on deep neural networks utilizing raw IQ for modulation classification," *Proceedings of IEEE Military Communications Conference*, vol. 5, pp. 121–127, Oct. 2017.

[6] T. O'Shea, R. Tamohgna, and T. C. Clancy, "Learning robust general radio signal detection using computer vision methods," in *Proceedings of the Asilomar Conference on Signals, Systems, and Computers*, Pacific Grove, CA, USA, Nov. 2017, pp. 829–832.

[7] H. Nguyen, H. Nguyen, and B. Nguyen, "An image processing approach to wideband spectrum sensing of heterogeneous signals," in *Proceedings of the International Conference on Cognitive*

Radio Oriented Wireless Networks. Lisbon, Portugal: Springer International Publishing, Sep. 2017, pp. 741–753.

[8] X. Zha, H. Peng, X. Qin, G. Li, and S. Yang, "A deep learning framework for signal detection and modulation classification," *Sensors*, vol. 19, no. 18, p. 4042, Sep. 2019.

[9] W. Li, K. Wang, and L. You, "A deep convolutional network for multitype signal detection and classification in spectrogram," *Mathematical Problems in Engineering*, vol. 2020, Sep. 2020.

[10] E. Lo and J. Kohl, "Internet of Things (IoT) discovery using deep neural networks," in *Proceedings of the IEEE Winter Conference on Applications of Computer Vision*, Snowmass Village, CO, USA, Mar. 2020, pp. 795–803.

[11] E. Fonseca, J. F. Santos, F. Paisana, and L. A. Da Silva, "Radio access technology characterisation through object detection," *arXiv e-prints*, p. arXiv:2007.13561, Jul. 2020.

[12] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25*, Lake Tahoe, NV, USA, Dec 2012, pp. 1097–1105.

[13] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation applied to handwritten zip code recognition," *Neural Computations*, vol. 1, no. 4, pp. 541–551, Dec. 1989.

[14] C. M. Bishop, *Pattern Recognition and Machine Learning*. Berlin, Germany: Springer-Verlag, 2006.

[15] S. J. Russell and P. Norvig, *Artificial intelligence: A modern approach*, 3rd ed. Upper Saddle River, NJ, USA: Prentice Hall, 2010.

[16] N. P. Jouppi *et al.*, "In-datacenter performance analysis of a tensor processing unit," *ACM SIGARCH Computer Architecture News*, vol. 45, no. 2, pp. 1–12, Jun. 2017.

[17] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 1, pp. 1–21, Jan. 2019.

[18] A. Garcia-Garcia, S. Orts-Escolano, S. Oprea, V. Villena-Martinez, and J. Garcia-Rodriguez, "A Review on Deep Learning Techniques Applied to Semantic Segmentation," *arXiv e-prints*, p. arXiv:1704.06857, Apr. 2017.

[19] L. Zhou, H. Palangi, L. Zhang, H. Hu, J. J. Corso, and J. Gao, "Unified vision-language pre-training for image captioning and vqa," *arXiv e-prints*, p. arXiv:1909.11059, Sep. 2019.

[20] S. Ren, K. He, R. Girshick, and J. Sun, "Faster R-CNN: Towards real-time object detection with region proposal networks," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 39, no. 6, pp. 1137–1149, Jun. 2017.

[21] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," in *Proceedings of the IEEE International Conference on Computer Vision*. Washington, DC, USA: IEEE Computer Society, Dec. 2015, pp. 1026–1034.

[22] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 770–778.

[23] J. Redmon and A. Farhadi, "YOLOv3: An Incremental Improvement," *arXiv e-prints*, p. arXiv:1804.02767, Apr. 2018.

[24] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proceedings of the International Conference on Learning Representations*, San Diego, CA, USA, May 2015, pp. 1–14.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik, "Rich feature hierarchies for accurate object detection and semantic segmentation," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Washington, DC, USA, Sep. 2014, pp. 580–587.

[26] J. Huang *et al.*, "Speed/accuracy trade-offs for modern convolutional object detectors," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, Hawaii, USA, 2017, pp. 3296–3305.

