Pulse Repetition Interval Pattern Classification with Deep Learning Methods for Computer Vision

Il Hwan Seo

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
30 credits

Department of Technology Systems
The Faculty of Mathematics and Natural Sciences

UNIVERSITY OF OSLO

Autumn 2023

Pulse Repetition Interval Pattern Classification with Deep Learning Methods for Computer Vision

Il Hwan Seo

Pulse Repetition Interval Pattern Classification with Deep Learning Methods for Computer Vision

http://www.duo.uio.no/

Printed: Reprosentralen, University of Oslo

Abstract

Electronic support is a subdivision of electronic warfare, which involves intercepting and identifying radiated electromagnetic energy to identify the emitter and unveil the associated platform's presence. A pulsed radar is often the target of such activity, where interception and measurement of the emitted radar pulses allow disclosure of information about the emitter, further allowing its identification. Recognition of a key attribute of the pulsed radar, called pulse repetition interval (PRI) is essential in the identification process.

This thesis investigates whether it is possible to automate the classification of PRI patterns formed by PRI values derived from a radar pulses' time of arrival. The novelty of this task lies in that classification is performed not on the general PRI modulation types, but on a particular set of PRI patterns commonly used by navigation radars. This provides a finer subdivision of the usual modulation types, better suited for recognizing and tracking individual navigation radars.

From a sequence of PRI values, data points for a scatter plot is generated. The plotted data points give a graphical pattern representing the particular PRI pattern of the radar. The images of patterns were processed and employed as a development set for training and evaluation of a deep learning based object detection and classification model. The detection model reliably inferred the position and size of the PRI pattern within the image, enabling the generation of new magnified image with improved suitability for classification. This ensured that the classifier mostly received images with optimally sized pattern.

The resulting classification gave near-flawless classification performance with a total accuracy of 0.989. The optimistic outcome was likely attributed to overfitting stemming from the employment of a powerful model, duplicate samples across the training and evaluation sets, and lack of variations within each pattern class after the image processing. While this does not render the model unusable, a careful consideration should be given to its deployment environment.

Future work involves modification and refinement of the detection model to enable precise inference of the lower and upper bounds of PRI values that constitute the PRI pattern. Furthermore, a classifier capable of open set recognition should be developed to mitigate misclassification caused by pattern class outside of the development set, and additionally allow more timely discovery of new pattern classes.

Sammendrag

Elektronisk støtte er en underkategori av elektronisk krigføring, som blant annet har som mål å fange opp og identifisere utstrålt elektromagnetisk energi, i den hensikt å identifisere emitteren og avdekke tilstedeværelsen til den tilknyttede plattformen. En pulsradar er ofte målet i denne aktiviteten, hvor måling av utstrålt radarpuls gir informasjon om senderen, og muliggjør identifisering. En nøkkelegenskap ved en pulsradar er pulsrepetisjonsintervall (PRI). Gjenkjenning av PRI spiller en sentral rolle i identifiseringsprosessen.

Denne oppgaven undersøker muligheten til å automatisere klassifisering av PRI-mønstre formet av PRI-verdier fra ankomsttidspunktene til radarpulser. Nytt i denne oppgaven er at klassifiseringen ikke utføres på de generelle PRI-modulasjonstypene, men bestemte PRI-mønster brukt av navigasjonsradarer. Dette gir en finere inndeling av de generelle modulasjonstypene, som er bedre egnet for gjenkjenning og sporing av individuelle navigasjonsradarer.

Ut i fra en sekvens med PRI-verdier, genereres datapunkter som plottes grafisk. Dette skaper et bilde av et mønster som representerer det bestemte PRI-mønsteret til radaren. Bildene ble prosessert, og inngikk i datasettet for trening og evaluering av dyplæringbasert objektdeteksjon- og klassifiseringsmodell. Deteksjonsmodellen predikerte posisjonen og størrelsen til mønsteret i bildet på en pålitelig måte, og tilrettela for å generere et nytt bilde av forstørret mønster egnet for klassifisering. Dette sørget for at klassifikatoren for det meste mottok bilder med optimal størrelse på mønsteret.

Den endelige klassifiseringen var nesten feilfri, og oppnådde en total nøyaktighet (accuracy) på 0,989. Det optimistiske resultatet var sannsynligvis forårsaket av overtilpasning, med opphav i bruk av en kraftig modell, dupliserte bildesampler på tvers av trening- og evaluerinssett, samt manglende variasjoner innen hver klasse. Dette gjør ikke modellen ubrukelig, men det bør dog utvises forsiktighet med hensyn til miljøet hvor denne modellen skal anvendes.

Forslag til fremtidig arbeid er modifikasjon og forbedring av deteksjonsmodellen for å muliggjøre presis bestemmelse av nedre og øvre grenser for PRI-verdier som utgjør mønsteret innenfor bildet. Videre bør det utvikles en klassifikator med evne til å gjenkjenne mønsterklasser utenfor klassene i treningssettet. Dette vil minske feilklassifiseringer forårsaket av ukjent mønsterklasse, samtidig tilrettelegge for betimelig oppdagelse av nye mønsterklasser.

Contents

List of Figures

List of Tables

List of Acronyms

FFI Forsvarets Forskningsinstitutt

GIoU Generalized Intersection over Union

IoGT Intersection over Ground Truth

IoU Intersection over Union

PRI Pulse Repetition Interval

ToA Time of Arrival

Acknowledgements

Three and a half years of being stretched thin between full time work and part time study is nearing its end. Before I crack open a cold one to celebrate the liberation from the tyranny of academic stress, I would like to express my gratitude to my supervisors Rune Sundgot and Narada Dilp Warakagoda at Forsvarets Forskningsinstitutt. Thank you for providing a task within a field that I'm deeply interested in, and thorough guidance along the way. I would also like to thank my employer, the Norwegian Armed Forces, and my colleagues for facilitating the completion of the thesis.

Chapter 1

Introduction

Electronic Warfare (EW) is any military action with the objective of controlling the electromagnetic spectrum (EMS). This objective is achieved through three interlocking subdivisions of EW: Offensive electronic attack (EA), defensive electronic protection (EP), intelligence gathering, and threat recognition through electronic support (ES) [1], [2]. The latter is the focus of this thesis.

ES is a subdivision of EW, involving actions to search for, intercept, identify, and locate or localize electromagnetic signals of interest. A central part of this activity is immediately recognizing an emitter and providing information required for further actions [3]. A type of radiated electromagnetic energy of interest in ES context is energy emitted from radars. A key instrument in ES is a set of equipment called an electronic support measure (ESM) system, which allows a military unit to detect the direction of the intercepted signal and identify its emitter[1], [4].

The frigate HNoMS Fridtjof Nansen depicted in Figure 1.1 is an example of a naval vessel equipped with an ESM system. The magnified section shows the antenna of its radar-ESM system.

An ESM system needs to measure the intercepted radar pulses to identify a radar emitter. The measured quantities, called radar pulse parameters, are compared to those of known emitters [5]. The known emitters' parameters reside in an ESM system's emitter database. Entries in the database are composed of information acquired from signals from prior collections. These collected signals are processed, analyzed, and verified before being stored in an emitter library, whence a database can be generated [4].

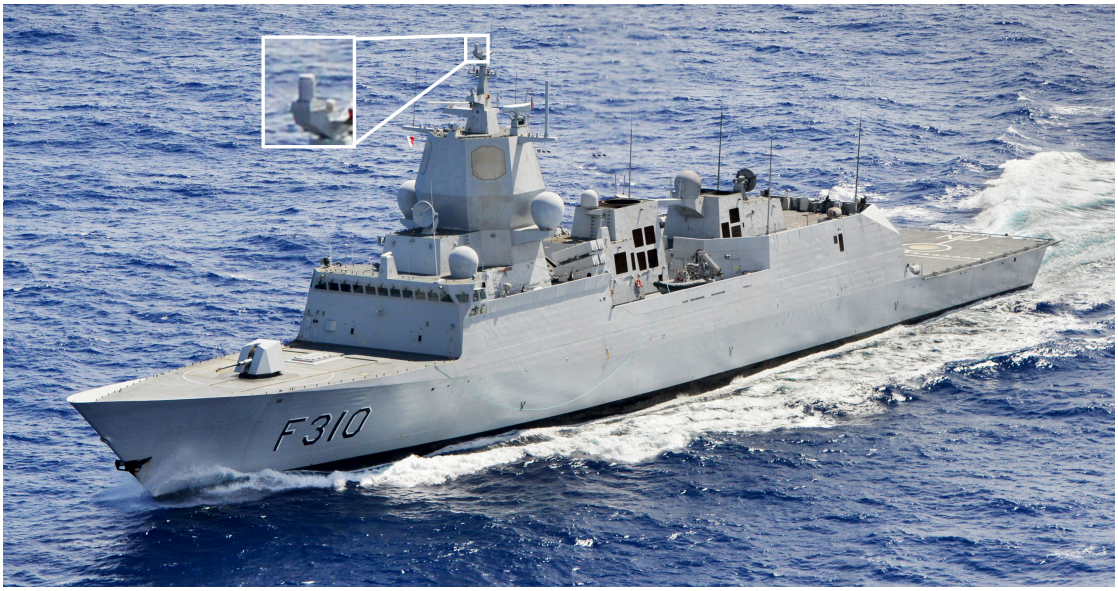
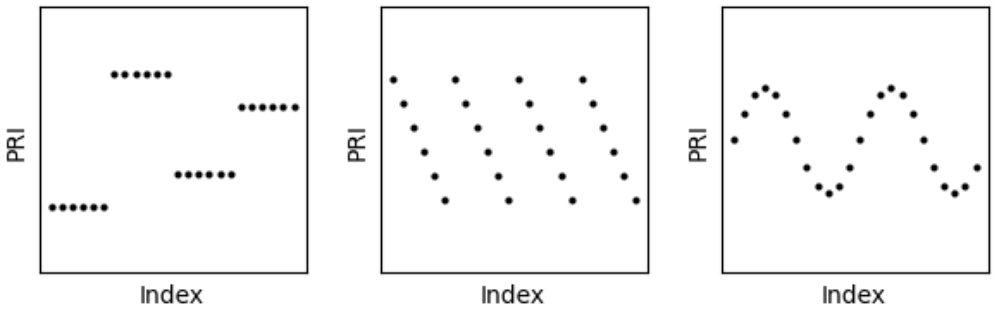


Figure 1.1: HNoMS Fridtjof Nansen. Photo by Marthe Brendefur, Norwegian Armed Forces

One of the most important pulse parameters for identifying a radar emitter is pulse repetition interval (PRI) [6], [7]. PRI is the time interval between the leading edges of two successive radar pulses. In other words, it is the interval between the start of a pulse and the start of the next pulse. Furthermore, it is common that radars operate with PRI that vary from pulse to pulse. The function that dictates this variation is called PRI modulation. PRI modulation can disclose knowledge of a radar's characteristics and function, enabling its identification [8]. Therefore, the emitter database must contain correct information regarding the PRI of radar that the ESM system seeks to identify. Recognizing PRI and modulation is an important component in emitter identification.

There are various methods of analyzing and verifying PRI modulation types. Simple statistical techniques like histograms have been used [8]. Other methods include visualizing the parameters with a scatter plot, which generates a graphical pattern [9]. Each modulation type generates its characteristic pattern, which an analyst can recognize and classify.



(a) (b) (c)

Figure 1.2: Examples of graphical patterns of PRI

1.1 Problem

Recognizing the PRI by visualizing the PRI values is a manual process. For each sequence of radar pulses, it is required to generate a suitable image that adequately visualizes the pattern to be recognized. This involves adjusting the image parameters to suitably depict the pattern for classification. Given the multiple operations that go into this analysis, the process can be tedious and increasingly time-consuming with the amount of data.

1.2 Purpose

This thesis aims to determine whether a deep-learning-based object detection and image classification can automate the manual inspection of images containing PRI patterns to recognize the particular type of pattern used. If successful, the result can relieve a significant part of the workload involved in a post-collection analysis process and prove to be a new method for PRI pattern classification.

1.3 Research Question

Is it possible to automate the manual process of inspecting PRI patterns to recognize PRI pattern types using deep learning methods for computer vision?

Chapter 2

Background and Theory

2.1 Radar

Radar is an acronym for *radio detection and ranging*. It is a sensor system utilizing electromagnetic waves for the detection and localization of objects [2]. This technology operates by emitting electromagnetic waves that reflect off objects and return to the radar. The range to the target is determined by measuring the time difference between the emission of energy and the reception of reflections. A radar typically uses a directional antenna to emit electromagnetic waves with directivity. The direction to the detected object is determined by measuring the direction of the antenna at the time when the reflection is received [10]. Emitted electromagnetic waves can be in the form of either pulses or continuous waves. Radars that operate with pulsed energy are called pulsed radars and are the most common [11].

Pulsed radars emit short bursts of electromagnetic waves called radar pulses. Each pulse is emitted for some duration, called pulse length or pulse width. After the pulse emission, there is a time period where radar waits for a possible return of a reflection from an object before the next pulse is emitted. This waiting time is influenced by a parameter called pulse repetition interval (PRI), which is the time interval between the start of a radar pulse and the start of the next pulse. The magnitude of these parameters can be in the order of tens of microseconds to a few milliseconds. PRI significantly influence the design of radars and directly impact the operational specifications, namely correct range determination or the ability to discern objects in close proximity [12].

Radars have diverse applications ranging from adaptive cruise control to navigational aid and weather forecasting. It has quintessential military applications, such as surveillance, target tracking, and weapon guidance. Each application imposes a distinct set of requirements, distinguishing a radar for its specific purpose. Pulse parameters are designed to meet the operational requirements of the radar's application.

2.2 Electronic Warfare

Electronic warfare (EW) is any military action involving the use of electromagnetic energy and directed energy to control the electromagnetic spectrum or to attack the enemy. The three divisions of the EW are electronic attack, electronic protection, and electronic support. The latter involves actions to search for, intercept, identify, and locate or localize sources of intentional and unintentional radiated electromagnetic energy for the purpose of immediate threat recognition, targeting, planning, and conducting future operations [13].

Pulses emitted from radars are an example of radiated electromagnetic energy that serves the purposes in the ES definition. These emissions can be intercepted and measured with an ESM system. The measured parameters from the pulses can disclose information about the radar, as they can be matched with the parameters of known emitters. This allows identification of the specific radar whence pulses originated. Furthermore, the identification can further associate the emitter with the platform carrying the identified radar, such as a vessel, aircraft or even an incoming missile, thereby disclosing information about the type of platform present and its intention [4].

For successful identification of radar type, the ESM system requires a storage of parameters of all considered radar types in a table. This table is called an emitter database and serves as a reference during the emitter identification process. When radar pulses are intercepted, the measurements are compared to the values in the parameter table. The applicable radar or signal type is reported upon a parametric match. In case of no parametric match, the emitter is reported as unknown [14]. One of the most critical parameters for successful identification is pulse repetition interval. Determining the PRI values and the PRI modulation type is essential in recognizing this pulse parameter. [6], [7].

2.3 Pulse Repetition Interval

PRI is the time interval between the emission of two successive radar pulses. This is a parameter, which impacts the radar's ability to correctly determine the range to an object. For a radar pulse sequence with n pulses, an ESM system can measure $n-1$ PRI values as illustrated in Table 2.1.

Time of Arrival of radar pulses	Pulse Repetition Interval
ToA_1 ToA_2 ToA_3	$PRI_1 = ToA_2 - ToA_1$ $PRI_2 = ToA_3 - ToA_2$
$\vdots$	$\vdots$
ToA_i ToA_{i+1}	$PRI_i = ToA_{i+1} - ToA_i$
$\vdots$	$\vdots$
ToA_{n-1} ToA_{n+1}	$PRI_{n-1} = ToA_n - ToA_{n-1}$

Table 2.1: Calculation of PRI from ToA of pulses

The calculated PRI values $PRI_1, PRI_i, \ldots, PRI_{n-1}$ constitute what is referred to as the PRI sequence.

2.3.1 Pulse Repetition Interval Modulations

In the context of a sequence of intercepted pulses, denoted by index i, and corresponding time of arrival denoted t_i, the following expression is applicable when PRIs are measured.

$$\begin{gathered} PRI_i = ToA_{i+1} - ToA_i \\ i \in \mathbb{N}_0 \end{gathered} \tag{2.1}$$

However, it is important to mind that the determination of PRI values of emitting radar is dictated by its PRI modulation, which follows a certain function.

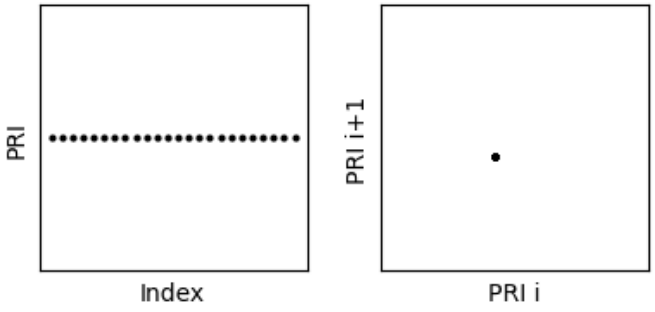
$$PRI_i = F(i) \tag{2.2}$$

PRI modulations can be categorized into six different types. The following descriptions of the modulations are accompanied by mathematical descriptions from the thesis work of Eric Norgren [14], and figures which depict the PRI modulation in two different methods. The first method is the *PRI, index plot* , where the x-axis is the index of the PRI while the y-axis is the value of the PRI of the given index. This illustrates the modulation in a temporal aspect. The second illustration method is PRI_i, PRI_{i+1} *plot* , which illustrates distinct shifts in PRI values in the sequence. It illustrates how the PRI value changes between each successive pulse pair. In this thesis, images generated in the latter method is referred to as PRI image, and is the focus of the thesis.

Constant

Constant PRI, also known as fixed PRI, has no PRI modulation. With this modulation type, the PRI between pulses remains unchanged [8].

$$PRI_i = k \tag{2.3}$$



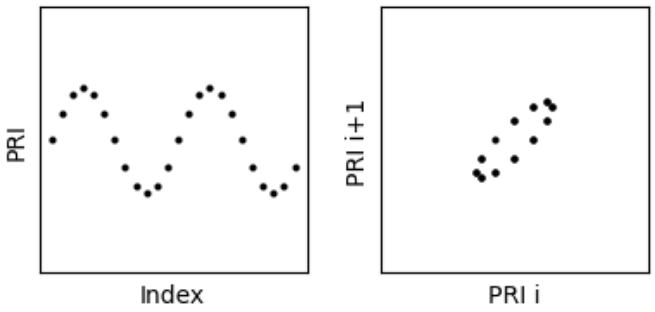
(a) (b)

Figure 2.1: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of constant modulation

Sinusoidal

With the sinusoidal PRI modulation, the PRI changes in a sinusoidal fashion. This modulation is also known as wobulated PRI [8].

$$PRI_i = PRI_0 + A * \sin(\omega * x_i + \phi) \tag{2.4}$$



(a) (b)

Figure 2.2: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of sinusoidal modulation

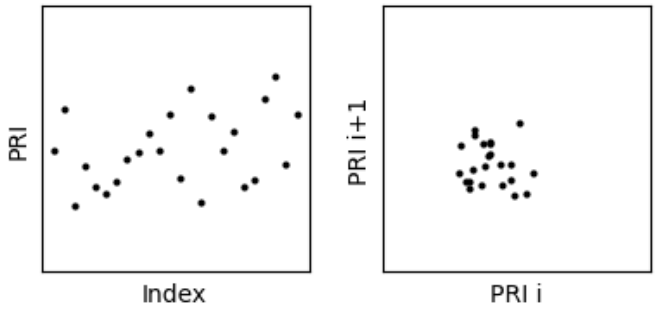
Jitter

With jitter PRI modulation, the PRI values are random. The PRI values are sampled from a uniform distribution with an upper and lower limit, or a Gaussian distribution with some mean and standard deviation [8].

$$PRI_i \sim U(min, max) \tag{2.5}$$

or

$$PRI_i \sim N(\mu, \sigma^2) \tag{2.6}$$



(a) (b)

Figure 2.3: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of jitter modulation

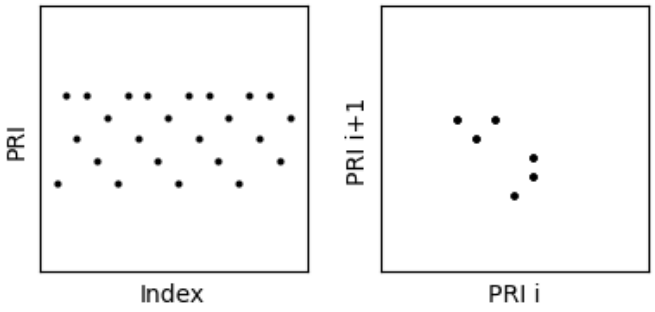
A variant of the jitter modulation also exists, known as *discrete jitter*, where PRI values are randomly sampled from a pool of predefined discrete values.

Stagger

With the stagger PRI modulation, the PRI takes a pattern, following an ordered sequence of discrete values before repeating. The number of discrete values in the ordered sequence is called elements, and the length of the sequence is called the number of positions. This sequence, with the length equal to the number of positions, is referred to as a frame [15].

$$PRI_i = y_j \tag{2.7}$$

Where $j = i \mod M$.
j is the position in the sequence and M is the number of positions in the frame.



(a) (b)

Figure 2.4: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of stagger modulation

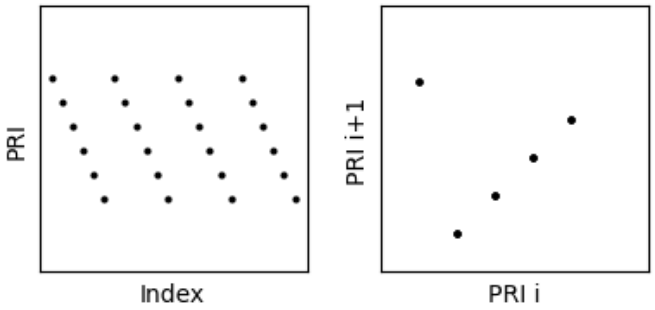
Depictions in Figure 2.4 are an example of countless variations within the stagger modulation. Combinations of frame lengths and element values, and permutations of arrangement order give a vast diversity to the resulting patterns.

Sliding

With sliding PRI modulation, the PRI increases or decreases monotonically until it reaches a limit, followed by a switch to the opposite limit. The monotonic change in PRI is not necessarily linear [8]. This modulation can be seen as a subset of stagger modulation, where the ordered sequence contains monotonically changing values.

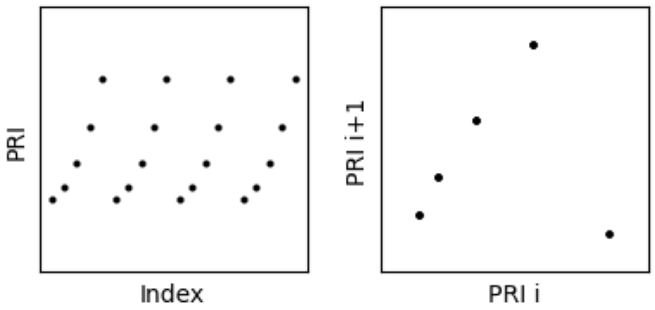
$$PRI_i = PRI_0 + \delta * (i \mod M) \tag{2.8}$$

$\delta \in \mathbb{R}$ is a value describing the rate of change.



(a) (b)

Figure 2.5: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of linearly sliding down modulation



(a) (b)

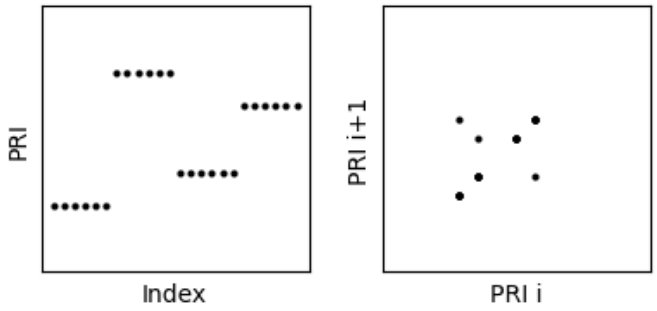
Figure 2.6: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of exponentially sliding up modulation

Dwell and Switch

In dwell and switch modulation, the PRI remains constant over a short duration. This duration is called the dwell. After this period, PRI switches to the next PRI value. Similar to the stagger modulation, there is an ordered list of PRI values. PRIs dwell on a value, before switching to the next value in the list. The list repeats itself after the final PRI value [15].

$$PRI_i = \begin{cases} y_1 & 0 \leq (i \mod x_M) < x_1 \\ y_2 & x_1 \leq (i \mod x_M) < x_2 \\ \vdots & \vdots \\ y_M & x_{M-1} \leq (i \mod x_M) < x_M \end{cases} \tag{2.9}$$

y_j represents the PRI value during the j-th dwell, given to the pulses indexed between x_{j-1} and x_j. The dwell and switch sequence resumes from y_1 after the pulse number x_M, denoting the total number of pulses within the entire dwell and switch sequence. This also implies that the sequence repeats itself after switching $M-1$ times.



(a) (b)

Figure 2.7: PRI, index plot (a) and PRI_i, PRI_{i+1} plot (b) of dwell and switch modulation

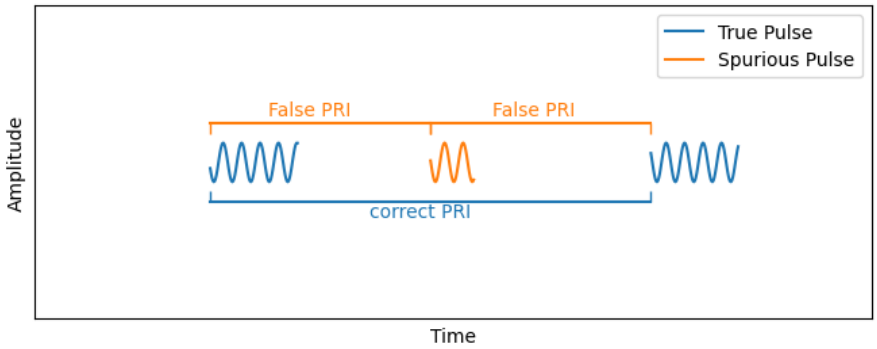
Like the stagger modulation, depictions in Figure 2.7 are a single example from countless variations within the dwell and switch modulation. Varying frame lengths, element values, and arrangement order give the resulting pattern a vast diversity.

2.3.2 Noise in PRI measurements

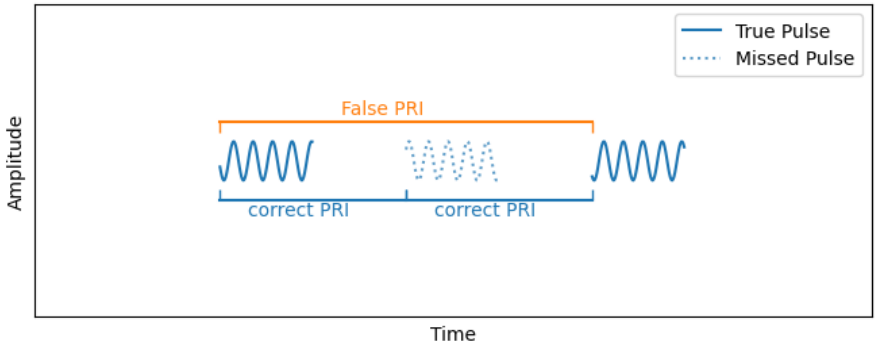
In related work regarding PRI measurement and modulation classification, two noise categories in PRI measurements stand as notable concerns. The first category is spurious pulses. This occurs when pulses not belonging to the radar under measurement are included in the measured pulse sequence. This can happen when an ESM system receives pulses from multiple radars simultaneously. In such scenarios, the deinterleaving process, which is the process of correctly sorting pulses associated with the same emitter, may fail to sort all pulses correctly [16].

Another cause of spurious noise is the combination of elevated background noise and heightened sensor sensitivity. This may cause the processing of random noise as if it were a real radar pulse. Consequently, the PRI is measured between the genuine pulse and a spurious (false) pulse, as depicted in Figure 2.8(a).

The second category of noise in PRI measurement is the missing pulses. This is caused by pulses that fail to be processed and are consequently excluded from the pulse sequence for the measured radar. This is caused by pulses simply not being detected, or pulses erroneously sorted to another emitter in the deinterleaving process. In such scenario, the measured PRI is erroneously greater than what it should have been without the absence of the missing pulses.



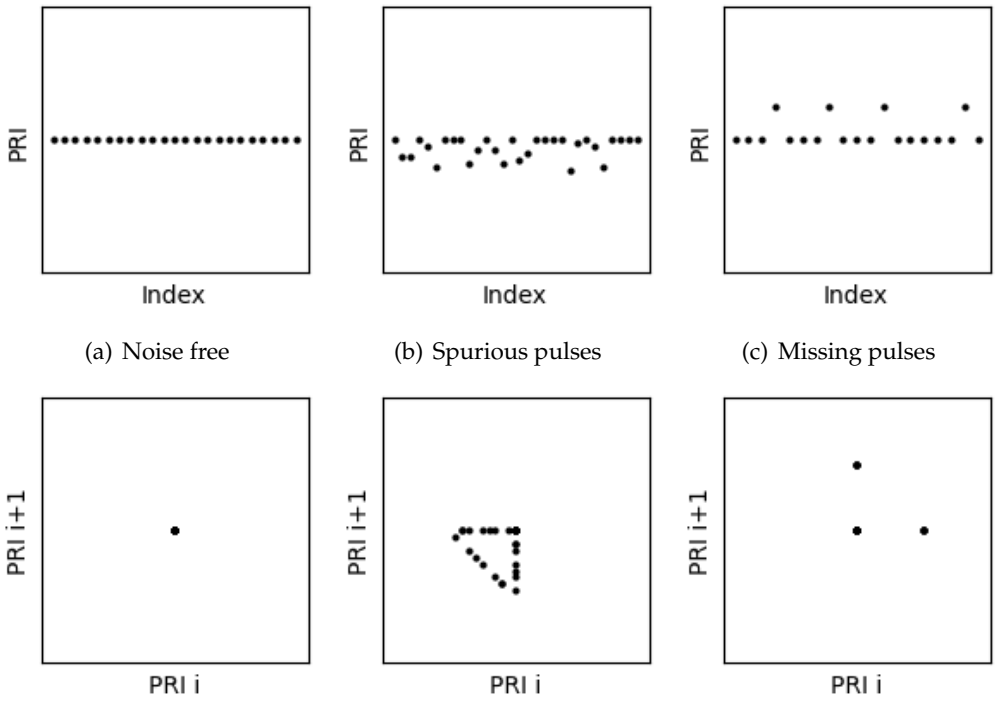
(a)



(b)

Figure 2.8: Spurious (a) and missing (b) pulse noise

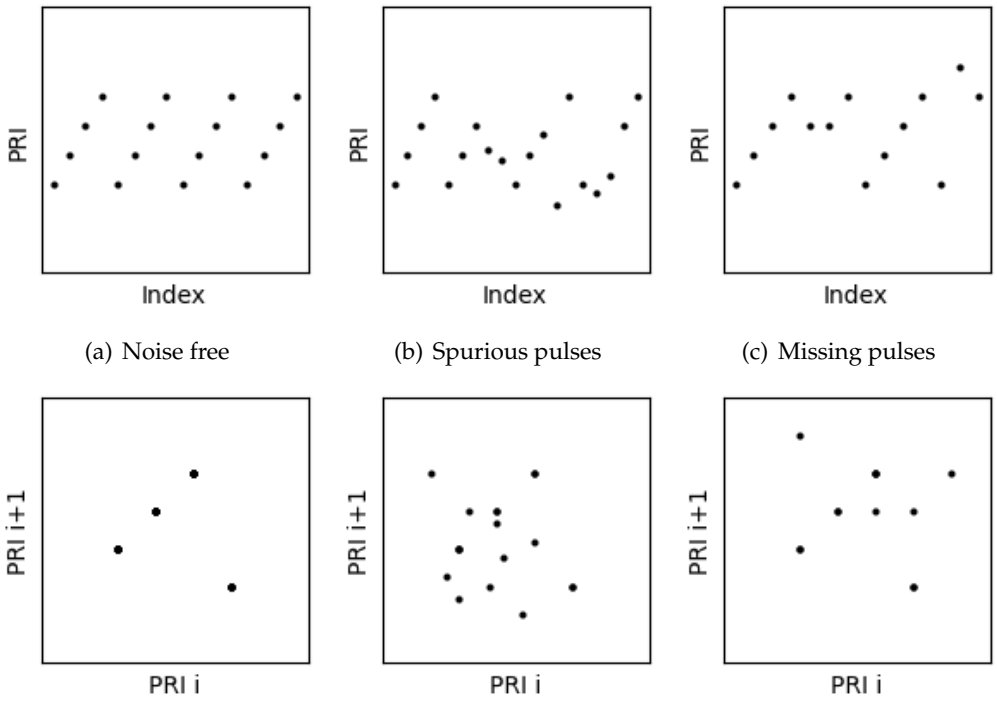
Figure 2.8 illustrates how PRI noise manifests during the PRI measurement. Figures in Figure 2.9 and 2.10 illustrate constant and sliding modulations with noise from the described categories, and demonstrate how noise can disrupt the accurate perception of the two modulation types.



(a) Noise free (b) Spurious pulses (c) Missing pulses

(d) Noise free (e) Spurious pulses (f) Missing pulses

Figure 2.9: Impact of PRI noise on constant modulation



(a) Noise free (b) Spurious pulses (c) Missing pulses

(d) Noise free (e) Spurious pulses (f) Missing pulses

Figure 2.10: Impact of PRI noise on sliding modulation

2.4 Deep Learning

Deep learning is a field within machine learning. Machine learning is the study of algorithms that can learn from experience. The algorithms are designed to gather experience from observed data, referred to as the training data. Its performance improves with the accumulation of this experience, improving its ability to make correct inferences [17]. The inferences, also known as predictions, can be numerical regression or categorizing given input data into predefined categories, called classification [18].

Deep Learning is one of many techniques within the field of machine learning. It is a particularly powerful technique which employs a data structure called a neural network. Deep learning has made groundbreaking advances in several machine learning applications, such as computer vision and natural language processing [17].

2.4.1 Artificial Neural Network

An artificial neural network is a data structure inspired by the functionalities of a biological brain and nervous system. It is a data structure consisting of directionally connected nodes, referred to as neurons, organized in layers. Among the many forms of an artificial neural network, the densely connected feed-forward neural network, also known as multi-layer perceptron (MLP), stands as the quintessential form [19].

The architecture of an MLP is characterized by three principal layers, known as the input, hidden, and output, that are directionally and densely connected, as depicted in Figure 2.11 [20]. Directionally connected refers to the unidirectional flow of information in a feed-forward manner. There is no connection going in the reverse direction. Densely connected, interchangeably known as fully connected, means that a neuron has connections to every neuron in the subsequent layer. In other words, every neuron in a given layer receives information from every neuron in the preceding layer and subsequently transmits information to every neuron in the succeeding layer.

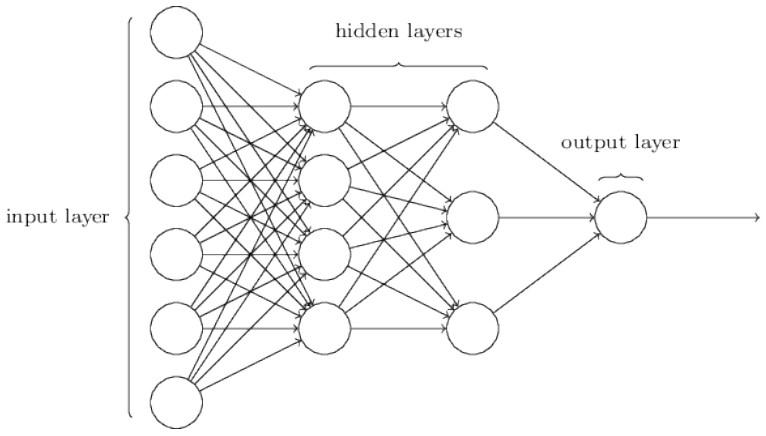


Figure 2.11: Principal layers of a multi-layer perceptron

The number of neurons in input and hidden layers reflect the dimension of input and output data, respectively. Hidden layers give depth to the network, facilitating increased complexity. Each connection represents a linear function, where the output from a neuron is multiplied by some weight designated to the connection. In the receiving node, the inputs are summed. With an added bias, this weighted sum serves as a signal that activates the neuron through a non-linear activation function. Notable examples of the activation functions are rectified linear unit (ReLU) and sigmoid functions [21].

The primary objective of a neural network is to approximate a desired function. This is accomplished through a training process, where the weights associated with each connection and the bias of each neuron are adjusted iteratively to push the output for each input closer to the respective desired output [19].

Training ANN

Given an adequate network size, the parameter of each connection can be adjusted to enable the network's approximation to the desired function. This is accomplished through an optimization procedure called gradient descent [19].

The concept of loss is a central part of the training process. Loss is a function of both the network's output (referred to as prediction) and the desired output (referred to as target). The loss function quantifies the distance, or dissimilarity, between the network's prediction and the desired target. The training process seeks to decrease this distance [22]. The idea is that decreasing loss minimizes the dissimilarity between the prediction and the target. In other words, this process moves the predictions towards the target.

The selection of an appropriate loss function is important and depends on the network's task. In instances involving regression tasks, commonly chosen loss functions are mean absolute errors and mean square errors, also known as L1 and L2 loss, respectively. In classification tasks, where the network is tasked to correctly predict the predefined category of the input data, the cross-entropy loss is widely chosen.

Gradient Descent

Within the context of deep learning, gradient descent is the optimization procedure where the network parameters are incrementally adjusted to minimize the loss. Following a prediction given by the forward pass of input data, the loss is calculated. Subsequently, the gradient of the loss is calculated, which is a vector containing the partial derivatives of the loss with respect to all tunable parameters of the network.

With the calculated gradient, every parameter of the network is adjusted in the opposite direction of the gradients. This updates the parameters in the direction that reduces the loss whence the gradient is computed. The scalar value of this update is determined not only by the gradient values, but also by a small factor known as the learning rate [22], [23].

How the gradient descent is performed is dictated by the optimization algorithm, simply known as the optimizer. An optimizer determines how the gradient is employed for gradient descent. It also integrates various techniques, such as adaptive adjustments to the learning rate. A popular optimizer is Adam (adaptive momentum), notable for its incorporation of a momentum term in order to adaptively adjust the learning rate based on the gradients calculated in the preceding iterations during the gradient descent [24].

Overfitting

Overfitting is a well-known phenomenon in machine learning, where the trained model performs well on the training data yet underperforms on new, unseen data. The goal of machine learning is to train a model with the capacity to generalize well, not the capacity to perfectly recall the data that is seen during training [25]. Two common causes of this phenomenon are insufficient data and model complexity. The first cause, insufficient data, refers to the lack of a sufficient number of distinct data samples in the training data which the model is trained with. This deficiency can cause the model to excessively conform to the data seen in training. The second cause, model complexity, is a common cause of overfitting for neural networks. With a large number of trainable parameters and thereby the ability to achieve a more complex model, a neural network is more prone to overfitting, particularly when trained upon data lacking the same level of complexity [26].

To detect overfitting, a common practice is to reserve a portion of the dataset exempted from training. The dataset reserved from training is used to evaluate the trained model's performance on unseen data. This reserved dataset is often further divided into two subsets; validation and test set. The validation set has two roles. It serves to evaluate the model's performance on data unseen during training. Additionally, it serves to evaluate and select the model's non-trainable parameters (hyperparameters) [27]. The test set is typically employed in the final evaluation to measure how the model, that is selected based on its performance on the validation set, performs on a completely unseen dataset. The three subsets, training, validation, and test set, constitute the development dataset [28].

2.4.2 Convolutional Neural Network

One of the most successful applications of deep learning is computer vision. Much of its success owes credit to the convolutional neural network (CNN), which started to achieve remarkably good results in image classification competitions in the period between 2011 and 2015. Today, CNNs are ubiquitous in the field of computer vision [29].

While an MLP can be applied to computer vision tasks, it has some significant drawbacks. MLP treats each input (pixel) indifferently regardless of their adjacency to one another. In other words, it disregards the spatial structure of an image. Additionally, the input layer dimension must match the input image dimension, which

increases the number of parameters that extends the training time and demands greater memory [30].

CNN addresses these drawbacks through convolution operations and shared filter weights. The operation performed in a convolutional layer of a CNN is a calculation of dot products with a sliding convolution filter over the spatial dimensions of the input. Each step calculates a dot product between the filter weights and a section of input. This dot product is placed in the corresponding position in the resulting output and represents multiple adjacent pixels from the input. The output from a convolutional layer is commonly called a feature map, implying that this output contains features extracted from the input image or an input feature map produced by a preceding convolutional layer.

The application of the same filter across the entirety of the layer input gives the CNN a property known as *translational invariance* . This property enables the detection of relevant features regardless of their position in the image. Furthermore, shared weights across the spatial input dimension reduce the number of parameters for training. In practical instances, the spatial dimensions of the filters are significantly smaller than those of the input features, giving a significantly smaller number of trainable parameters compared to a fully connected layer [31].

The feature map derived from a convolutional layer is aggregated through a process called pooling, which is performed by the pooling layer. This operation reduces the spatial size of the feature map, giving a coarser feature map compared to the input. The pooling layer has no parameters, as it is a fixed operation. The common pooling operations are max pooling and average pooling.

2.5 Applications of CNN in Computer Vision

2.5.1 Image Classification

Image classification is a quintessential computer vision task. This task aims to predict the category, or the class, of the object in the input image. A classic example of an image classification task is the recognition of handwritten digits, where there are ten known classes (numbers zero to nine) [32].

Deep learning models designed for image classification are typically composed of convolutional layers and dense layers. The convolutional layers function as feature extractors, obtaining relevant information within the input image. The extracted feature map is subsequently passed to the dense layers, which conduct the classification. The number of neurons in the final dense layer reflects the number of known classes in the task, where each output neuron represents one of the predefined classes. The classification of the object within the input image belongs to the class represented by whichever node that gives the highest output value [31].

2.5.2 Object Detection

Object detection is another field within the field of computer vision. The goal of object detection is to detect all instances of objects from one or several predefined classes in an image. Each detection produces some form of pose, which tells about the location and scale of the object within the image [33].

The pose of an object within an image is commonly described through a bounding box. A bounding box of a rectangular shape can be defined by four parameters: x, y, w and h. Parameters x and y represent the position of the bounding box within the image, with the reference point being the center or one of the corners of the box. Meanwhile, parameters w and h represent the width and height of the bounding box [34].