[27] J. Redmon, "Darknet: Open source neural networks in C," http://pjreddie.com/darknet/, 2013-2016.

[28] C. Szegedy, Wei Liu, Yangqing Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Boston, MA, USA, Jun. 2015, pp. 1–9.

[29] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman, "The Pascal Visual Object Classes (VOC) Challenge," *International Journal of Computer Vision*, vol. 88, no. 2, pp. 303–338, Jun. 2010.

[30] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. "Visual Object Classes Challenge 2012 (VOC2012)", The PASCAL Visual Object Classes Homepage. http://host.robots.ox.ac.uk/pascal/VOC/ [Accessed Aug. 7, 2020].

[31] A. Ahmed, K. Yu, W. Xu, Y. Gong, and E. Xing, "Training hierarchical feed-forward visual recognition models using transfer learning from pseudo-tasks," in *Proceedings of the European Conference on Computer Vision*, Marseille, France, Oct. 2008, pp. 69–82.

[32] M. Oquab, L. Bottou, I. Laptev, and J. Sivic, "Learning and transferring mid-level image representations using convolutional neural networks," in *Proceedings ofthe IEEE Conference on Computer Vision and Pattern Recognition*, Columbus, OH, USA, Jun. 2014, pp. 1717–1724.

[33] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?" in *Advances in Neural Information Processing Systems 27*, Dec. 2014, pp. 3320–3328.

[34] C. Trabelsi, O. Bilaniuk, Y. Zhang, D. Serdyuk, S. Subramanian, J. F. Santos, S. Mehri, N. Rostamzadeh, Y. Bengio, and C. J. Pal, "Deep complex networks," in *Proceedings of the*

International Conference on Learning Representations, Vancouver Canada, May 2018.

[35] M. Arjovsky, A. Shah, and Y. Bengio, "Unitary evolution recurrent neural networks," in *Proceedings of Machine Learning Research*, vol. 48, New York, New York, USA, Jun. 2016, pp. 1120–1128.

[36] T. J. O'Shea, J. Corgan, and T. C. Clancy, "Convolutional radio modulation recognition networks," in *Engineering Applications of Neural Networks*. Aberdeen, UK: Springer International Publishing, Sep. 2016, pp. 213–226.

[37] M. Kulin, T. Kazaz, I. Moerman, and E. De Poorter, "End-to-end learning from spectrum data: A deep learning approach for wireless signal identification in spectrum monitoring applications," *IEEE Access*, vol. 6, pp. 18 484–18 501, Mar. 2018.

[38] R. Girshick, "Fast R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, Washington, DC, USA, Dec 2015, pp. 1440–1448.

[39] J. Uijlings, K. van de Sande, T. Gevers, and A. Smeulders, "Selective search for object recognition," *International Journal of Computer Vision*, vol. 104, no. 2, pp. 154–171, Sep. 2013.

[40] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Las Vegas, NV, USA, Jun. 2016, pp. 779–788.

[41] J. Redmon and A. Farhadi, "YOLO9000: Better, Faster, Stronger," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, Jul. 2017, pp. 6517–6525.

[42] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. Reed, C.-Y. Fu, and A. C. Berg, "SSD: Single Shot MultiBox Detector," in *Proceedings of the European Conference on Computer Vision*, B. Leibe, J. Matas, N. Sebe, and M. Welling, Eds., Amsterdam, Netherlands, Oct. 2016, pp. 21–37.

[43] K. He, G. Gkioxari, P. Dollar, and R. Girshick, "Mask R-CNN," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017, pp. 2980–2988.

[44] T. Acharya and P.-S. Tsai, "Computational foundations of image interpolation algorithms," *Ubiquity*, vol. 8, no. 42, pp. 1–17, Oct. 2007.

[45] A. Paszke *et al.*, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing System*, Vancouver, Canada, Dec. 2019, pp. 8024–8035.

[46] L. Strydom. "Detectorch: A Custom Faster R-CNN Implementation", GitHub. http://www.github.com/LlewellynS96/detectorch/ [Accessed Nov. 30, 2020].

[47] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*, Orlando, Florida, USA, Nov 2014, pp. 675–678.

[48] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," in *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, Honolulu, HI, USA, Jul. 2017, pp. 936–944.

[49] L. Strydom and W. P. du Plessis, "Object detection for signal separation with different time-frequency representations," *IEEE Transaction on Signal Processing*, submitted for publication.

[50] D. Hoiem, Y. Chodpathumwan, and Q. Dai, "Diagnosing error in object detectors," in *Proceedings of the European Conference on Computer Vision*, vol. 3, Florence, Italy, 2012, pp. 340–353.

[51] L. Weng. "Object Detection Part 4: Fast Detection Models", Lil'Log. https://lilianweng.github.io/lil-log/2018/12/27/object-detection-part-4.html [Accessed Nov. 21, 2020].

[52] L. Strydom. "Darktorch: A Custom YOLO Implementation", GitHub. http://www.github.com/LlewellynS96/darktorch/ [Accessed Nov. 30, 2020].

[53] O. Russakovsky, L. Li, and L. Fei-Fei, "Best of both worlds: Human-machine collaboration for object annotation," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, Columbus, OH, US, Oct. 2015, pp. 2121–2131.

[54] Y. Zeng, Y. Liang, and R. Zhang, "Blindly combined energy detection for spectrum sensing in cognitive radio," *IEEE Signal Processing Letters*, vol. 15, pp. 649–652, Oct. 2008.

[55] P. Pham, J. Li, J. Szurley, and S. Das, "Eventness: Object detection on spectrograms for temporal localization of audio events," in *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, Calgary, Canada, Apr. 2018, pp. 2491–2495.

[56] J. Yu, J. Li, B. Sun, J. Chen, and C. Li, "Multiclass radio frequency interference detection and suppression for SAR based on the Single Shot MultiBox detector," *Sensors*, vol. 18, no. 11, Nov. 2018.

[57] J. B. Allen, "Short term spectral analysis, synthesis, and modification by discrete fourier transform," *IEEE Transactions on Acoustics*, vol. 25, no. 3, pp. 235–238, Jun. 1977.

[58] E. R. Zilberman and P. E. Pace, "Autonomous time-frequency morphological feature extraction algorithm for LPI radar modulation classification," in *Proceedings of the International Conference on Image Processing*, Atlanta, GA, USA, Oct. 2006, pp. 2321–2324.

[59] M. Gandetto, M. Guainazzo, and C. S. Regazzoni, "Use of Time-Frequency Analysis and Neural Networks for Mode Identification in a Wireless Software-Defined Radio Approach," *EURASIP Journal on Applied Signal Processing*, vol. 2004, no. 12, pp. 1778–1790, Dec. 2004.

[60] K. Markwardt, "Wavelet analysis and frequency band decompositions," in *Proceedings of the International Conference on the Application of Computer Science and Mathematics in Architecture and Civil Engineering*, Weimar, Germany, Jul. 2006, pp. 1–22.

[61] G. Stimson, *Introduction to Airborne Radar*, 2nd ed. SciTech Pub., 1998.

[62] J. R. Nunez, C. R. Anderton, and R. S. Renslow, "Optimizing colormaps with consideration for color vision deficiency to enable accurate interpretation of scientific data," *PLOS ONE*, vol. 13, no. 7, pp. 1–14, Aug. 2018.

[63] S. Guiasu and A. Shenitzer, "The principle of maximum entropy," *The Mathematical Intelligencer*, vol. 7, no. 1, pp. 42–48, 1985.

[64] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA, USA: The MIT Press, 2016.

[65] J. G. Proakis and M. Salehi, *Digital Communications*, 5th ed. McGraw Hill, 2007.

[66] S. Marcel and Y. Rodriguez, "Torchvision the machine-vision package of torch," in *Proceedings of the ACM International Conference on Multimedia*, Firenze, Italy, Oct. 2010, pp. 1485–1488.