Given the goal of object detection, a prediction of a model consists of both classification and pose of the detected object. In a simplified scenario with a single object that may belong to one of K different predefined classes, the prediction can be expressed with Eq. 2.10

$$pred = \begin{bmatrix} c_0 \\ c_1 \\ \vdots \\ c_K \\ x \\ y \\ w \\ h \end{bmatrix} \tag{2.10}$$

Where c_i is the prediction score for object belonging to class i [35].

Intersection over Union

Intersection over union (IoU) is a metric used for evaluating predicted bounding boxes and is widely recognized as the de facto evaluation metric for object detection [36]. It quantifies the ratio of the overlapping area between the target and predicted bounding box (intersection), and the area covered by the prediction and target bounding box (union) [37].

$$IoU = \frac{|A \cap B|}{|A \cup B|} \tag{2.11}$$

IoU is a metric that considers both the position and size of the predicted bounding box when comparing it to the target bounding box. Its value ranges from zero to one, with a value of one indicating perfect overlap and a diminishing value indicating decreasing overlap [38].

Generalized Intersection over Union

The generalized intersection of Union (GIoU) serves as both a performance metric and a loss function. Introduced by Rezatofighi et al. in 2019, GIoU addresses issues with traditional loss functions like L1 or L2 for training detection models [36]. The first issue is the lack of scale invariance in said losses. Variation in image size affects the loss value, even when the relative size and position of bounding boxes remain the same. The second problem is that there is a disconnect between the loss function and performance metric. Different predictions with varying IoU scores can give identical loss values. From the perspective of loss, such predictions are equally good or poor predictions despite the IoU scores telling otherwise, as shown in Figure 2.12 from their paper [36].

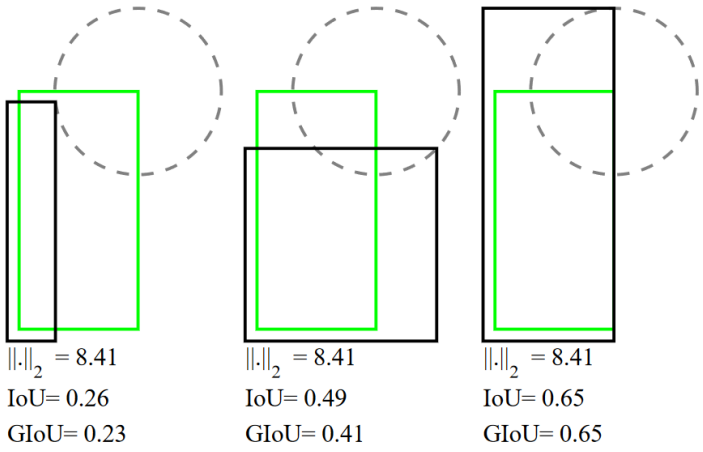


Figure 2.12: Different IoU scores for predictions with equal L2 loss (denoted $||.||_2$).

For this reason, it is desirable to use IoU as a loss function. However, a major problem prevents the employment of IoU as a loss function. In case of no overlap between the predicted and target bounding box, the resulting IoU is zero. This results in zero gradients, which cannot be optimized [36].

GIoU addresses this issue by producing a negative, non-zero value in cases with no overlap between the target and predicted bounding boxes. The distance between the target and the prediction increases the magnitude of the loss, providing a measure of how inaccurate the predicted bounding box is in terms of position. This enables the use of IoU as both a performance metric and a loss function[36].

The calculation of GIoU between two bounding boxes A and B is described in Algorithm 1 [36].

Algorithm 1 Generalized Intersection over Union

Input: Two arbitrary convex shapes: $A, B \subseteq \mathbb{S} \in \mathbb{R}^n$

Output: $GIoU$

1: For A and B, find the smallest enclosing convex object C, where $C \subseteq \mathbb{S} \in \mathbb{R}^n$

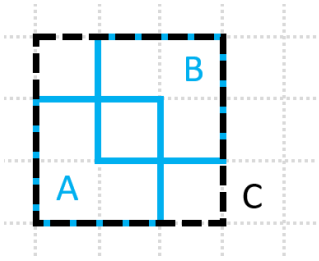
2: $IoU = \frac{|A \cap B|}{|A \cup B|}$

3: $GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|}$

A loss function, derived from GIoU is given by in Eq. 2.12

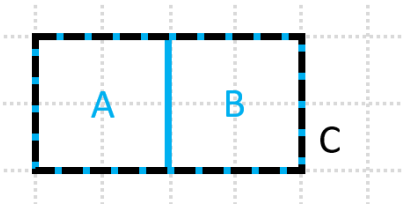
$$\mathcal{L}_{GIoU} = 1 - GIoU \tag{2.12}$$

The Figures 2.14 to 2.16, and accompanying calculations show examples of GIoU and GIoU loss for different bounding box pairs. The area of a grid cell in the illustrations is equal to one. C is the smallest enclosing convex object involved in the algorithm.



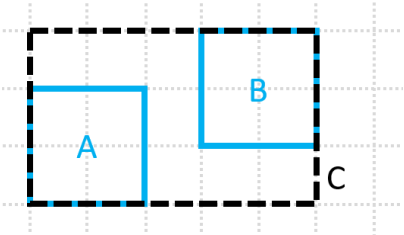
$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{1}{7}$$
$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|} = \frac{1}{7} - \frac{2}{9} = -\frac{5}{63}$$
$$\mathcal{L}_{GIoU} = 1 - GIoU = 1 - (-\frac{5}{63}) = \frac{68}{63} \approx 1.08$$

Figure 2.13: GIoU Example 1



$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{0}{8} = 0$$
$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|} = 0 - \frac{0}{8} = 0$$
$$\mathcal{L}_{GIoU} = 1 - GIoU = 1 - 0 = 1$$

Figure 2.14: GIoU Example 2



$$IoU = \frac{|A \cap B|}{|A \cup B|} = \frac{0}{8} = 0$$
$$GIoU = IoU - \frac{|C \setminus (A \cup B)|}{|C|} = 0 - \frac{7}{15} = -\frac{7}{15}$$
$$\mathcal{L}_{GIoU} = 1 - GIoU = 1 - (-\frac{7}{15}) = \frac{22}{15} \approx 1.47$$

Figure 2.15: GIoU Example 3

2.6 Related Work

Extensive research has been conducted on the application of deep learning for PRI recognition. In 1999, Noone introduced a neural method for classifying four PRI modulation classes [8]. His approach extracts features from a pulse sequence of length 64. The feature extraction is based on using the second difference of ToA of pulses, called D^2ToAs. The vector of D^2ToAs is used to form a signum vector through a modified signum function, which supposedly gives distinct vector for each modulation type. The architecture for the neural network was a multilayer perceptron with three layers. Based on the simulation, this approach is shown to be both robust and reliable even when classification is conducted on a rather short pulse sequence of only 64 pulses [8].

Building upon Noone's feature extraction, Liu and Zhang proposed an improved PRI modulation classification in 2017 [39]. Based on Noone's approach, their approach extracts three features; stationarity, monotonic directionality, and symmetry. These three features were passed to a three-layer MLP for classification. This approach improved reliability and accuracy, and reduced computational complexity by limiting the number of input features to three [39].

In 2018, Nguyen et al. [6] introduced an approach using deep learning on features extracted with autocorrelation and symbolization, which are methods proposed by Ahmadi et al. and Song et al., respectively [40], [41]. Subsequently, in 2019, they proposed another approach employing CNN on raw PRI sequences, yielding promising results while also eliminating the need for preprocessing and feature extraction [42]

The Swedish aerospace and defense company Saab has hosted several theses exploring deep learning methods for PRI modulation classification. In 2019, Norgren compared Long Short-Term Memory (LSTM) against Liu and Zhang's method, revealing that LSTM consistently outperformed the latter, albeit with increased complexity [14]. Another thesis in 2022 by Svensson compared an attention-based model (transformer) against a k-nearest neighbor (KNN) model. The attention-based model demonstrated robust performance across realistic noise levels, while the KNN model's performance declined with rising noise levels [43].

The work of Barrios in 2021, also hosted by Saab, stands out as the closest related work [9]. In her thesis, the PRI modulation classification is formulated as an image classification problem. A PRI_i, PRI_{i+1} image is created from a simulated PRI sequence with various levels of noise. The images depict the change of PRI values in the sequence. The images are used as training data for a convolutional neural network for image classification that classifies the images into one of the modulation types [9].

The novelty of my thesis lies in the exploration of what the host institute of the thesis, Forsvarets Forskningsinstitutt (FFI), considers as detailed PRI classification, which is classifying a particular PRI sequence pattern into finer categories than the six categories typically used for modulation classifications. For instance, patterns derived from stagger or dwell and switch modulations exist in many variants. Furthermore,

a radar can operate with a composite of modulation types, as evident in the provided data for this thesis. These factors give diversity to PRI patterns that extend beyond the descriptions of PRI modulation types. Classification on a subset of patterns from this diversity is what FFI considers as detailed classification. The ability to distinguish pulse sequences from radars based on such finer detail is beneficial, as they can be more closely attributed to a specific radar, thus narrowing down the search field during the emitter identification process. This can eventually contribute to more robust emitter recognition, and tracking them over time.

Chapter 3

Method

This chapter describes the provided data, the data processing steps taken, and the implementation of the deep learning methods employed to answer the research question.

The data provided by FFI served to train and evaluate the deep learning models. The provided data were sequences of PRI values and PRI_i, PRI_{i+1} images generated from the sequences, referred to as PRI sequences and PRI images, respectively. Processing and labeling the images were necessary to render them suitable for training and evaluation.

Two types of deep learning models were implemented to fulfill the objective of the thesis. The first model, an object detection model, was designed to detect and localize a PRI pattern within an image for subsequent classification. The objective of this model was to automate the manual work during the analysis process, which involves detecting and focusing in on a PRI pattern. The prediction from detection, which contains the position and size of the pattern within the image, could then be translated into parameters for generating a new PRI image from the PRI sequence. Generating a new image is necessary if the detected pattern has insufficient size for classification. In such a case, a new image is generated, ideally containing a pattern represented with optimal size for classification.

The second deep learning model was an image classifier, classifying the PRI pattern class depicted in the image. This model seeks to automate the manual classification task.

In contrast to the conventional object detection task, where the prediction contains both the classification and pose of the detected object [33], a decision was made to split these tasks into separate models. The decision is based on recognizing that the pattern to be classified might be suboptimally represented in the image. Therefore, the task was split into two stages, ensuring that the classification model received images optimally representing the pattern. Pipelining the two models seeks to automate the manual human process taking place during an analysis process, involving manually focusing in on a pattern and classifying the pattern type.

3.1 Data

FFI has provided two types of data: PRI sequences and PRI images. These data originated from radar pulses emitted by real shipborne navigational radars and were collected in Oslofjord. The original data containing ToA sequences and other parameters were not provided. Only PRI sequences generated from these ToA sequences (and PRI images) were given.

Data from a total of 65 unique pulse sequences was used, corresponding to an equivalent number of 65 unique PRI sequences that were provided. Each PRI sequence forms a PRI pattern associated with one of the 14 pattern classes. Notably, these 14 classes are among the most common navigation radar PRI patterns observed by FFI (although not exhaustive), and represented a more finer division than the general PRI modulations. The nomenclature for each pattern class is assigned based on the visual representation they give in the resulting PRI image. The distribution of the classes is illustrated in Table 3.1.

Pattern Class	**PRI Modulation**	**Number of Unique PRI sequences**
AntiDiag	Stagger	4
Const	Constant	5
LineDown	Stagger and Sliding	5
LineUp	Stagger and Sliding	5
LineUp+	Undetermined	5
LongLines	Undetermined	5
Stg2	Stagger	5
Stg9	Stagger	5
Strange	Undetermined	5
Three5Lines	Stagger and Sliding	3
Three8Lines	Stagger and Sliding	3
TwoLines	Stagger and Sliding	5
TwoStairs	Undetermined	5
X	Stagger or Discrete Jitter	5

Table 3.1: PRI pattern classes

3.1.1 PRI Sequences

PRI sequences were the first type of data provided by FFI. For each of the 65 unique pulse sequences, a list of PRIs was generated, forming the provided PRI sequences

The original pulse sequence was presented in a Scan Description Word (SDW) format, where radar pulses were organized into scans. A scan represents the duration during which the receiving sensor is within the directivity of the emitting radar's antenna, enabling the sensor to receive the emitted pulses. Conversely, when the sensor is outside of the scan, the radar continues to emit, but the sensor can no longer receive.

The SDW data format contains descriptions of up to 20 radar pulses from a single scan. The provided PRI sequence was generated by calculating the difference in ToA between successive pulses within each scan. The difference between the last pulse in a scan and the first pulse in the next scan was not included in the PRI sequence, as they were disjoint. For illustrative purposes, Table 3.2 shows an example of how a PRI sequence is derived from SDWs.

$Scan\ 1$	ToA_1^1	$PRI_1^1 = ToA_2^1 - ToA_1^1$
	ToA_2^1	$PRI_2^1 = ToA_3^1 - ToA_2^1$
	$\vdots$	$\vdots$
	ToA_i^1	$PRI_i^1 = ToA_{i+1}^1 - ToA_i^1$
	$\vdots$	$\vdots$
	$ToA_{p^1}^1$	$PRI_{p^1-1}^1 = ToA_{p^1}^1 - ToA_{p^1-1}^1$
$Scan\ 2$	ToA_1^2	$PRI_1^2 = ToA_2^2 - ToA_1^2$
	ToA_2^2	$PRI_2^2 = ToA_3^2 - ToA_2^2$
	$\vdots$	$\vdots$
	ToA_i^2	$PRI_i^2 = ToA_{i+1}^2 - ToA_i^2$
	$\vdots$	$\vdots$
	$ToA_{p^2}^2$	$PRI_{p^2-1}^2 = ToA_{p^2}^2 - ToA_{p^2-1}^2$
$Scan\ s$	ToA_1^s	$PRI_1^s = ToA_2^s - ToA_1^s$
	ToA_2^s	$PRI_2^s = ToA_3^s - ToA_2^s$
	$\vdots$	$\vdots$
	ToA_i^s	$PRI_i^s = ToA_{i+1}^s - ToA_i^s$
	$\vdots$	$\vdots$
	$ToA_{p^s}^s$	$PRI_{p^s-1}^s = ToA_{p^s}^s - ToA_{p^s-1}^s$

Table 3.2: Calculation of PRI from ToAs grouped in scans

The superscript denotes the index of the scan, while the subscript denotes the index within the scan.
ToA_i^s denotes ToA of i-th pulse in the s-th scan.
PRI_i^s denotes the i-th calculated PRI in the s-th scan.

p^s denotes the number of pulses in the scan s. Note that this number is limited to 20. In scan s, containing p^s pulses, $ToA^s_{p^s}$ denotes the ToA of the last pulse in the scan. The calculated PRI values were gathered in a file, with separation between every scan. This collection of PRI values constitutes a single PRI sequence.

3.1.2 PRI Images

PRI images were the second category of data provided by FFI. Images were generated from PRI sequences. They served to visually represent the variations in PRI values within a PRI sequence. It is noteworthy that the pattern cannot always determine the PRI modulation. While the provided PRI images selected from 14 different PRI patterns all have different appearances, it may, for instance, be impossible to determine whether an image displays a discrete jitter modulation or a stagger modulation with a long cyclicity. This highlights that most of the temporal aspect of the PRI sequence is lost in these images.

A simple description of image generation is to consider a PRI image as a scatterplot. Each data point characterizes the difference between two successive PRI values in the PRI sequence. To elaborate further, a data point with an index i is plotted at a position within the image given by x_i and y_i expressed with Eq. 3.1 and 3.2.

$$x_i = PRI_i \tag{3.1}$$

$$y_i = PRI_{i+1} \tag{3.2}$$

A table with data points for scatter plotting can be generated from a PRI sequence of length n, shown by 3.3. Columns of the table represent the position where the data point is plotted.

x-axis	**y-axis**
PRI_1	PRI_2
PRI_2	PRI_3
$\vdots$	$\vdots$
PRI_i	PRI_{i+1}
$\vdots$	$\vdots$
PRI_{n-1}	PRI_n

Table 3.3: PRI image data points

It is noteworthy that the above table assumes that there is no disconnect between all successive PRI values, which is not the case for the provided PRI sequence derived from SDW. Therefore, PRI values at the edges of each scan do not form a data point with the PRI at the edge of the adjacent scan.

For illustrative purposes, Table 3.4 shows how data points are derived from a PRI sequence, which considers the disconnectivity between scans. Definitions of PRI_i^s and p^s remain unchanged from subsection 3.1.1 x_i^s, y_i^s denote the i-th data point derived from within s-th scan.

Scan 1	PRI_1^1	$x_1^1, y_1^1 = (PRI_1^1, PRI_2^1)$
	PRI_2^1	$x_2^1, y_2^1 = (PRI_2^1, PRI_3^1)$
	$\vdots$	$\vdots$
	PRI_i^1	$x_i^1, y_i^1 = (PRI_i^1, PRI_{i+1}^1)$
	$\vdots$	$\vdots$
	$PRI_{p^1-1}^1$	$x_{p^1-2}^1, y_{p^1-2}^1 = (PRI_{p^1-2}^1, PRI_{p^1-1}^1)$
Scan 2	PRI_1^2	$x_1^2, y_1^2 = (PRI_1^2, PRI_2^2)$
	PRI_2^2	$x_2^2, y_2^2 = (PRI_2^2, PRI_3^2)$
	$\vdots$	$\vdots$
	PRI_i^2	$x_i^2, y_i^2 = (PRI_i^2, PRI_{i+1}^2)$
	$\vdots$	$\vdots$
	$PRI_{p^2-1}^2$	$x_{p^2-2}^2, y_{p^2-2}^2 = (PRI_{p^2-2}^2, PRI_{p^2-1}^2)$
Scan s	PRI_1^s	$x_1^s, y_1^s = (PRI_1^s, PRI_2^s)$
	PRI_2^s	$x_2^s, y_2^s = (PRI_2^s, PRI_3^s)$
	$\vdots$	$\vdots$
	PRI_i^s	$x_i^s, y_i^s = (PRI_i^s, PRI_{i+1}^s)$
	$\vdots$	$\vdots$
	$PRI_{p^s-1}^s$	$x_{p^s-2}^s, y_{p^s-2}^s = (PRI_{p^s-2}^s, PRI_{p^s-1}^s)$

Table 3.4: Calculation of data points from scans

To illustrate the generation of a PRI image deriving all the way from the time of arrival of the radar pulses, consider an example of a radar pulse sequence with the following time of arrival. It is assumed that they belong to the same scan.

$$\textbf{ToA} = [0, 1, 3, 6, 10, 11, 13, 16]$$

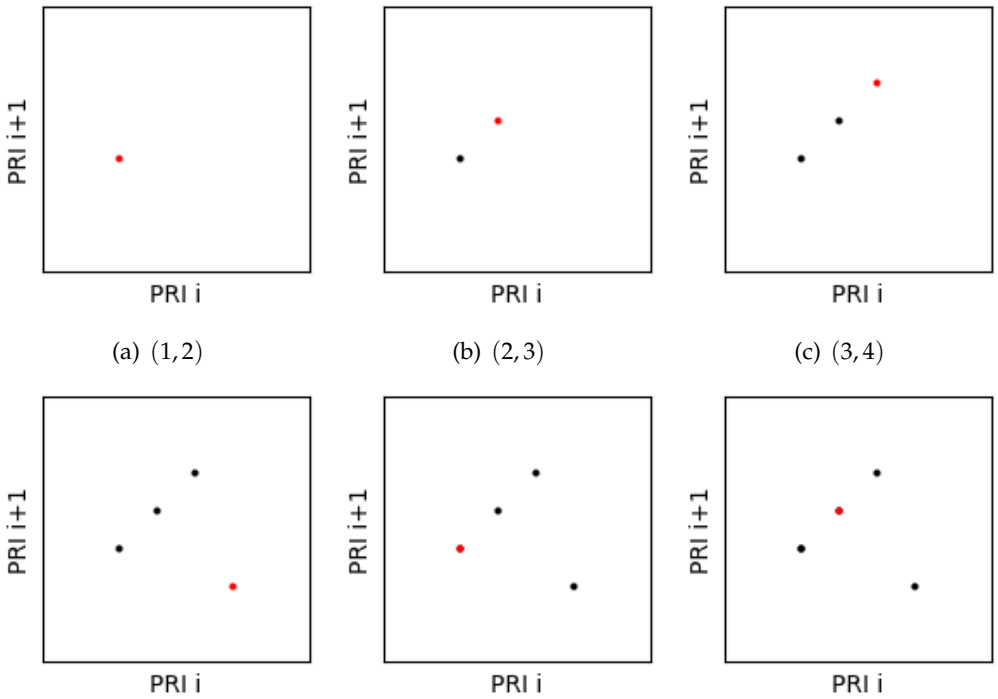
From these ToAs, PRI sequence is calculated.

$$\textbf{PRI} = [1, 2, 3, 4, 1, 2, 3]$$

Finally, data points are composed from the calculated PRIs.