[67] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Soft-NMS – Improving Object Detection with One Line of Code," in *Proceedings of the IEEE International Conference on Computer Vision*, Venice, Italy, Oct. 2017, pp. 5562–5570.

[68] M. Tan, R. Pang, and Q. V. Le, "Efficientdet: Scalable and efficient object detection," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, Virtual Address, Jun. 2020, pp. 10 781–10 790.

[69] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," *arXiv e-prints*, p. arXiv:2005.12872, May 2020.

[70] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "YOLOv4: optimal speed and accuracy of object detection," *arXiv e-prints*, p. arXiv:2004.10934, Apr. 2020.

[71] G. Jocher *et al.*, "Ultralytics/YOLOv5: v3.1 - Bug Fixes and Performance Improvements," Oct. 2020.

[72] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift,” in *Proceedings of the International Conference on International Conference on Machine Learning*, vol. 37, no. 32, Lille, France, Jul. 2015, pp. 448–456.

Addendum A DEEP LEARNING TERMINOLOGY

A.1 INTRODUCTORY REMARKS

This addendum contains explanations for several terms and techniques that are commonly used in deep learning. These explanations should provide readers, with no background in deep learning, with enough knowledge to understand these concepts in the context of this thesis. This is by no means a comprehensive list of deep learning terms, and nor does it aim to be as such.

A.2 BATCH NORMALISATION

Batch normalisation has been proposed as a method to make it easier and faster to train deep neural networks [72]. The batch normalisation layer is commonly used in CNNs and can be placed between any two convolutional layers.

The problem that batch normalisation addresses is that of *internal covariate shift*, *i.e.* the changes in how inputs are distributed at each layer during training. Batch normalisation attempts to remedy this by normalising the data across each dimension in a mini-batch. Specifically, for a layer with d input channels $x = \left(x^{(1)} \dots x^{(d)}\right)$ the normalisation is given by

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathrm{E}\left[x^{(k)}\right]}{\sqrt{\mathrm{Var}\left[x^{(k)}\right]}}, \tag{A.1}$$

where the expectation and variance is calculated as the running mean over all mini-batches in the training set.

If all inputs are normalised, it may change what the layer can represent, since most of the inputs will be constrained to a small, possibly linear, region of the nonlinearity, and hence the batch normalisation

layer adds a scale and shift parameter. The output of the batch normalisation layer is given by

$$y^{(k)} = \gamma^{(k)}\hat{x}^{(k)} + \beta^{(k)}. \tag{A.2}$$

A.3 CONVOLUTIONAL LAYER

The convolutional layer is the fundamental building block of any CNN. The convolutional layer consists of several filters – also called kernels – which comprise a set of weights which is convolved with the input of the layer. The convolution operation means that the same filters are applied at several different positions across the input space, which gives it its shift invariant property.

A convolutional layer is often followed by an additive bias and an activation function, both of which are often considered part of the convolutional layer. Convolutional layers are often stacked in order for them to learn complex, nonlinear features.

The output of a convolutional layer is referred to as a feature map. A convolutional layer with C filters which produces a feature map of width H and height W produces an output that can be represented by a tensor of dimensions $H \times W \times C$.

A.4 FULLY-CONNECTED LAYER

A fully-connected layer is a neural network building block where each of the N input neurons are connected to each of the M output neurons by separate weights. The weights can be represented by an $N \times M$ matrix.

A.5 POOLING

A pooling function replaces the output values of a convolutional layer at a certain location with a summary statistic of the nearby outputs [64]. There are three common reasons for including pooling layers in CNNs. The first is to help the network to be approximately invariant to small transformations of the input, and this is true for any choice of pooling function [64]. Secondly, by reporting summary statistics for regions that are spaced k pixels apart, the computational and memory requirements are