$$(\mathbf{x}, \mathbf{y}) = [(1, 2), (2, 3), (3, 4), (4, 1), (1, 2), (2, 3)]$$

Data points are plotted one by one, as illustrated in Figure 3.1

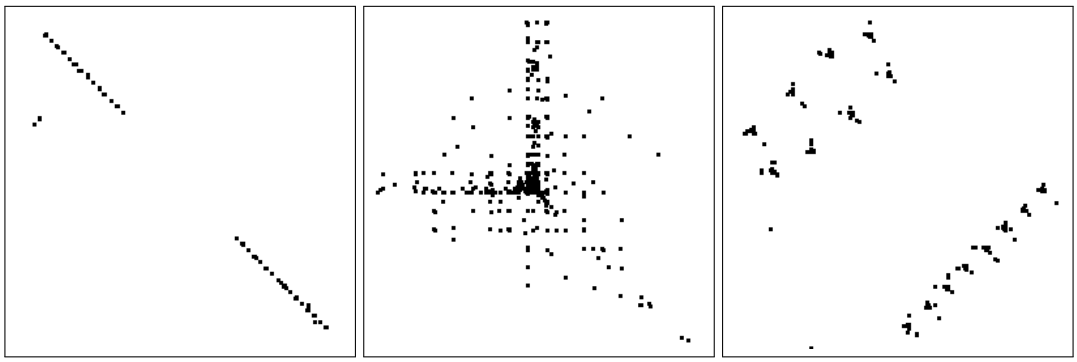


(a) $(1, 2)$ (b) $(2, 3)$ (c) $(3, 4)$

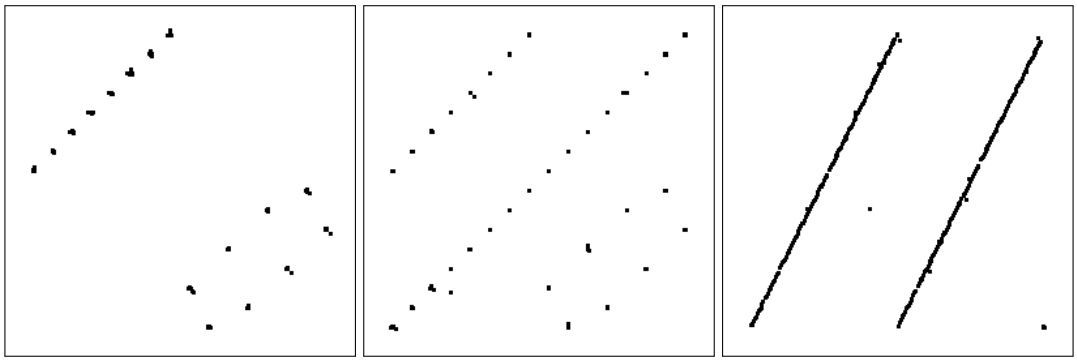
(d) $(4, 1)$ (e) $(1, 2)$ (f) $(2, 3)$

Figure 3.1: Plotting calculated data points

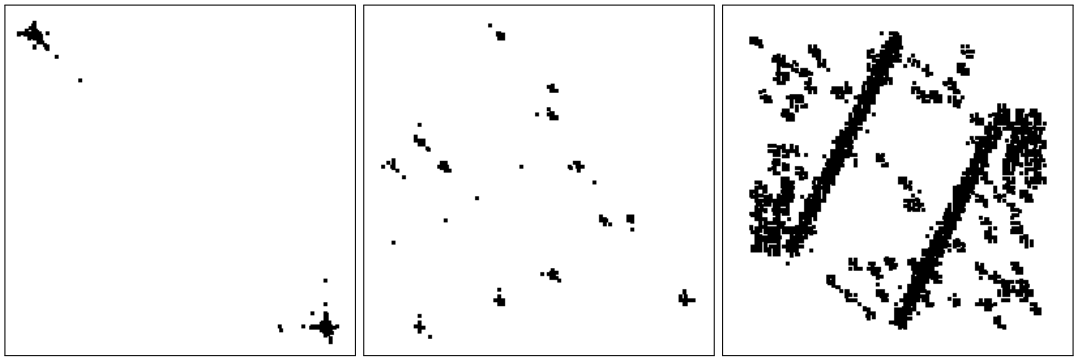
The figures in Figure 3.2 are PRI images of each pattern class provided by FFI. All PRI images are depicted with x-axis representing PRI_i and y-axis representing PRI_{i+1}. This applies for all PRI images for the remainder of the document.



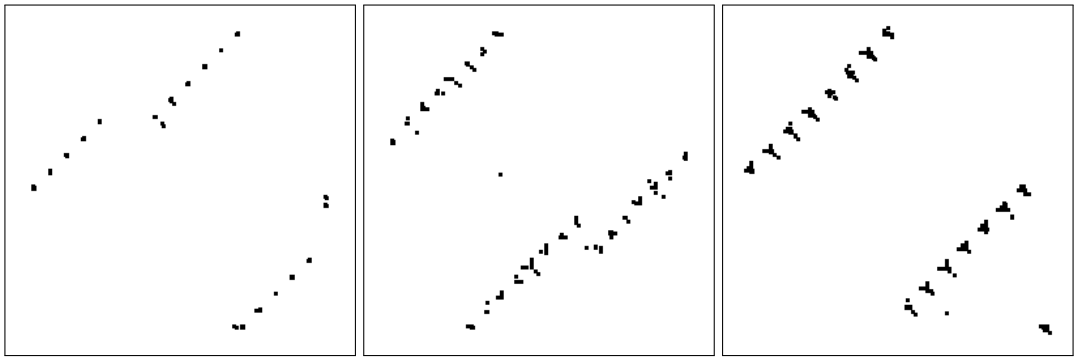
(a) AntiDiag (b) Const (c) LineDn



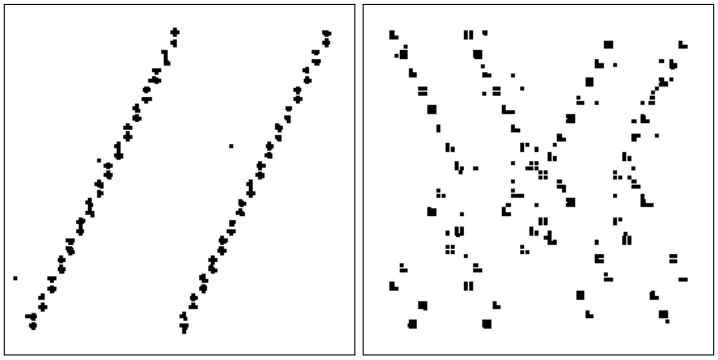
(d) LineUp (e) LineUp+ (f) LongLines



(g) Stg2 (h) Stg9 (i) Strange



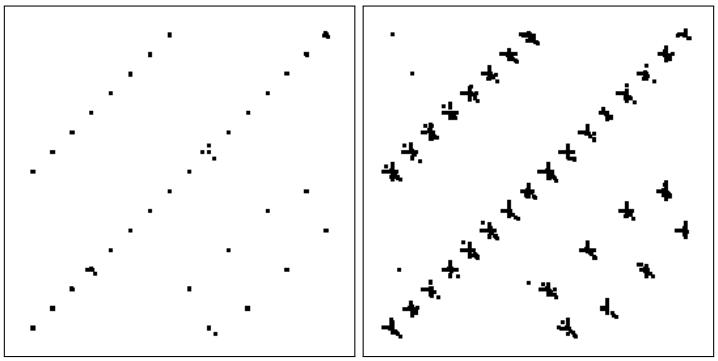
(j) Three5Lines (k) Three8Lines (l) TwoLines



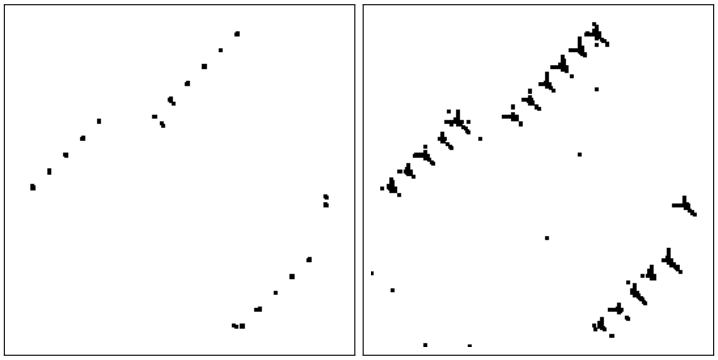
(m) TwoStairs (n) X

Figure 3.2: PRI Image of each class

Some of the provided PRI sequences contain more PRI noise than others, which is visible among the provided images. While the overall structure of the pattern remains consistent, the impact of noise is visible in the scattered data points around the points that constitute the pattern's structure. The figures show patterns LineUp+ and ThreeLines with different amounts of noise.



(a) LineUp+, few noise (b) Line Up+, more noise



(c) Three5Lines, few noise (d) Three5Lines, more noise

Figure 3.3: PRI pattern with low noise (a, c) and PRI pattern with higher noise (b,d)

An important parameter when generating the image is the *image PRI range*. Figure 3.4 visualizes the image PRI range, defined by minimum and maximum image PRI. It also visualizes minimum and maximum pattern PRI, constituting the pattern PRI range.

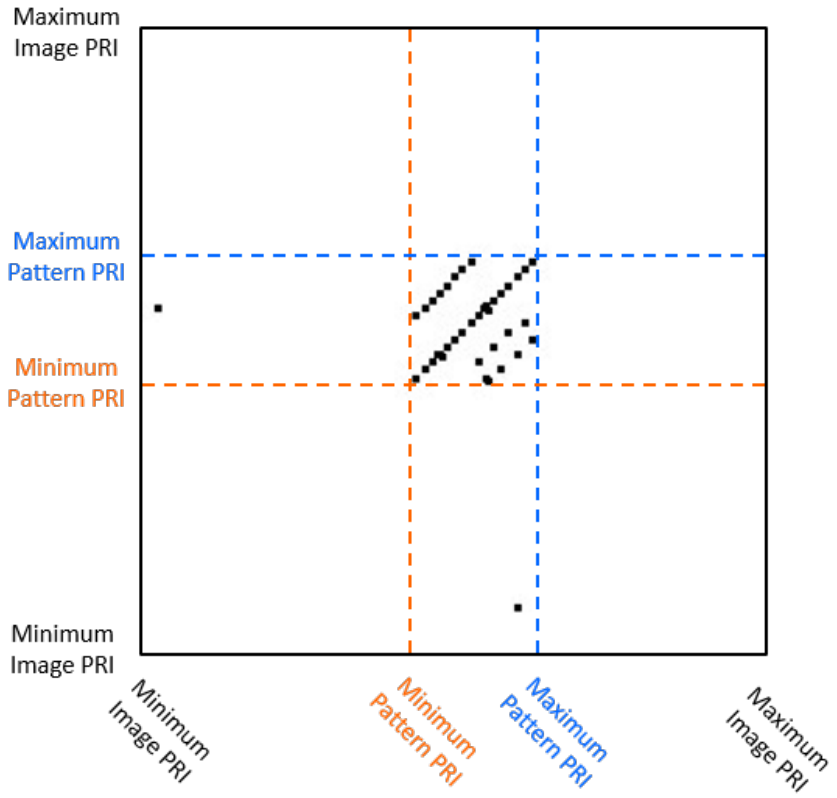
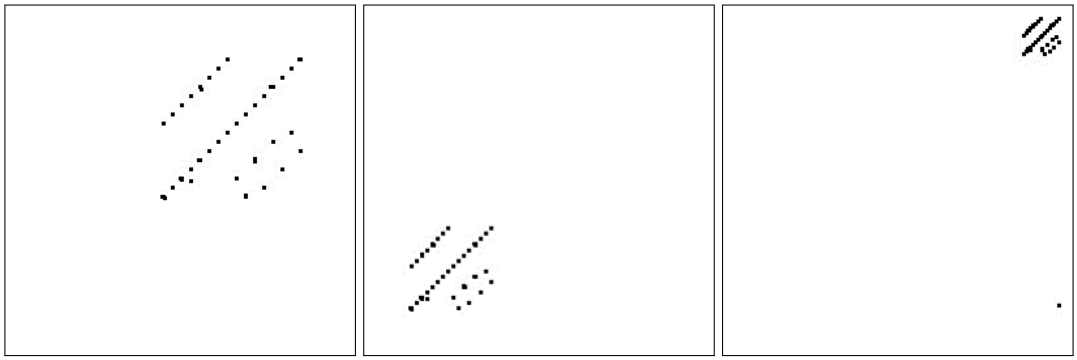


Figure 3.4: Minimum and maximum image PRI (Image PRI range), and minimum and maximum pattern PRI (Pattern PRI range).

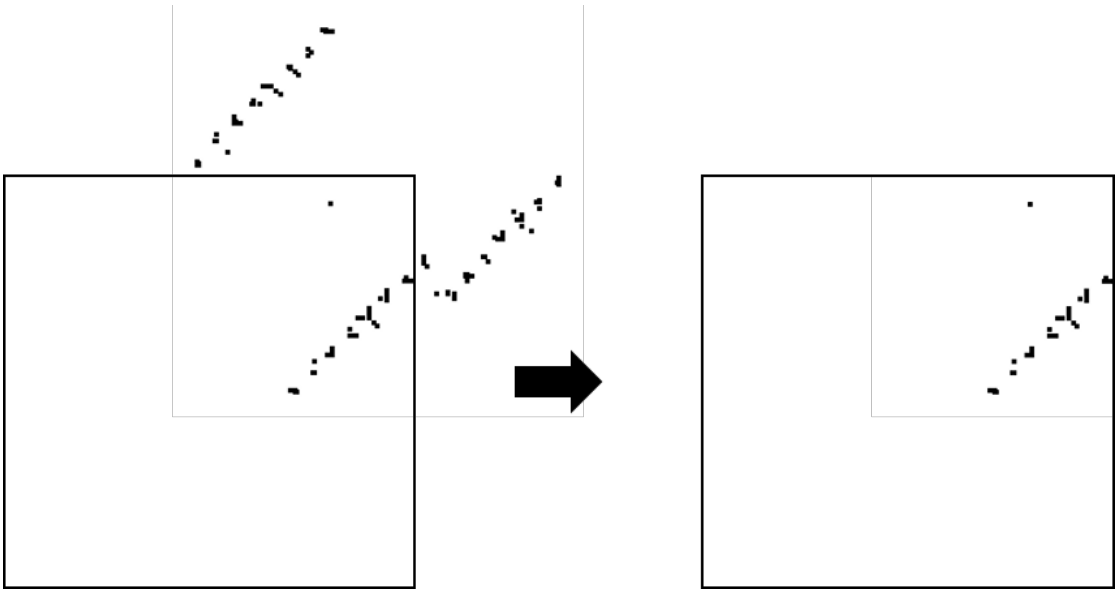
Image PRI range dictates the position and size of the pattern within the image. The examples in Figure 3.5 illustrate how varying image PRI ranges result in different positions and sizes for the pattern from the same PRI sequence. The narrower range gives a larger pattern size, while the wider range gives a smaller pattern size. The images in Figure 3.2 illustrating the 14 classes, the image PRI range was manually adjusted to fit each PRI pattern perfectly.



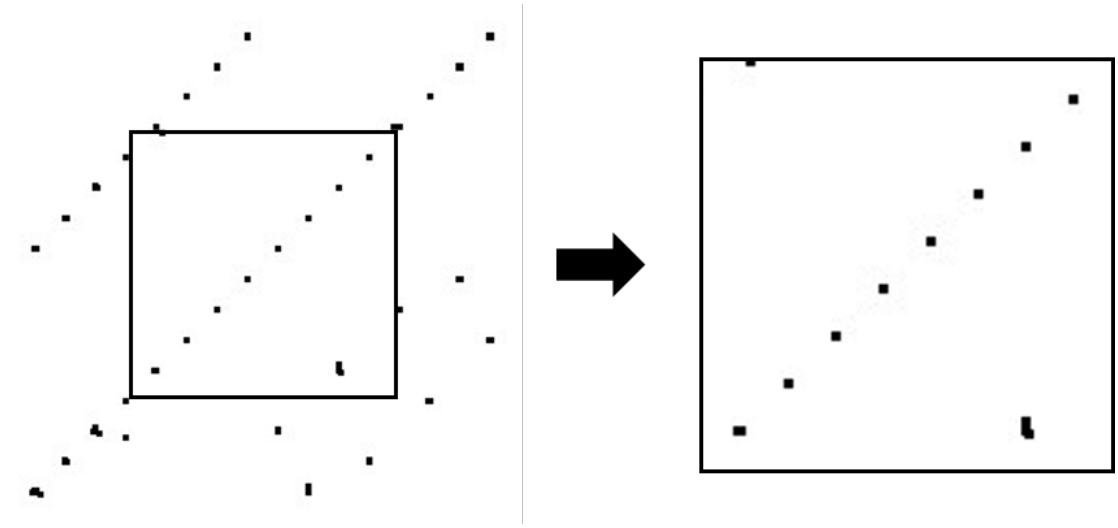
(a) LineUp+, Image PRI range 1 (b) LineUp+, Image PRI range 2 (c) LineUp+, Image PRI range 3

Figure 3.5: Pattern LineUp+ with three different image PRI Ranges

It is important that the image PRI range covers both the minimum and maximum PRI values that form the pattern. Failure to do so may lead to the exclusion of a portion of the pattern in the resulting image, which can potentially impact the subsequent classification task.



(a) Three8Lines with misplaced image PRI range



(b) LineUp+ with too narrow image PRI range

Figure 3.6: Pattern Three8Lines and LineUp+ with erroneous image PRI range

Note that in Table 3.3, every PRI value, excluding the first and the last, is employed as both the x-axis and y-axis positions for the scatterplot. This gives two important properties to the PRI image. The first property is that the center point of the resulting pattern lies along the diagonal axis (from lower left to upper right corner) of the image. The second is that the horizontal and vertical extents of the pattern are identical.

These two properties allow practical simplifications during the implementation of the detection model. First, the model needs to predict only one value for the position. The center of the pattern is along the diagonal axis, which means that the x-position is equal to the y-position. Secondly, the model only needs to predict one value for the size of the pattern, as the equal horizontal and vertical extent allows the use of a square-shaped bounding box. The size of the square is described with a single value, representing the common size of the sides of the square.

The PRI images provided by FFI are created from the 65 unique PRI sequences described in 3.1.1. For every unique sequence, a set of 101 images was generated, each with a different image PRI range. Each image PRI range covers the complete pattern and ensures that no part of the pattern is excluded. The purpose of having this diversity was to emulate that image PRI range will be unknown. For new data, the visual inspection starts from a large image PRI range. This range is narrowed down, possibly generating new PRI images that are better suited to the PRI range of the pattern. To facilitate learning this process, variation in position and size of the PRI pattern in the training set was required.

Each PRI image was labeled with the following labels in the image's filename:

- Pattern's detailed class name
- Minimum Pattern PRI
- Maximum Pattern PRI
- Minimum Image PRI
- Maximum Image PRI

The provided images were not directly employed in the implementation. However, the image PRI ranges used for generating the provided images were repurposed for generating images with the different method, as described in the next section.

3.2 Image Processing

3.2.1 Generating PRI images as heatmap

The first step in the processing addresses the noise. Each point in the provided image is visualized indifferently, implying that numerous data points concentrated in one position had the same intensity as a single data point in another position. In other words, a single noise value had the same intensity as each element that constitutes the relevant PRI pattern. Figure 3.7 shows a sample of AntiDiag pattern affected by noise. The image PRI range of this particular PRI image is wide and includes noise values far outside the pattern.

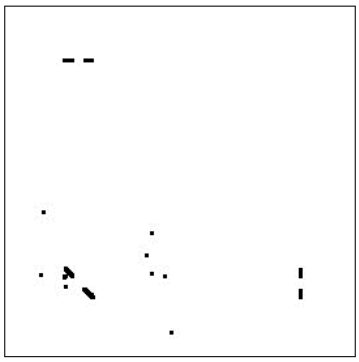
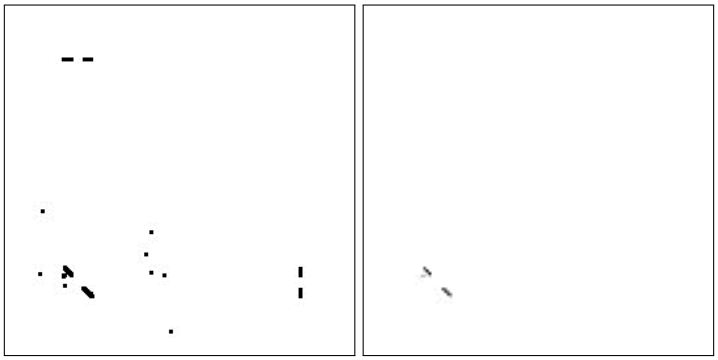


Figure 3.7: AntiDiag with noise

To address this issue, PRI images were generated anew as heatmaps, also known as 2D histograms. This method resembles a scatterplot, utilizing data points with the same x and y positions. However, a key difference was that the value of the position was incremented for each data point. This was different from the PRI images provided by FFI, where the position value was set to a fixed value and not further increased when the data point is repeated. The new method enabled the visualization of occurrence volume, allowing viewers to differentiate between relevant data points and noise with relatively fewer occurrences.

The produced heatmap was normalized by dividing it by the maximum value in the heatmap. Given the method of heatmap generation, the noise values may persist as very small numbers after normalization. Therefore, pixel values of the images were thresholded to suppress noise values to zero. The threshold was set to 0.25, meaning any pixel values below 0.25 were reduced to zero.

The image pair in Figure 3.8 compares the image provided by FFI, and the same pattern with the same PRI range generated as a heatmap. The heatmap (b) has effectively suppressed the noise depicted in the original image (a), leaving only the relevant pattern visualized.



(a) Original image (b) Heatmap

Figure 3.8: Original image (a) and heatmap (b)

It is important to note that this method is effective when each data point belonging to the pattern is abundantly populated. However, in cases where a data point in the pattern is sparsely populated compared to others, it may potentially be suppressed to zero by this method. This could result in an image with a pattern missing that specific data point. While it is possible to adjust the threshold to include every data point, the process can be tedious. This issue was not prevalent within the scope of the provided data. However, it should be considered if this method is selected to generate images from other PRI sequences.

3.2.2 Image Augmentation

The generated heatmaps were augmented with two photometric augmentation techniques: Gaussian blurring and random noise. These augmentations generated samples with randomness and different image quality. Examples in Figure 3.9 show the results of the implemented augmentation techniques.

Gaussian blurring

Gaussian blurring is often applied to reduce noise in images and smoothen the image features. In this thesis, the Gaussian blurring was used to introduce samples representing reduced image resolution. The goal was to have the blurring operation magnify the non-zero pixels, thus giving them the appearance of a lower resolution. The effect of this varied depending on the pattern size within the image. The pattern stayed distinguishable for images where the pattern is relatively large, although each pixel is scaled up. Conversely, for images with relatively small pattern, the pattern became indistinguishable as it was reduced to a concentration of coarse pixels in a small area.

A module within PyTorch library was utilized to implement the Gaussian blurring. This module is one of many image transformation modules within the PyTorch library

[44]. Gaussian blurring with filter size three and sigma three was chosen for this augmentation.

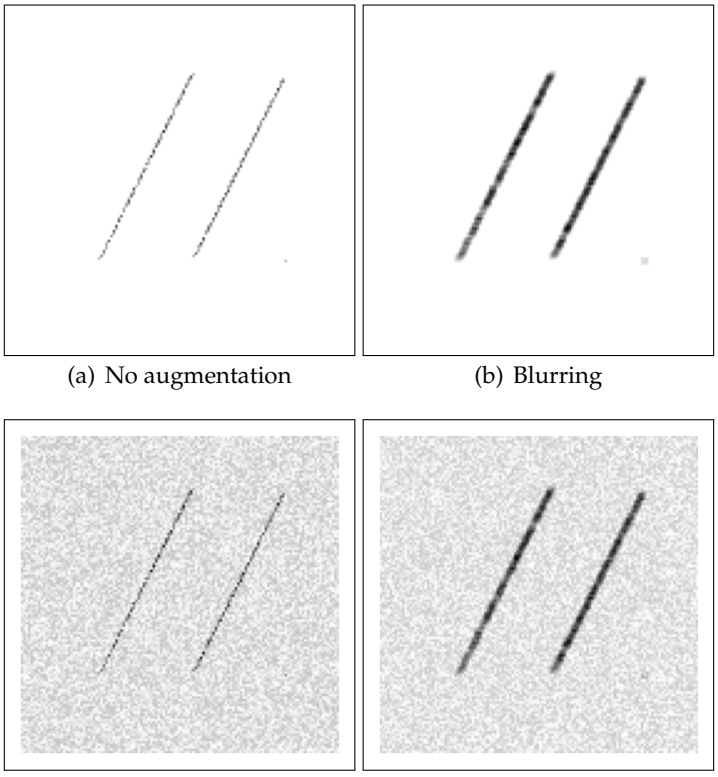
Following the blurring operation, the image was normalized by dividing it by its new maximum value post-blurring.

Background noise

The purpose of the background noise augmentation technique was to introduce samples that work as regularization and overfitting prevention [45].

During training, a randomly generated noise was added to the background every time a training image sample was loaded. The noise intensity was kept weak, preventing the pattern from drowning in the noise. The noise gave variations to the background, which the model cannot fit to, as the noise was randomly generated every time the image was loaded [45]. In the implementation, the intensity of the noise map was sampled from a uniform distribution from 0 to 0.125.

To ensure the reproducibility of the results, the noise map added to the samples in the validation and test sets remains fixed. When evaluation samples are loaded, a fixed random seed is placed, ensuring the generated noise map is the same for every evaluation instance.



(a) No augmentation (b) Blurring

(c) Background noise (d) Both blurring and background noise

Figure 3.9: Heatmap with augmentations

3.2.3 Distribution of Development Dataset

There were a total of 6565 images in the development dataset. These were distributed into training, validation, and test datasets, with a ratio of 0.7-0.1-0.2, respectively. The validation data was used to track performance on non-training data during training and to select the best model. The test dataset was used only once to test the detection and classification models.

3.3 Implementation Overview

The implementation was divided into two parts, followed by a final evaluation. The process is illustrated in Figure 3.10.

- Implementation and selection of the detection model.
  - The detection model predicts the position and size of the pattern within a PRI image. The predicted position and size can be translated to a new image PRI range for generating a new image. The new image will be generated from the predicted value if the predicted size is below a predetermined threshold. The pattern will have an optimal scale for the classification in the new image.
- Implementation and selection of classification model
  - This is a classic image classification model which predicts the class of the PRI pattern.
- Pipelining the new models and final evaluation
  - As the final evaluation of both models, unseen test data is passed through selected detection and classifier models. Depending on the size predictions of the detection model, a new image will be generated. The classifier will receive the images, whether newly generated or not, and perform the classification.

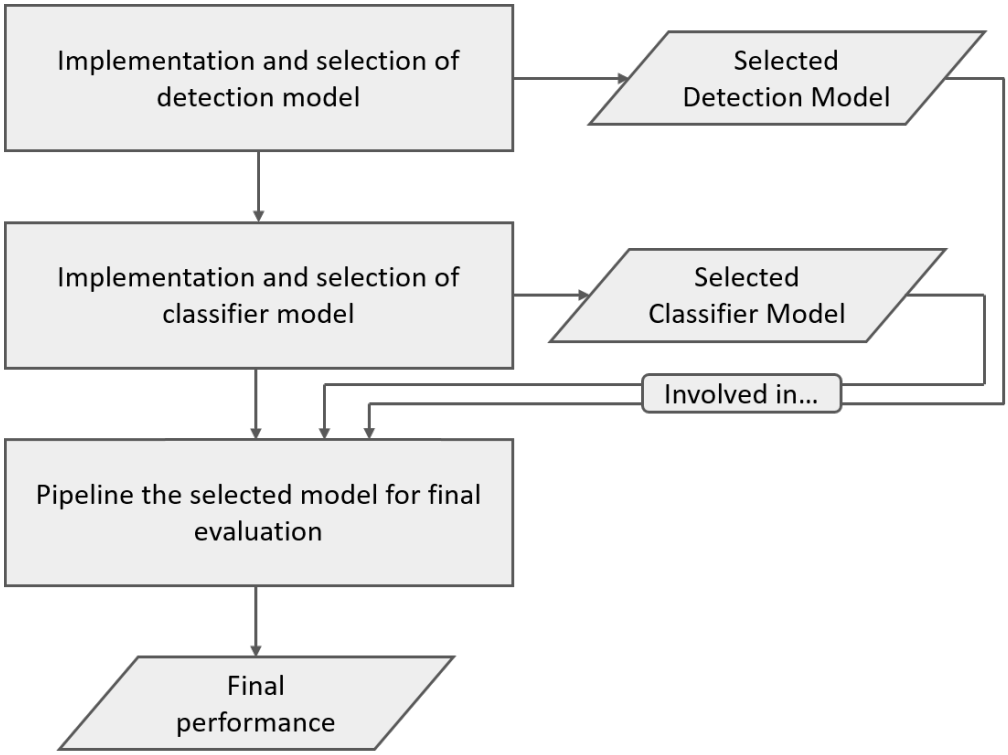


Figure 3.10: Implementation Overview

3.4 Implementation of the Detection Model

3.4.1 Setting Training Targets

Setting the Bounding Box for the Pattern

Setting the ground truth bounding box for each image was a prerequisite before employing the provided images as training data. Only two values were needed to describe the square bounding box along the diagonal axis of the image; position and size.

The position value is within the range $[0, 1]$, and describes the relative position of the pattern's center point along the diagonal axis, which is identical for both the x- and y-axis. The position can be calculated with Eq. 3.3

$$Pos = \frac{P_{min} + P_{max}}{2(I_{max} - I_{min})} - \frac{I_{min}}{I_{max} - I_{min}} \tag{3.3}$$

Where P_{min} and P_{max} denote minimum and maximum pattern PRI, respectively. Furthermore, I_{min} and I_{max} denote minimum and maximum image PRI, respectively.

The size value is also within the range $[0, 1]$, and describes the size of the side of the square bounding box. It was calculated with Eq. 3.4

$$Size = \frac{P_{max} - P_{min}}{I_{max} - I_{min}} \tag{3.4}$$

With the bounding box's calculated position and size, the bounding box's ground truth was a vector of two values as described with Eq. 3.5

$$GT = \begin{bmatrix} Pos \\ Size \end{bmatrix} \tag{3.5}$$

The vector GT in Eq. 3.5 was not employed as the training target during model training. Instead, another vector, with augmented $Size$ is employed as the target. The GT vector was involved in calculating the evaluation metric described in the subsequent subsection.

Target Size

The detection model should predict a bounding box slightly larger than the ground truth. If the predicted bounding box is smaller than the extent of the pattern, a portion of the pattern may be excluded as shown in Figure 3.6. Passing such images to the classifier is undesirable and can impact the classification.

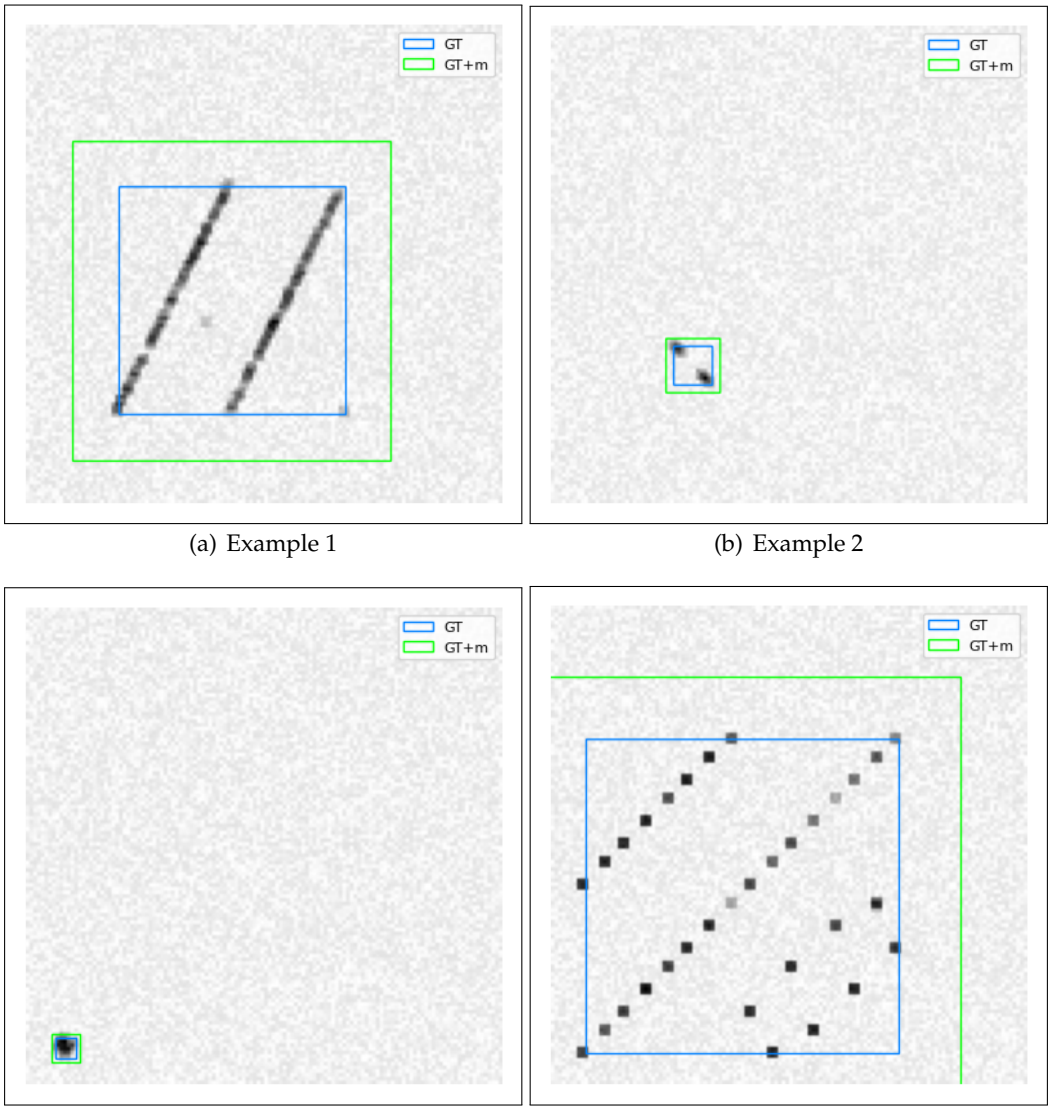
Even when the model attempts to predict the exact size of the ground truth bounding box, any slight inaccuracy can cause the prediction to be smaller than the ground truth. Additionally, even if the model correctly predicts the exact size, it relies on an equally accurate position prediction to include the entire pattern. Inaccuracies in both position and size prediction can cause a seemingly adequate prediction to not include the entire PRI pattern, ultimately producing an image that misses a portion of the pattern. It is unclear how impactful this can be for the classification task. However, it should be avoided to eliminate an avoidable factor that can affect the classification.

To avoid this, the ground truth bounding box was scaled up with a factor m. This larger bounding box was set as the training target, denoted as GT_{+m}.

The bounding box was scaled up by extending both ends of each side of the bounding box by factor m. In training, the factor m was fixed to 0.2. This implies that each side of the original bounding box was extended by 20% on both ends, resulting in a 40% increase in the size of each side.

$$GT_{+}m = \begin{bmatrix} Pos \\ Size * (1 + 2m) \end{bmatrix} \tag{3.6}$$

The following figures depict the original bounding box, denoted GT, and training target bounding box GT_{+m}



(a) Example 1 (b) Example 2

(c) Example 3 (d) Example 4

Figure 3.11: Examples of GT and GT_{+m}

The special case of the pattern class Const

Pattern class Const depicts constant PRI modulation where PRI values remain fixed. This gives $P_{min} = P_{max}$, which leads to the size of the bounding, expressed in Eq. 3.4 to be equal to zero. As a workaround, GT and GT_{+m} for Const pattern were set to 0.05.

$$GT_{constant} = \begin{bmatrix} Pos \\ 0.05 \end{bmatrix} \tag{3.7}$$

Randomness to target size

As an augmentation method, randomness has been added to the target training size during training. The randomness was added to the factor m and varies uniformly between -0.05 and 0.05. This gave factor m to vary in the range $[0.15, 0.25]$. The enlarged training target size with added randomness was denoted GT_{+m+r}

$$GT_{+m+r} = \begin{bmatrix} Pos \\ Size * (1 + 2(m + r)) \end{bmatrix} \tag{3.8}$$

$$r \sim U([-0.05, 0.05])$$

The randomness was set for Const pattern class so that the training target varies in the range [0.05, 0.10].

$$GT_{constant+r} = \begin{bmatrix} Pos \\ 0.05 + r_{constant} \end{bmatrix} \tag{3.9}$$

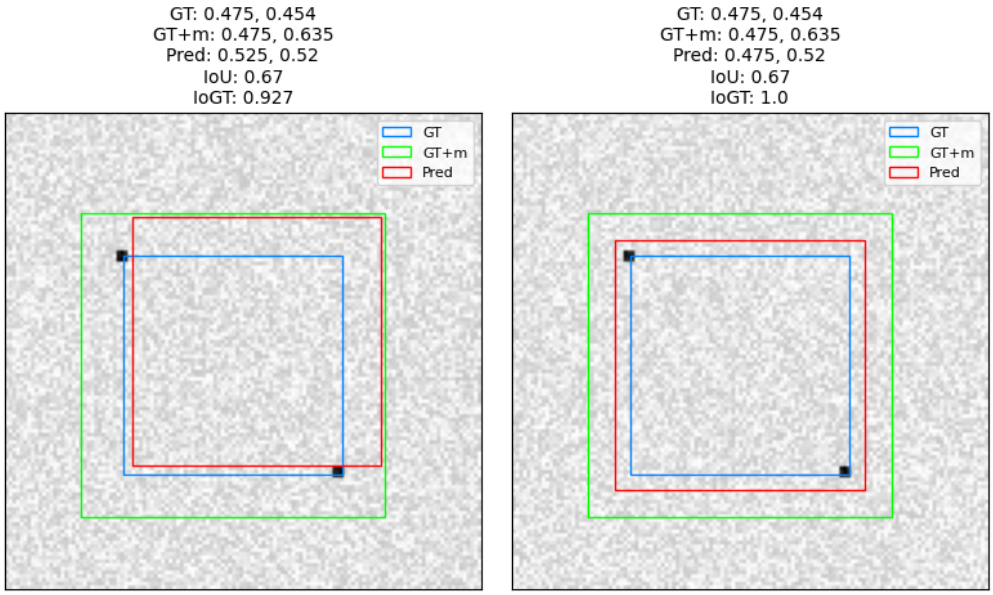
$$r_{constant} \sim U([0, 0.05])$$

3.4.2 Metrics for detection model performance and selection

Two different metrics were employed to evaluate the detection model's performance. The first metric was Intersection over Union. The second metric was *Intersection over Ground Truth*, which is an ad hoc metric for the detection task in this thesis.

IoU, as described in the background chapter, is considered as the de facto performance metric for detection [36]. However, IoU does not represent the purpose of the detection model in this case. The goal of the detection model is to predict a bounding box that, while precise, covers the entirety of the pattern. IoU between prediction and GT_{+m} does not necessarily describe whether this goal is achieved. A prediction may give a suboptimal IoU score if the predicted bounding box is not precisely on top of the GT_{+m}. However, the prediction is considered *successful* if the predicted bounding box is placed to cover the entire PRI pattern. Conversely, in some cases, the suboptimal IoU may represent that detection indeed is unsuccessful, which leaves a part of the pattern outside of the predicted bounding box.

As shown in Figure 3.12, IoU between prediction and GT_{+m} are equal for both bounding box predictions. However, the prediction has failed in Figure 3.12(a), where all graphical elements of the pattern are left outside of the predicted bounding box.



(a) Failed detection (b) Successful detection

Figure 3.12: IoU and IoGT of a failed (a) and a successful (b) detection

In other words, IoU between prediction and GT_{+m} alone does not indicate whether the predicted bounding box covers the pattern. To answer this, intersection over ground truth (IoGT) was employed. IoGT is calculated by dividing the area of the intersection by the area of the ground truth bounding box. The ground truth bounding box (GT) is not to be confused with target bounding box (GT_{+m})

$$IoGT = \frac{|Pred \cap GT|}{|GT|} \tag{3.10}$$

IoGT is equal to one when the predicted bounding box has complete coverage over the ground truth. If a portion of the ground truth is excluded, then IoGT will be below 1, indicating that the prediction has failed to include the entirety of the pattern. A bounding box prediction with IoGT score one is considered successful detection.

In Figure 3.12(a), IoGT falls below 1, indicating a failed detection. In Figure 3.12(b), the prediction that includes the entirety of the pattern receives IoGT 1.

IoGT was used as the primary metric for evaluating the detection model, while IoU served as a secondary metric for the precision of the prediction. Following forwarding the training samples through the detection model, the proportion of predictions with successful detection was calculated. This proportion of successful detection is referred to as *IoGT score*, and served as a performance metric for each epoch. The average score of the last five validation epochs was used to select the best model.

3.4.3 Detection model training and selection

Five different models were tested for the detection task, each distinguished by the number of convolutional and pooling layers, referred to as convolutional blocks. A convolutional block for these models is composed of two convolutional layers and one max pool layer. The model name corresponds to the number of convolutional blocks the model contains, i.e., the model named 2Conv has two convolutional blocks.

The following models were tested.

- 1Conv
- 2Conv
- 2Conv with feature fusion
- 3Conv
- 4Conv

The model 2Conv with feature fusion was a variant 2Conv model. This variant fuses feature maps from the first and second convolutional blocks, following methods presented in the original Feature Pyramid Network paper by Lin et al. [46]. The feature pyramid network proposes fusing feature maps from different levels in convolutional layers to improve the detection of objects with varying sizes in the image [46]. The purpose of testing this model was to assess whether such a technique would benefit the detection task.

Appendix A provides diagrams detailing the structure of each model.

The last phase of the detection model selection tested different loss functions. The model selection was conducted with L2 loss. Thereafter, L1 loss and GIoU Loss was tested on the selected model to assess whether they could further improve its performance.

Common hyperparameters for both model and loss function selections were:

- Optimizer: Adam
- Learning rate: 0.0005
- Number of Epochs: 200

3.5 Implementation of the Classification model

This subsection describes the implementation of the classification model. The implementation of the classification model was a relatively smaller task compared to the implementation of the detection model. There was less requirement for feature engineering, as images were already labeled with target classes. Furthermore, candidate classification models were chosen from pre-existing classification models.

3.5.1 Training data

Training and validation data for the detection model was reused for the classifier. However, in the case of the classifier, the images with a pattern size below 0.1 were excluded from the training set under assumption of that the classifier will only receive images with optimal pattern size. If the detection model performs as intended, the classifier should not encounter any images with such a small pattern size. Augmentations remained unchanged; training and validation data contained images without augmentation and images with blurring, noise, and both.