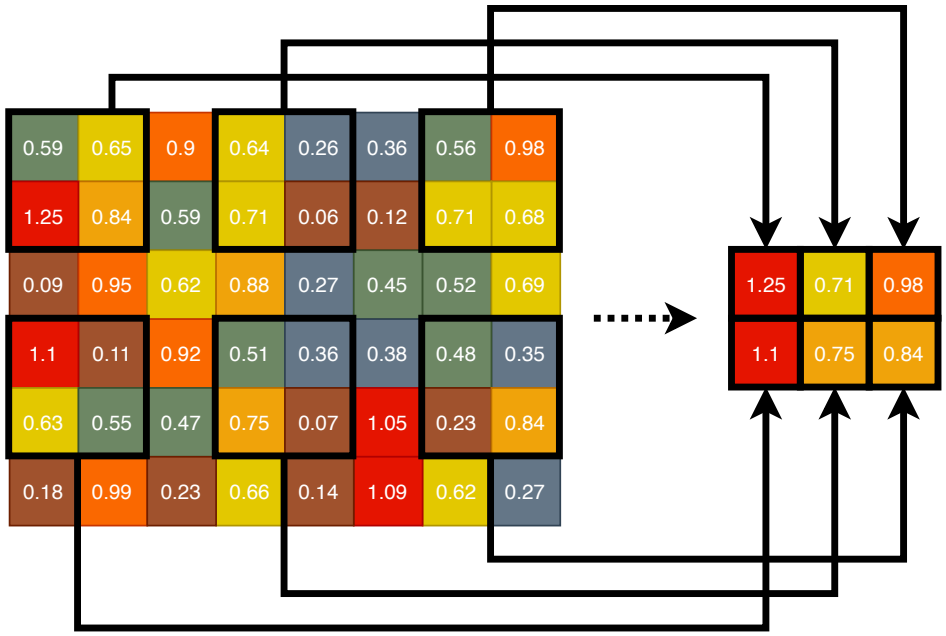


Figure A.1. This example shows max pooling with a kernel size of 2×2, and a vertical and horizontal stride of 3. It is common to use a stride that is either smaller than or the same size as the kernel.

approximately reduced by k [64]. Pooling layers also make it possible to handle variable size inputs, by producing fixed size outputs [22, 38].

A.5.1 Max Pooling

First introduced in 1988, even before CNNs, max pooling has become a common building block for modern deep CNNs. The max pooling function reports the maximum output value from a local rectangular region. If these regions are spaced more than 1 pixel apart (values of 2 and 2 are common) then this has the effect of reducing the size of the input feature map, while also producing features that are more resilient. The kernel size of a pooling function is the size of the regions that are considered, while the stride refers to the number of pixels between adjacent regions. An example of max pooling is shown in Figure A.1.

A.5.2 ROI Pool and ROI Align

These pooling functions are discussed in Section 2.3.1.1 and Section 2.3.1.2 respectively.

A.6 RECTIFIED LINEAR UNIT

Both fully-connected and convolutional layers are linear operations that can be represented by simple affine transformations. These layers are thus incapable of learning non-linear functions, unless their outputs are passed through a non-linear activation function. This is the reason all neural networks use non-linear activations for the hidden layers.

ReLUs are similar to linear units, except that they do not pass negative values. The activation function for a ReLU is

$$g(z) = \max(0, z). \tag{A.3}$$

This makes them easy to optimise, since they have a gradient that is constant across half its domain and zero elsewhere, which mitigates the problems associated with activation functions that introduce second-order effect [64].

Many state-of-the-art deep CNNs add ReLUs after each convolutional layer [21, 22].

A.7 TENSOR

A tensor is an array of numbers arranged on a regular grid with a variable number of axes [64]. In computer science, tensors are often simply referred to as matrices.