3.5.2 Metrics for classification model performance and selection

Accuracy was employed as the evaluation metric for classification performance. Accuracy is expressed in Eq. 3.11

$$Accuracy = \frac{N_{CorrectPredictions}}{N_{TotalPredictions}} \tag{3.11}$$

In the same fashion as the evaluation and selection of the detection model, the average score from the last five validation epochs was used to select the best classification model.

3.5.3 Model selection

There were several image classification CNN models to choose from for the classification task. Three candidate models were selected for this task.

AlexNet

Alexnet was proposed in 2012 by Alex et al. [47]. The network architecture is relatively small, making it easier to comprehend than many modern state-of-the-art models. AlexNet used in this thesis is based on Krizhevsky's paper *One weird trick for parallelizing convolutional neural networks* [48], not the original paper from 2012 [49]. The network has eight layers (five convolutional and three dense) [49]. Compared to the next candidate model, AlexNet is smaller. The purpose of having AlexNet as a candidate model was to test the performance of a compact modern deep CNN on our data set.

DenseNet121

DenseNet was presented by Huang et al. in 2016 [50]. The architecture employs dense connections between layers, where all layers are connected. The variant used in the thesis consists of 121 layers (hence the name) [48]. The purpose of having DenseNet as a candidate model was to test the performance of a state-of-the-art model on our data set.

Interestingly, its noteworthy that the depth of the DenseNet121, compared to AlexNet, does not translate to more parameters. According to PyTorch documentation, DenseNet121 used in this thesis has 7978856 parameters, whereas the number for AlexNet is 61100840.

The model implemented in Barrios' thesis work

The final candidate was the model implemented in the directly related thesis of Barrios [9]. The purpose of including Barrios' model was to assess how well it performs on PRI images involved in my thesis.

All models were slightly modified to suit the context of the task in this thesis. An additional dense layer was added to AlexNet and DenseNet that matches our number of classes. For Barrios' model, the last layer was modified to output 14 classes. All classification models were trained for 100 epochs with cross-entropy loss function and ADAM optimizer with a learning rate of 0.0005.

3.6 Pipelining Detection and Classification Models

The selected detection and classification models were pipelined together for final evaluation on the test set. The flow of image samples through the pipelined models is described in the following pseudocode.

1: Pass a PRI image to the detection model to infer the position and size of the pattern within the image.
2: **if** The predicted pattern size is too small (size below 0.1) **then**
3: Generate a new PRI image from the PRI sequence with parameters of the predicted bounding box and image PRI range of the original image.
4: The newly generated image becomes classifier input
5: **else**
6: Image passed to the detection model becomes classifier input
7: **end if**
8: Pass the classifier input to the classifier for classification

3.6.1 Generating new images from the detection

If the size of the detected pattern is smaller than the threshold, a new image for classification was generated from the PRI sequence belonging to the image. The image was generated utilizing the parameters of the predicted bounding box. The new image covered a smaller PRI range than the original image, so the depicted pattern received a larger size.

The position and size of the predicted bounding box, with the PRI range of the image, were involved in the calculation of the new PRI range given by minimum and maximum Image PRI expressed in Eq. 3.12 and 3.13.

$$\Delta I = I_{max} - I_{min}$$

$$I^*_{min} = I_{min} + \Delta I(\widehat{Pos} - \frac{\widehat{Size}}{2}) \tag{3.12}$$

$$I^*_{max} = I_{min} + \Delta I(\widehat{Pos} + \frac{\widehat{Size}}{2}) \tag{3.13}$$

I^*_{min} and I^*_{max} are the new minimum and maximum image PRI, respectively.
$\widehat{Pos}$ and $\widehat{Size}$ are the position and size of the predicted bounding box, respectively.

3.6.2 Evaluation of the pipelined models

The research question is about the possibility of automation of PRI inspection for the classification of the PRI pattern. Therefore, the evaluation of the final pipelined models emphasized the classification task. The same method for evaluating the standalone classification model was utilized, using accuracy score as the metric.

Given that the detection model is a significant part of the thesis, evaluation will also consider the impact of detection. For each sample that is passed through the detection, its following outcomes were tracked. This involved whether a new image was generated based on the detection, and if so, whether it was based on a successful detection.

Chapter 4

Results

4.1 Selection of the Detection Model

Figure 4.1 shows overview of the selection process for the detection model.

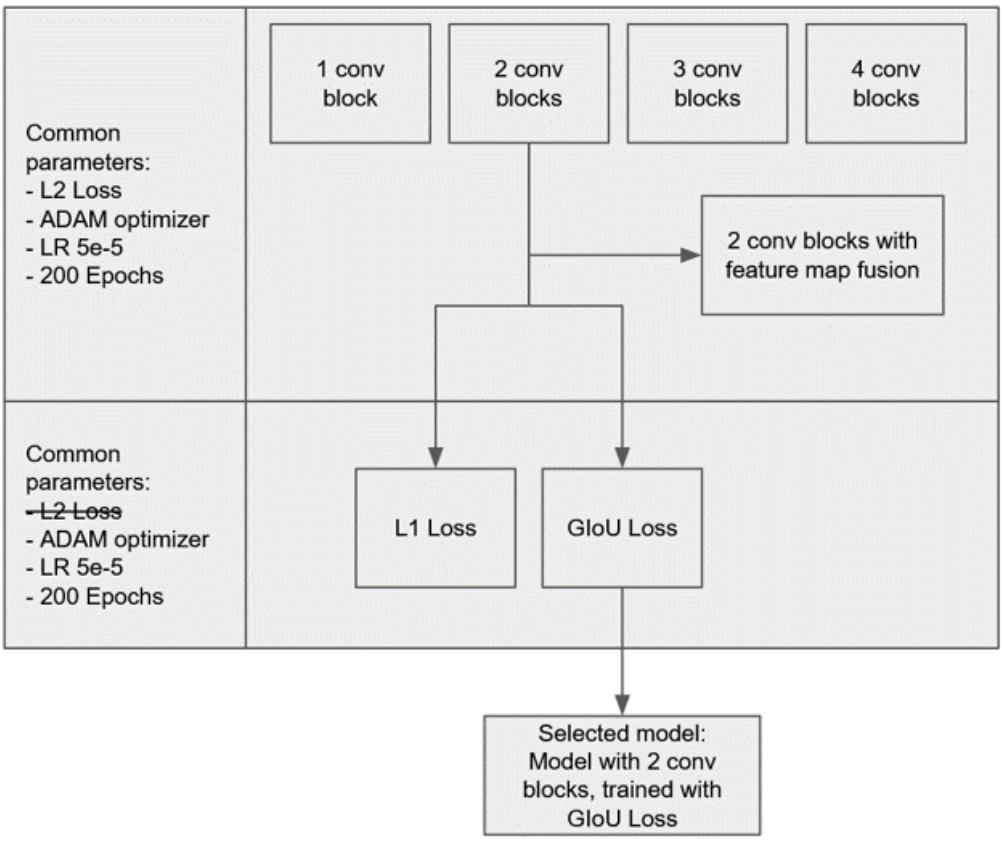


Figure 4.1: Overview of detection model selection process

Table 4.1 shows the performance metrics of the detection models involved in the selection. All models performed similarly, as indicated by the metrics. The difference between the lowest and highest scores was only a few hundredths. Furthermore, the best model's IoGT score surpassed the next-best score by only a few thousandths. This pattern was also apparent across IoU scores, where the models showed no significant differences. 2Conv, the model containing two convolutional blocks, achieved the highest performance metrics based on the mean IoGT score of the last five validation epochs. This model was selected for further testing with L1 and GIoU Loss functions.

	1Conv	**2Conv**	**2Conv with feature fusion**	**3Conv**	**4Conv**
Mean IoGT score of last five epochs	0.914	0.936	0.924	0.931	0.926
Mean IoU of last five epochs	0.800	0.831	0.807	0.814	0.819

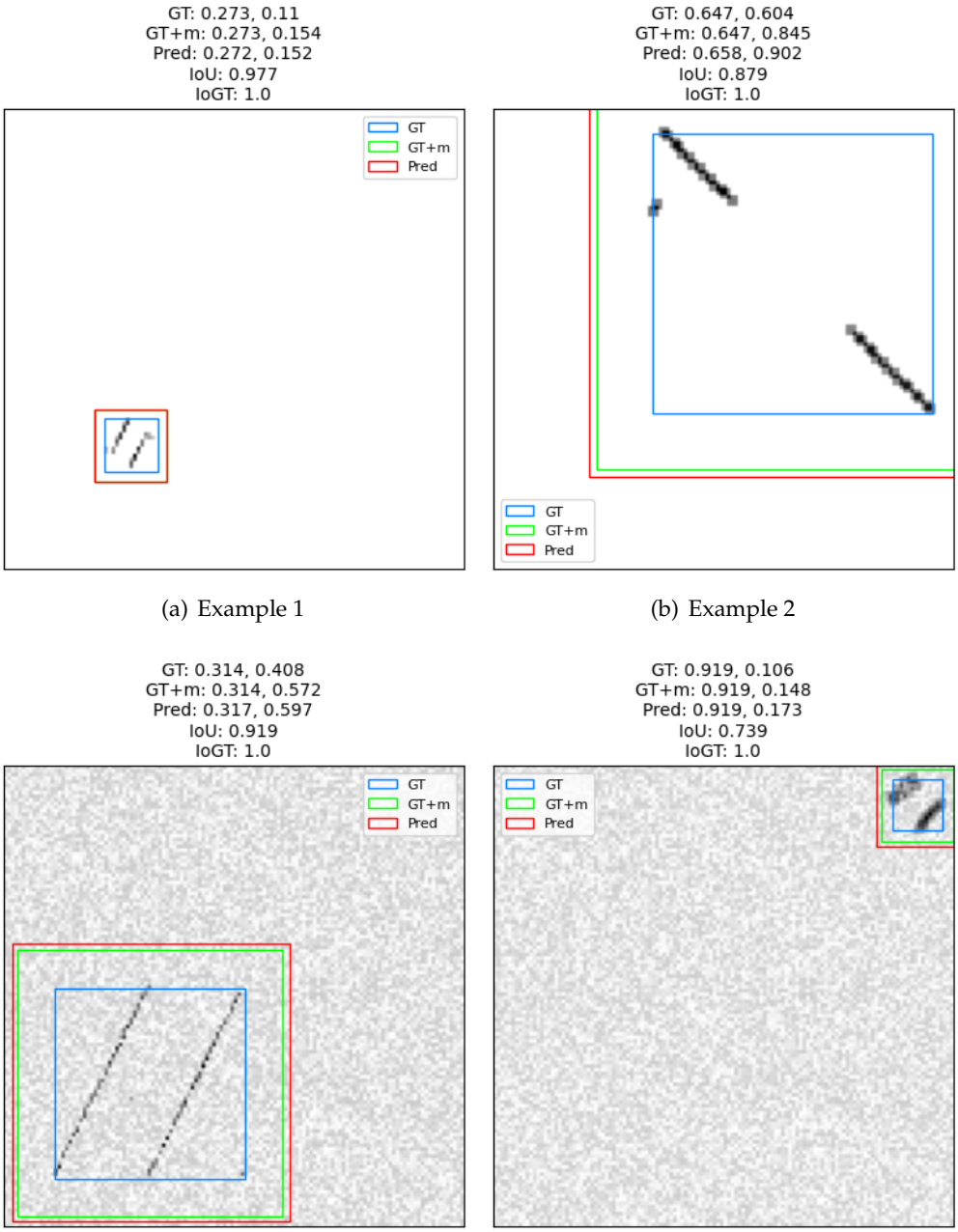
Table 4.1: Performance metrics of different detection models

Table 4.2 shows the performance metrics of 2Conv model trained with different loss functions. The differences in scores across the tested loss functions are slight. The scores did not reveal any significant performance difference among the models. The Conv2 Model trained with GIoU loss emerged on top by a slight margin. The final selected detection model was 2Conv model trained with GIoU loss and hyperparameters described in Figure 4.1.

	2Conv, L1 Loss	**2Conv, L2 Loss**	**2Conv, GIoU Loss**
Mean IoGT score of last five epochs	0.943	0.936	0.949
Mean IoU of last five epochs	0.846	0.838	0.848

Table 4.2: Performance metrics of detection models trained with different loss functions

Figures in Figure 4.2 demonstrate predicted bounding box by the selected model.



(a) Example 1

(b) Example 2

(c) Example 3

(d) Example 4

Figure 4.2: Detection Examples

4.1.1 Detection Model Classwise performance

Appendix B shows the IoU and IoGT score distribution of all predicted bounding boxes by the tested models on the validation data set. For each class, the bar on the left in the figure shows the distribution of IoU scores of predicted bounding boxes. This bar gives insight into the precision of the predictions. The bar on the right for each class shows the distribution of IoGT for each class. In other words, this bar shows the proportion of successfully detected samples.

For most classes in the validation set, the model successfully detected the patterns in all samples. The scores on Const, Stg2 and Stg9 were slightly lower than the rest. Furthermore, successful detections of Const and Stg2 have generally lower IoUs than the rest. In the case of Const, the likely cause is that models are trained to infer a size larger than the target size in the validation set, considering how the random size variation during the training was implemented.

These figures also reveal some noticeable differences in performance despite the similar evaluation metrics across the models and loss functions as shown in Table 4.1 and 4.2. Figures 4.3, 4.4 and 4.5 are from Appendix B. Comparing Figure 4.3 and 4.4, it is apparent that the number of convolutional blocks affected successful detection on some of the classes, namely Const and Stg2. Comparing Figures 4.3 and 4.5, it is apparent that model trained with GIoU loss improved the rate of successful detections compared to the L2 loss, where the model trained with the former successfully detected all samples in most classes.

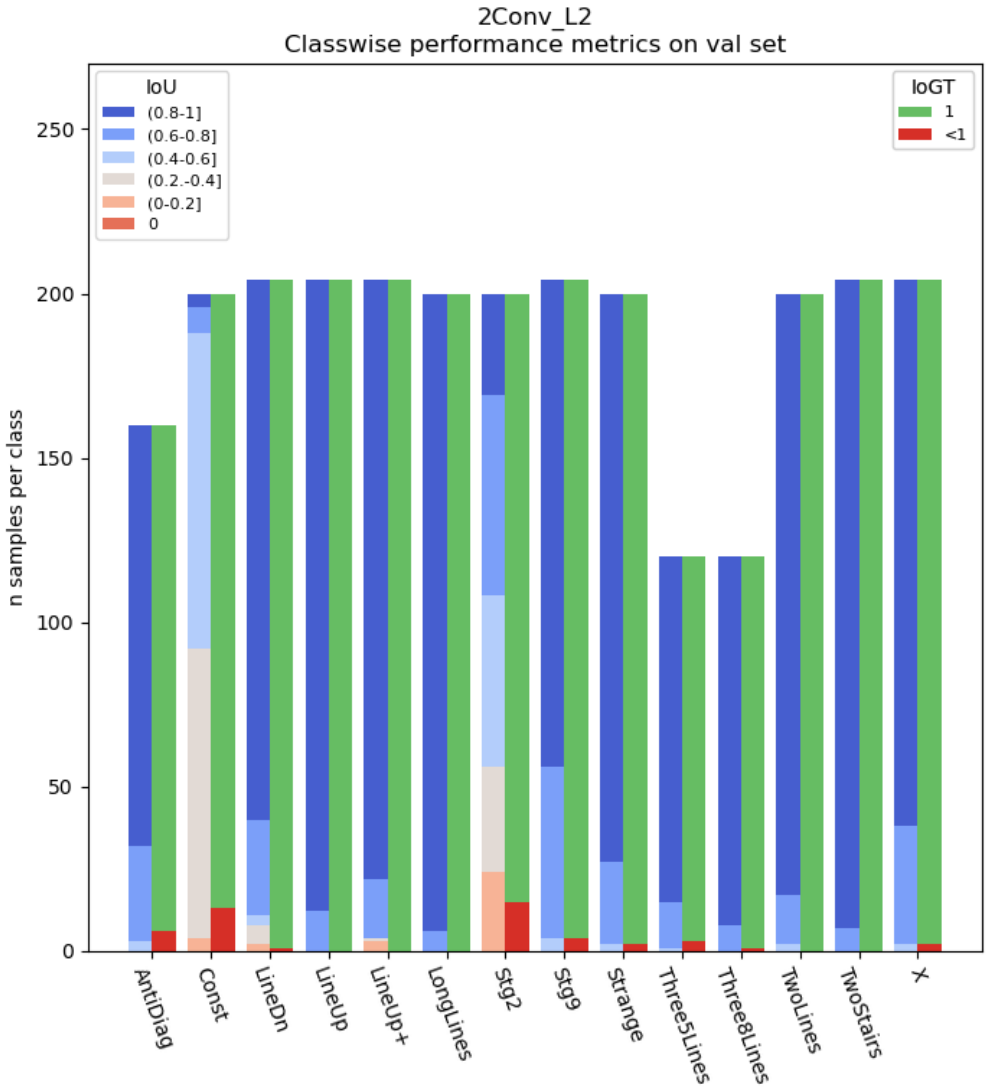


Figure 4.3: Excerpt from Appendix B. Classwise performance metric distribution of 2Conv trained with L2 Loss

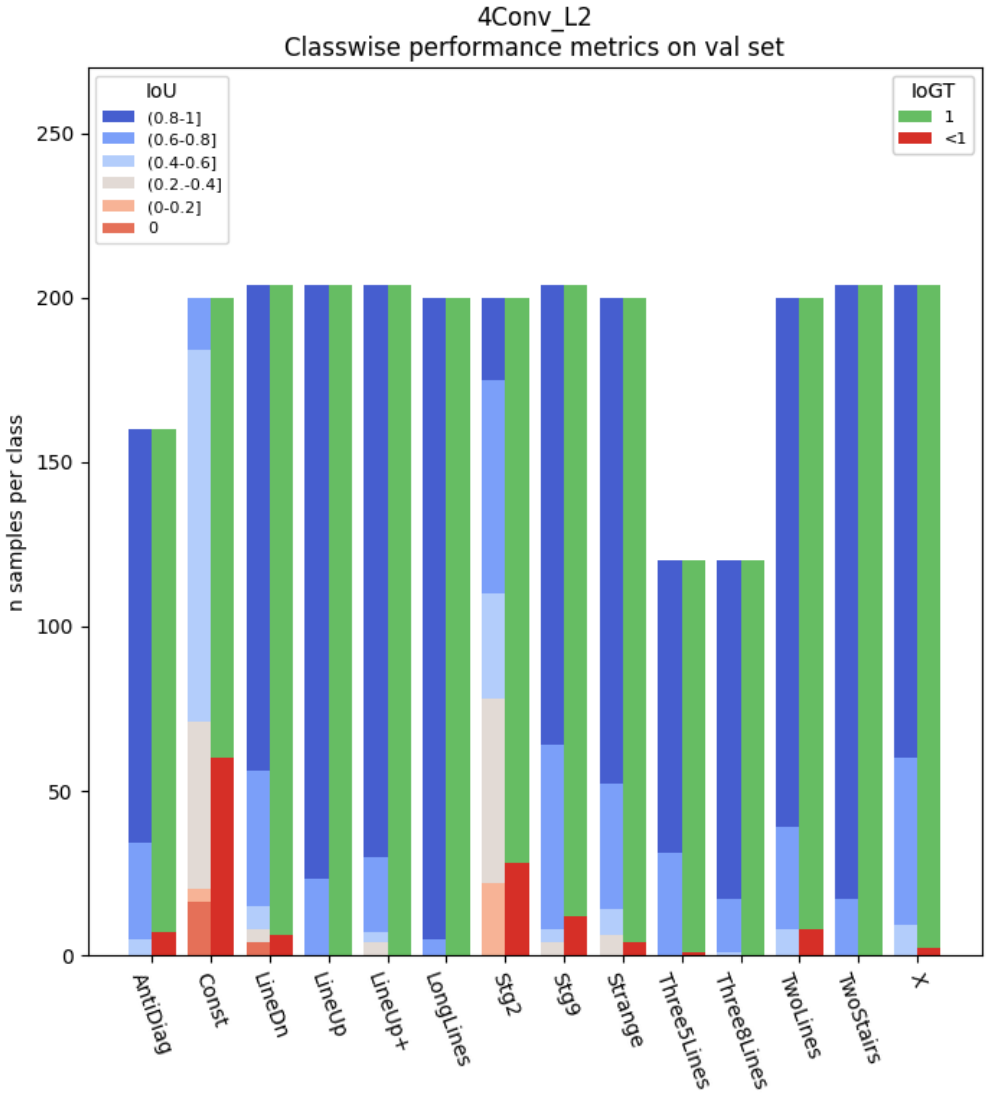


Figure 4.4: Excerpt from Appendix B. Classwise performance metric distribution of 4Conv trained with L2 Loss

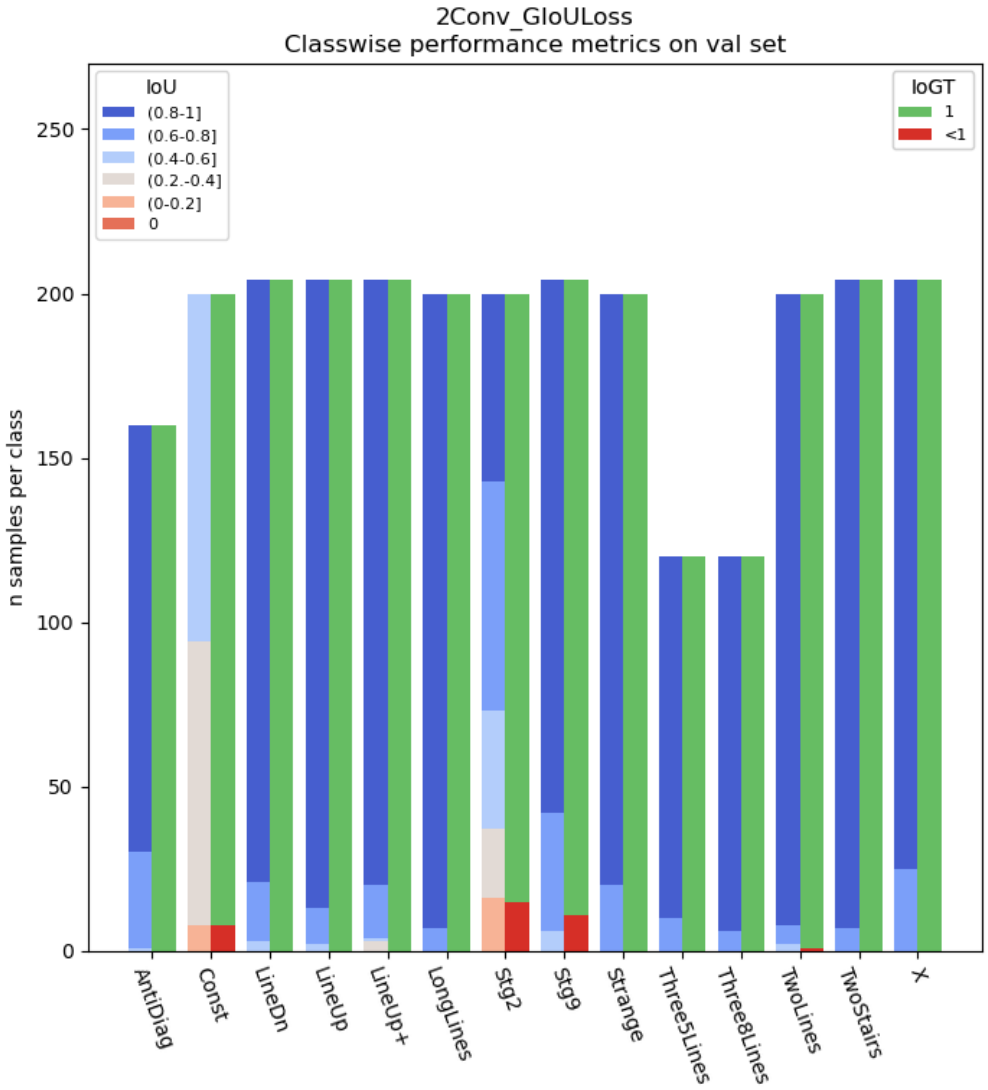


Figure 4.5: Excerpt from Appendix B. Classwise performance metric distribution of 2Conv trained with GIoU Loss

4.2 Selection of the Classification Model

Table 4.3 shows the mean accuracy of each classification model's last five validation epochs.

	Barrios'Model	**AlexNet**	**DenseNet121**
Mean Accuracy, last five epochs	0.934	0.975	0.993

Table 4.3: Accuracy of the classification models

Accuracy scores show that Barrios' model performed inferior to the other two classification models. The confusion matrix of Barrios' model and AlexNet in Figures 4.6 and 4.7 reveal that the most significant inaccuracy in the two models lies in confusion between the LongLines-TwoStairs and Antidiag-Stg2 pairs. As shown in subsection 3.1.2, LongLines and TwoStairs share a strong visual resemblance. The same can be said with Antidiag and Stg2 in the instances where the pattern size of the former is suboptimally small, causing the pattern to take the for m of two diagonally placed dots similar to the pattern of Stg2.

DenseNet121 achieved near-perfect performance on the validation set with a total accuracy of 0.993 and was selected as the final classification model for evaluation on the test set. The validity of this overly optimistic performance is discussed in the next chapter, as it is likely caused by overfitting.

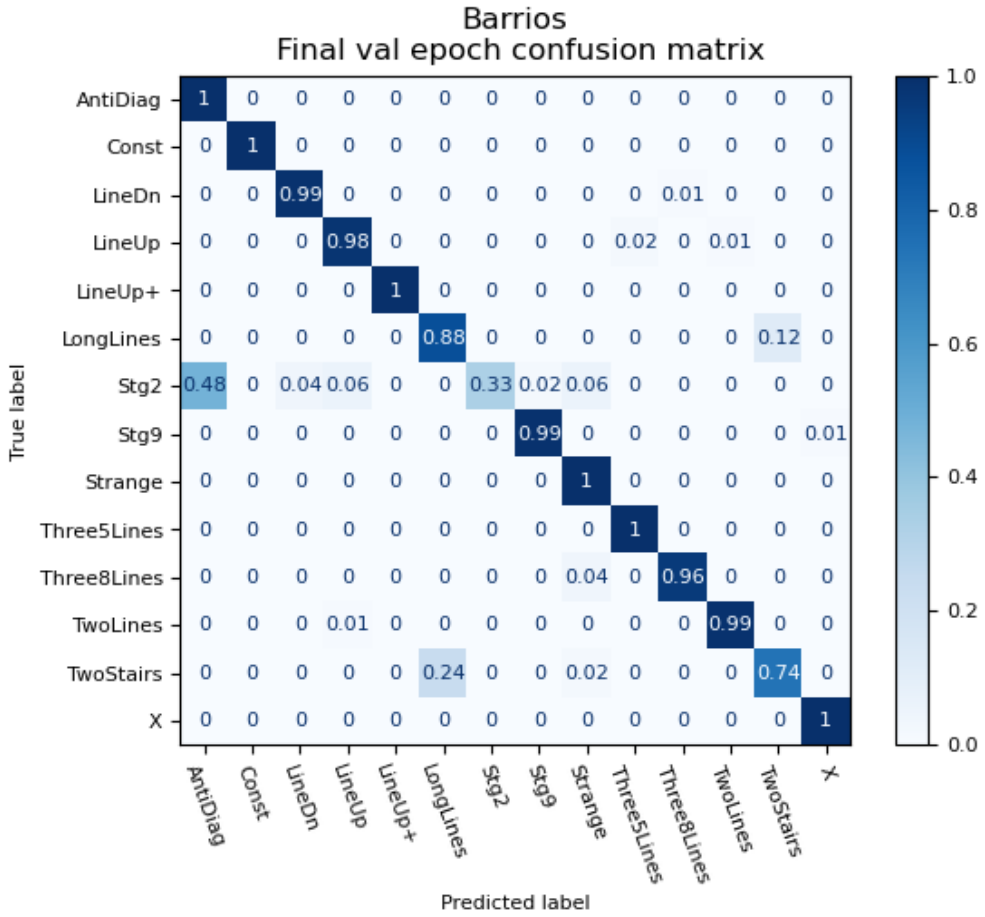


Figure 4.6: Confusion matrix of Barrios' model classification performance

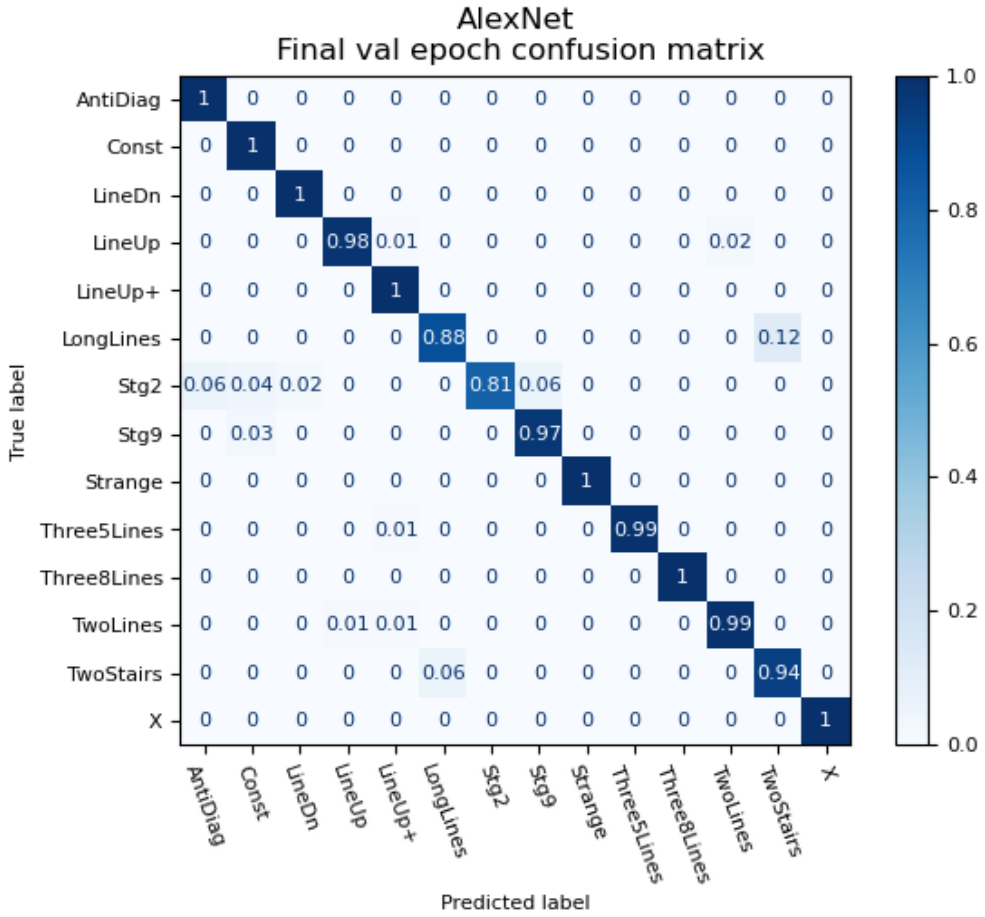


Figure 4.7: Confusion matrix of AlexNet classification performance

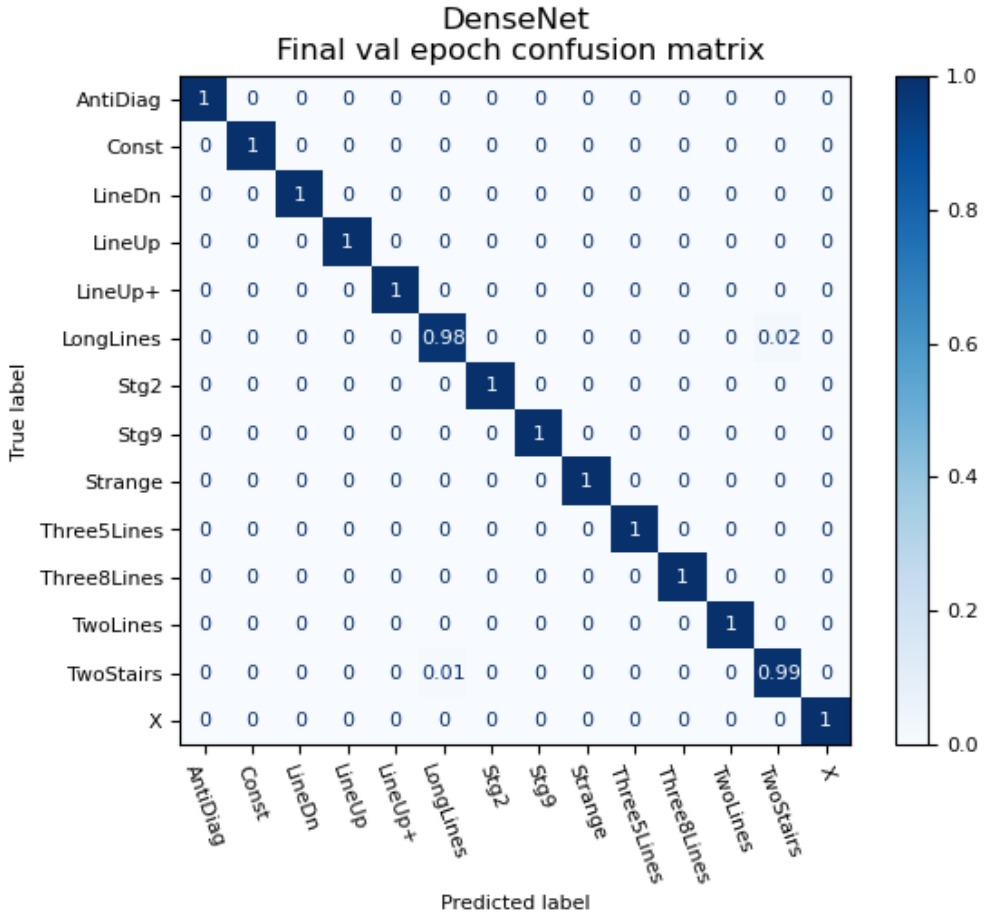


Figure 4.8: Confusion matrix of DenseNet121 classification performance

4.2.1 Impact of pattern size on classification task

The Figures 4.9, 4.10 and 4.11 show classifiers' performance based on the pattern size. In these figures, the bar on the left for each class shows the size distribution for the correct classifications. The bar on the right shows the distribution of misclassifications. Recall that the validation set does not contain images with a pattern size below 0.1, as described in subsection 3.5.1. Therefore, the range (0, 0.2] contains patterns with size within the range [0.1, 0.2]. Frequent misclassifications on smaller patterns are apparent for Barrios' model and AlexNet.

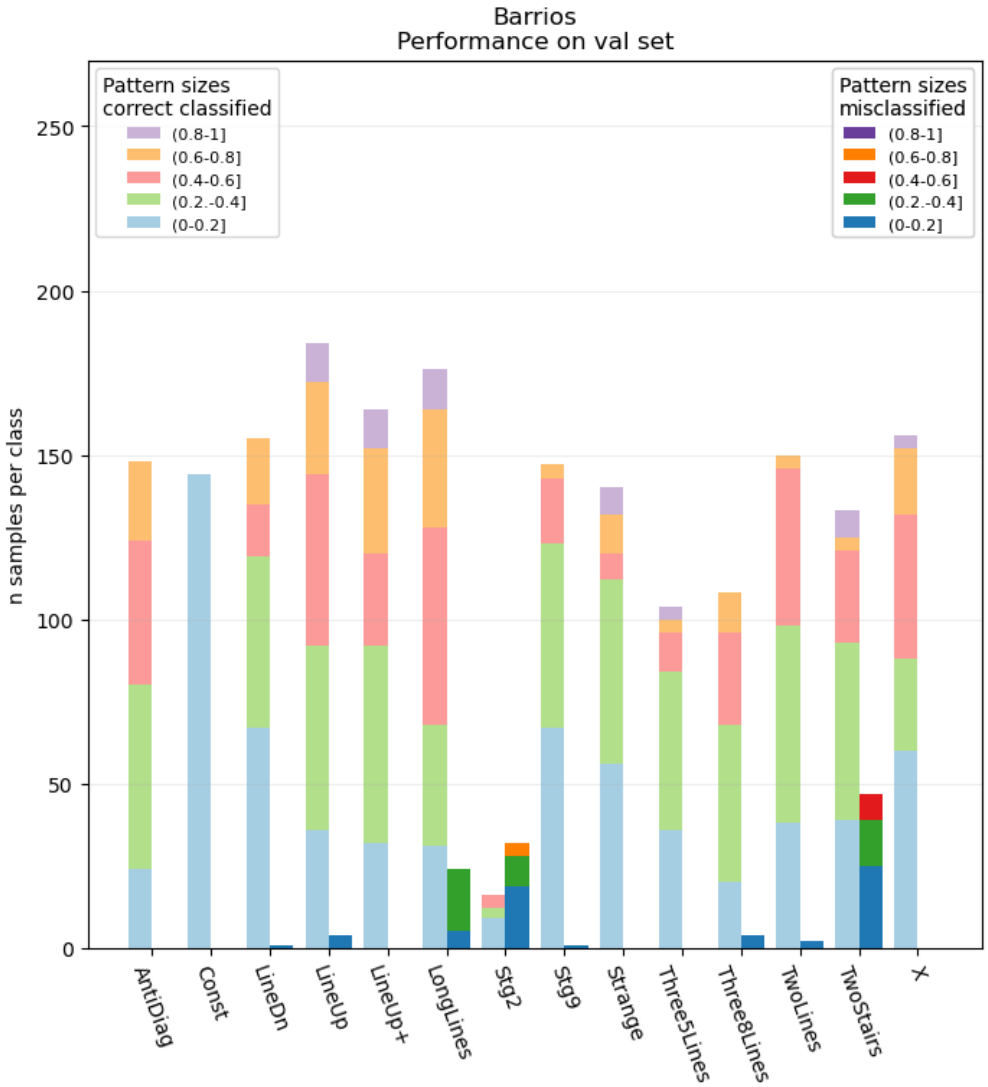


Figure 4.9: Barrios' model classwise distribution of image sizes of predictions

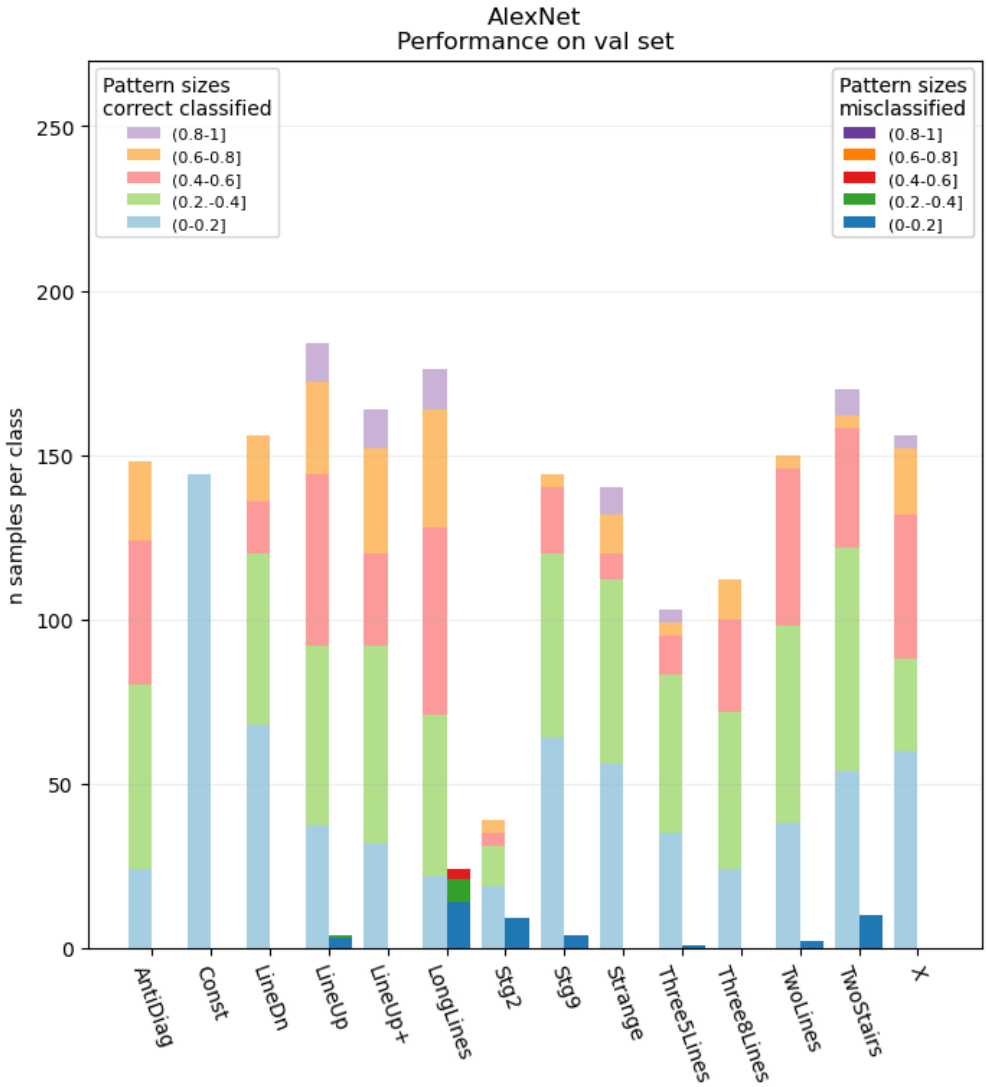


Figure 4.10: AlexNet model classwise distribution of image sizes of predictions

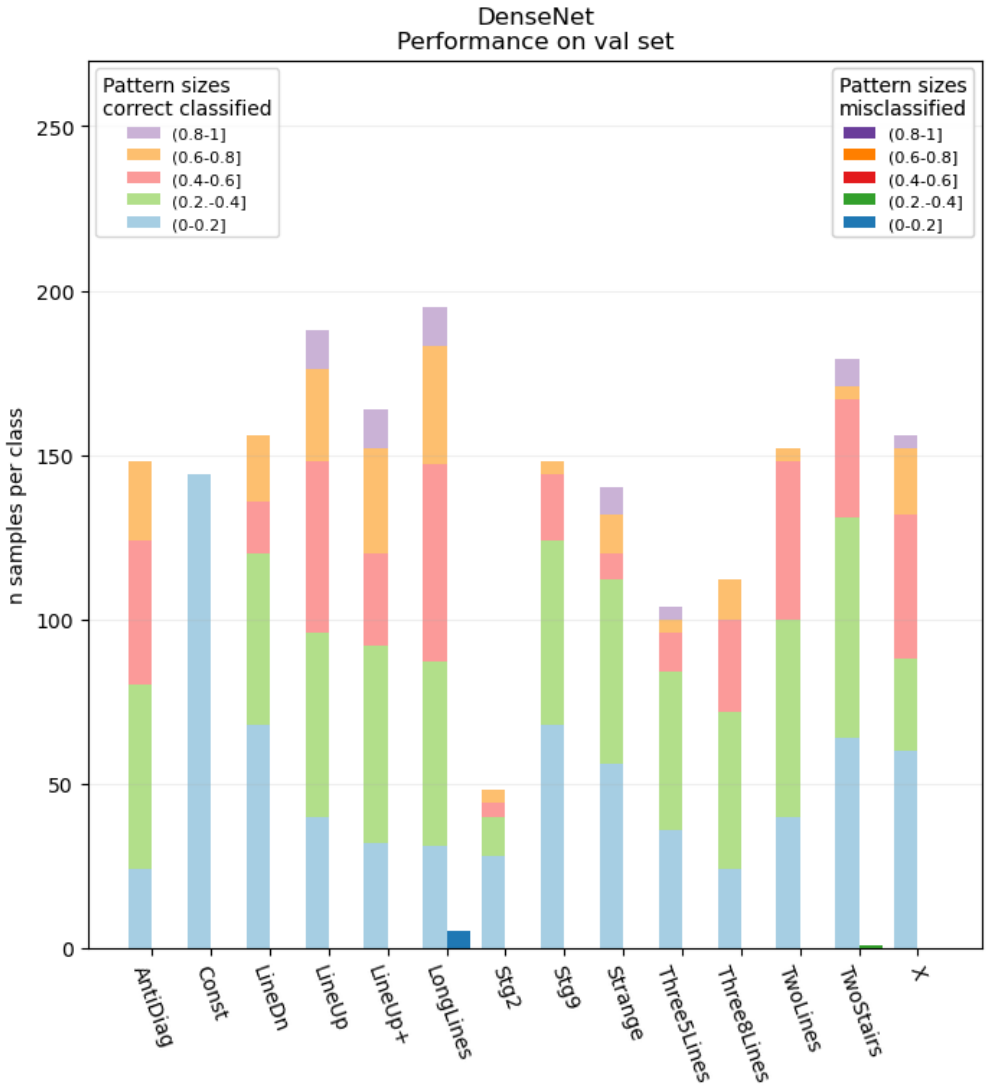


Figure 4.11: DenseNet model classwise distribution of image sizes of predictions

4.3 Performance of the Pipelined Detection and Classification Models

4.3.1 The flow of test samples through the pipeline

The Sankey diagram in Figure 4.12 illustrates the flow of samples in the test dataset through the pipeline. The two nodes in the middle show the number of samples directly forwarded to the classifier and the number of samples resulting in new image generation, respectively. In other words, they represent how the detection model inferred the size of the patterns. Out of the 2628 samples in the test set, the detection model inferred that 551 of the images contained pattern too small (smaller than 0.1) for classification. Consequently, new PRI images were generated with size parameters inferred from the detections.