Addendum B OBJECT DETECTION TERMINOLOGY

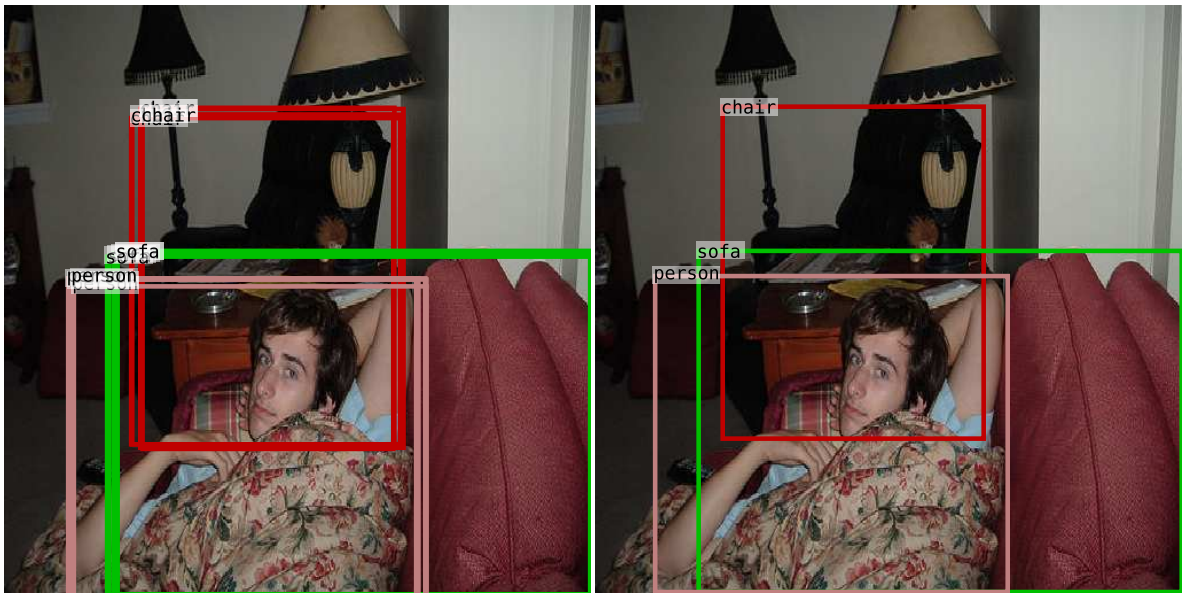
B.1 INTRODUCTORY REMARKS

This addendum contains explanations for several terms and techniques that are commonly used in object detection. These explanations should provide readers with enough knowledge to understand these concepts in the context of this thesis. This is by no means a comprehensive list of deep learning terms, and nor does it aim to be such a list.

B.2 ANCHOR BOXES

Anchors, bounding box candidates or priors, as they are often referred to in the statistical parlance, are default bounding boxes that are spaced at regular intervals across an image, as they would be in a sliding window approach. They are used to paramaterise bounding box outputs, so as to limit the outputs to more sensible ranges than if they were parameterised according to the size of the image. Most approaches that make use of anchors use several anchors at each position, each associated with a different scale and aspect ratio. The range of anchors used should represent all of the objects to be detected, large and small.

If the stride of the sliding window, which determines the position of the anchors, is chosen such that there are W positions across and H positions down, and the number of anchors at each position is k, then there are WHk anchors in total. In the case of the RPN in Faster R-CNN, anchors are centred at each position on the final convolutional feature map, which typically has dimensions 60×40, for a total of 2 400 anchors.



(a) Without NMS. **(b)** With NMS.

Figure B.1. An example of NMS applied to the output of an object detection algorithm.

B.3 NON-MAXIMUM SUPPRESSION

NMS is a technique which reduces the number of overlapping detections for each instance of an object. NMS has been shown to improve the AP, and thus mAP, of many object detection algorithms [40], and is even deemed a crucial and necessary step for some [25, 42]. NMS can be applied to the output of any object detection algorithm that produces a confidence score for each predicted bounding box.

In practice, NMS is performed on a per-class basis, which means that overlapping detections of different classes will not affect one another. The first step when performing NMS in this fashion is to sort the detections (of a single class) according to their confidence scores. The list of detections is then traversed, starting with the detection with the highest score. All detections that have an IoU less than some threshold for all the previously selected detections are selected. This process is then repeated for each class separately.

An example of NMS at work is given in Figure B.1. The bounding boxes were synthetically produced to ensure multiple predictions and an IoU threshold of 0.45 was used.