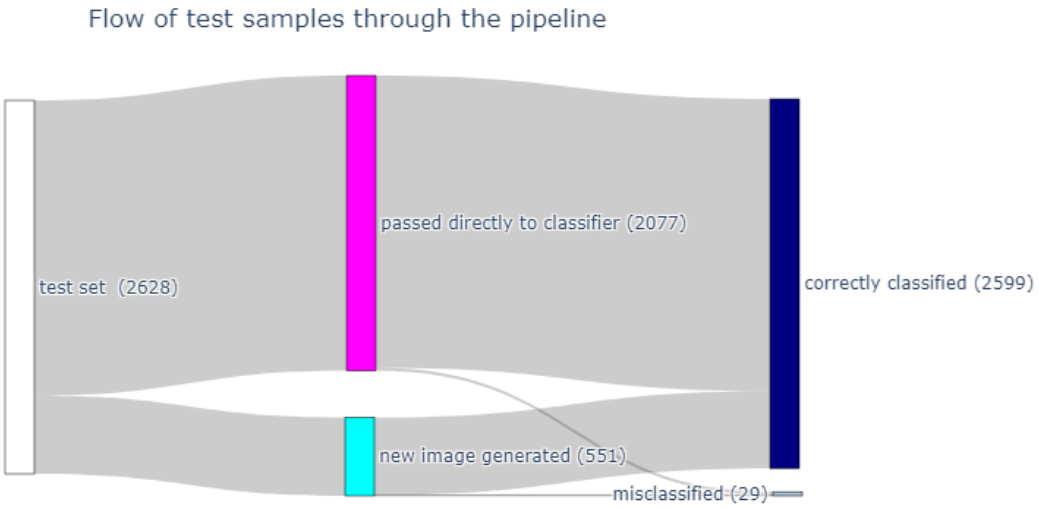


Figure 4.12: Flow of Test Samples Through the Pipelined Detection and Classification Model

The Sankey diagram in Figure 4.13 focuses on the flow of samples that resulted in new image generation. For these samples, the detection model predicted that the pattern size is less than 0.1, which consequently prompted generating a new image. The two nodes in the middle show whether it is based on successful detection. The final nodes show whether images generated from these detections were correctly classified. The detection model inferred that 551 samples have a pattern size below 0.1. 34 of these samples had erroneous detection. Out of these failed detections, only seven were misclassified.

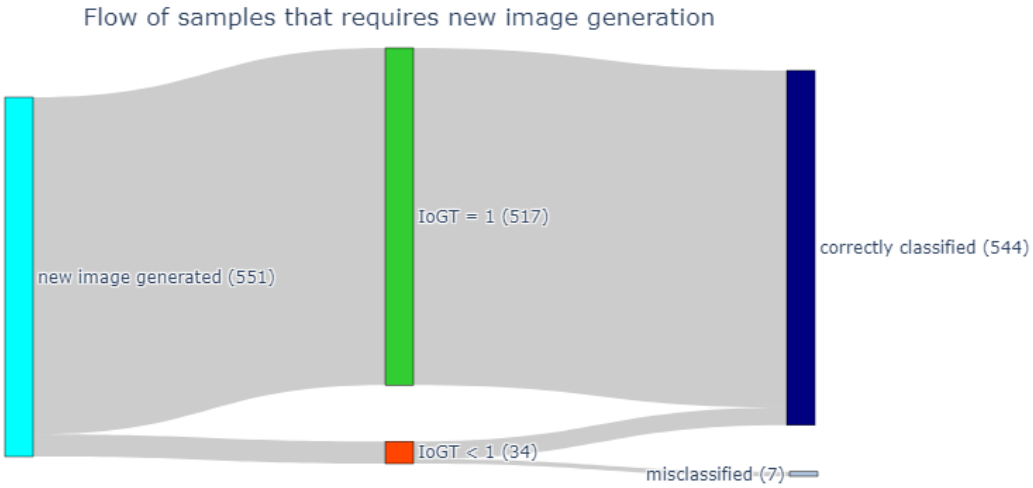


Figure 4.13: Flow of Test Samples Resulting in New Image Generation

4.3.2 Final classification on test set

Figure 4.14 shows confusion matrix of the pipelined model's classification on the test dataset. Similar to the performance during classification model selection, the model attained an almost perfect prediction with total accuracy of 0.989 on the test dataset passed through the detection model. This overly optimistic final classification performance is likely caused by overfitting and is addressed in the discussion.

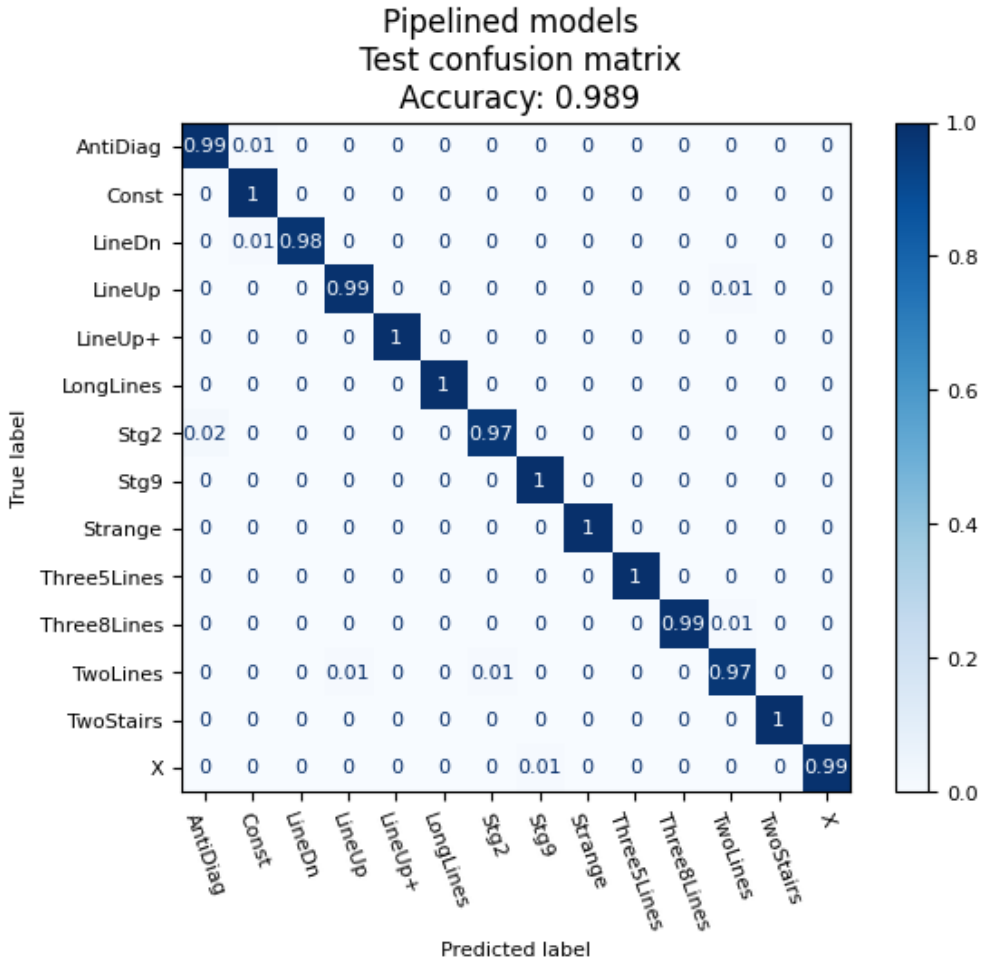


Figure 4.14: Confusion matrix of pipelined models' classification performance

Chapter 5

Discussion

5.1 Model Performance

5.1.1 Impact of Detection Model

The classification accuracy of the pipelined model does not directly illustrate how impactful the detection model and subsequent new image generation were for the correct classifications. The near-flawless classification performance could be attributed to the powerful DenseNet121 model. However, Figures 4.9, 4.10 and 4.11 showing the impact of pattern size on classification indicate that misclassifications, albeit very few, occur primarily on smaller patterns. Although it may not be the case for the selected DenseNet121, it highlights the challenge of attempting classifications on smaller patterns, necessitating a mechanism for detecting too small patterns and subsequently generating a new image with optimal image PRI range.

Furthermore, Sankey diagrams in subsection 4.3.1 indicate that most of the newly generated images are correctly classified. Notably, a significant portion of these newly generated images belong to the pattern class Const, which inherently have a small pattern size of 0.05. This implies that all images of Const pattern predicted with accurate size underwent new image generation. Generating new images of all successfully detected Const patterns might appear redundant. However, any pattern that resembles Const pattern, i.e. a single dot, is ambiguous until it has been enlarged in an image with a narrower image PRI range. Any pattern can appear as dot, given a large enough image PRI range, thus justifying generating new images at instances with Const-like patterns.

5.1.2 Overfitted Classification Model

The selected classification model achieved near-perfect classification on the validation and test set. In many cases, high performance on the evaluation sets usually suggests the model's capacity to generalize to data unseen during training. Furthermore, as shown by the loss curves in Appendix D, the validation loss does not start to increase at any point during training except for few noisy spikes. These observations do not

indicate that the model is overfitted to the training data. However, this near-perfect performance is caused by our insufficient dataset, which contains numerous duplicates spread across the data splits.

The dataset provided by FFI is generated by augmenting 65 unique samples of PRI images. There were three to five individual samples in each class. Upon inspection, the patterns were almost identical within each class. The only contributing factor was PRI noise, which was later eliminated with the heatmap and thresholding presented in section 3.2.1. This may have further exacerbated the issue of limited variance. These 65 samples were subjected to scale and translational augmentations. These geometrically augmented images constitute the images provided by FFI, populating the original 65 unique samples to 6565 samples. Whether the provided geometric augmentations contributed to increased variation in each pattern class is questionable. First, CNN is translational invariant, as described in section 2.4.2. Generating several images of the same pattern with similar size but in different positions does not contribute to the variation in the representation of the pattern class. This will only duplicate the pattern sample with some size and contribute to overfitting.

Whether the applied Gaussian blurring and random noise contributed to the variation within the data set is questionable. The reason behind skepticism is that the photometric augmentations were applied to a dataset that already contained duplicates. It did not bestow any uniqueness or distinctiveness and merely created photometrically altered versions of the duplicates.

According to FFI, the limited of variations within each pattern class are an inherent nature of the PRIs of navigational radars provided in this thesis. While this is not a flaw, it likely caused our model to overfit. Given the data that models are trained upon, the viability and applicability of the model depend on the domain in which it is deployed. It can be assumed that the model performs well if the input to the model is PRI images depicting one of the 14 patterns that have undergone the same image processing as in the thesis. Furthermore, given the limited variations within each class in our dataset, a less powerful model or technique could suffice, as there is less need for a model that can generalize to rich variations.

5.2 Model Evaluation And Selection

5.2.1 IoGT as ad hoc metric

For evaluating and selecting the detection model, successful detection was defined as a model's ability to predict a bounding box that covers the entire pattern to avoid excluding part of the pattern. To quantify this, Intersection over Ground Truth was employed, which assesses whether square A (predicted bounding box) has coverage over square B (a square that covers the pattern). This metric was the deciding metric used to select the best model. However, given the results from the training and validation, the outcome of model selection would remain the same if IoU was solely used for evaluation. Results show a connection between IoU and IoGT i.e., the model

with the highest IoGT also had the highest IoU.

Furthermore, IoGT should only be employed as an evaluation metric and not be incorporated into a loss function. The reason is that IoGT does not consider the prediction's precision. It only considers predicted bounding boxes' coverage over the ground truth. To elaborate, a model can achieve an IoGT score of one (max score) only by predicting an excessively large bounding box covering the ground truth bounding box in all instances.

5.2.2 Model Selection

Different models and loss functions were tested under detection model selection. The impact of the number of convolutions and pooling operations was coincidentally discovered, which prompted the testing of different numbers of convolutional blocks. The loss functions were chosen based on their relevance to the object detection problem. This selection method does not adhere to any standardized system or principle for model testing and selection. Simultaneously, exploring all possible combinations of models, loss functions, and other training hyperparameters would result in an overwhelming number of tests, thus leading to the limitation of testing procedures to different models and loss functions.

Given the apparent issue of overfitting, there may be limited benefits to be gained from exploring different classification models as they are likely to overfit to the development data set. One area that could yield benefit is exploring models and techniques that employ stronger learning regularization to tackle the overfitting encountered. Furthermore, utilizing images provided by FFI, dotted with spurious and missing pulse noise, could also contribute to tackling this problem.

Chapter 6

Conclusion

Detection and classification models were developed to automate the PRI pattern classification as an image classification task. The detection model was developed to infer the size and location of the PRI pattern within the image. The predicted size and location were used to generate a new image containing the same pattern with optimal scale for the classification task. Classifications were made on these images with optimal pattern size, resulting in a near-perfect performance.

The optimistic outcome was likely attributed to overfitting, caused by images with insufficient variations and complexity within each pattern class. Utilizing a powerful state-of-the-art classification model and noise suppression of images may have further exacerbated the overfitting problem. The model will likely achieve similar performance in a deployment environment that provides data with the same predefined classes and processing techniques involved in the development of the model. The model is confined to the closed set of the 14 classes in the training set, so it cannot recognize a new unseen PRI pattern.

In light of these results and reflections, the following conclusion is drawn: Automating the classification of PRI pattern types with deep learning methods for computer vision is possible within the scope of the provided data. Careful consideration should be given before the model's deployment with respect to the deployment environment and the development set employed for training the models.

6.1 Future Work

The first proposal of future work is to implement a classification model with open set recognition. In the current configuration, the classification model will exclusively recognize PRI image classes in the training set and will not recognize an unseen class as unknown. FFI assesses that the the pattern classes involved in this thesis are among the classes most frequently used by navigation radars, and together they cover the majority of radars, at least in the recordings collected from the Oslofjord. However, a deployed model could benefit from a mechanism that detects when an input image contains a pattern that does not belong to any of the predefined classes involved in training. This

mechanism will not only mitigate misclassification, but will also allow more timely discovery of new pattern classes.

The second proposal is to modify and refine the implemented detection model so that it can be employed to precisely determine the minimum and maximum PRI of the pattern. A bounding box that precisely encloses the pattern can be translated to the pattern PRI range (P_{max} and P_{min}). The precision of inferring pattern PRI range can be further improved by an iterative approach, where the model iteratively focuses on the pattern based on its predicted bounding boxes before splitting its focus to edges of the bounding box for precisely determining P_{min} and P_{max}. This can potentially automate the part of the analysis that seeks to determine the radar's minimum and maximum PRI, which is also an important part of the PRI analysis and parameterization of an emitter, which contributes to building a reliable emitter database for ESM systems.

Bibliography

[1] A. De Martino, '1.1 definition and EW role in the military field,' in *Introduction to modern EW systems*, Artech House, 1st Aug. 2012, ISBN: 978-1-60807-207-1.

[2] *IEEE standard for radar definitions*, ISBN: 9781504440622, 23rd Mar. 2017. DOI: 10.1109/IEEESTD.2017.8048479. [Online]. Available: https://ieeexplore.ieee.org/document/8048479/ (visited on 22/08/2023).

[3] *JP 3-85 joint electromagnetic spectrum operations*, 22nd May 2020. [Online]. Available: https://www.jcs.mil/Portals/36/Documents/Doctrine/pubs/jp3_85.pdf.

[4] A. Eneroth, 'EW system functions and effectors,' in *Applied Radar EW*, Stockholm: Eneroth Publishing, 1st Oct. 2019, ISBN: 978-91-519-0546-4.

[5] D. Adamy, *EW 101: A First Course in Electronic Warfare*. Artech House, 1st Feb. 2001, ISBN: 1-58053-169-5.

[6] H. P. K. Nguyen, H. Q. Nguyen and D. T. Ngo, 'Classification of pulse repetition interval modulations using neural networks,' in *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, Bangalore, India: IEEE, Nov. 2018, pp. 1739–1743, ISBN: 978-1-5386-9276-9. DOI: 10.1109/SSCI.2018.8628913. [Online]. Available: https://ieeexplore.ieee.org/document/8628913/ (visited on 22/08/2023).

[7] H. Rong, W. Jin and C. Zhang, 'Application of support vector machines to pulse repetition interval modulation recognition,' in *2006 6th International Conference on ITS Telecommunications*, Chengdu, China: IEEE, Jun. 2006, pp. 1187–1190, ISBN: 978-0-7803-9586-2. DOI: 10.1109/ITST.2006.288819. [Online]. Available: http://ieeexplore.ieee.org/document/4068799/ (visited on 23/08/2023).

[8] G. Noone, 'A neural approach to automatic pulse repetition interval modulation recognition,' in *1999 Information, Decision and Control. Data and Information Fusion Symposium, Signal Processing and Communications Symposium and Decision and Control Symposium. Proceedings (Cat. No.99EX251)*, Adelaide, SA, Australia: IEEE, 1999, pp. 213–218, ISBN: 978-0-7803-5256-8. DOI: 10.1109/IDC.1999.754156. [Online]. Available: http://ieeexplore.ieee.org/document/754156/ (visited on 22/08/2023).

[9] K. Barrios, 'Automation of pulse repetition interval modulation classification,' 24th Jun. 2021. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-301651.

[10] C. Wolff and K. Hoel. 'Radar basics: Direction determination,' Radar basics: Direction Determination. (2009), [Online]. Available: https://www.radartutorial.eu/01.basics/Direction-determination.en.html (visited on 08/11/2023).

[11] D. Adamy, '2.1.2 radar modulations,' in *Introduction to Electronic Warfare Modeling and Simulation*, Boston: Artech House, 2003, ISBN: 1-58053-495-3.

[12] C. Wolff and K. Hoel. 'Radar principle - radartutorial.' Publisher: Dipl.-Ing. (FH) Christian Wolff. (), [Online]. Available: https://www.radartutorial.eu/01.basics/Radar%20Principle.en.html (visited on 22/08/2023).

[13] *JP 3-13.1 electronic warfare*, 8th Feb. 2012. [Online]. Available: https://info.publicintelligence.net/JCS-EW.pdf.

[14] E. Norgren, 'Pulse repetition interval modulation classification using machine learning,' 2019. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-241152.

[15] A. Eneroth, 'Pulse measurements,' in *Applied Radar EW*, Stockholm: Eneroth Publishing, 1st Oct. 2019, ISBN: 978-91-519-0546-4.

[16] A. De Martino, '3.7 emitter deinterleaving and sorting,' in *Introduction to modern EW systems*, Artech House, 1st Aug. 2012, ISBN: 978-1-60807-207-1.

[17] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '1. introduction,' in *Dive Into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[18] Y. Goodfellow, Y. Bengio and A. Courville, '5 machine learning basics,' in *Deep Learning*, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[19] Y. Goodfellow, Y. Bengio and A. Courville, '6 deep feedforward networks,' in *Deep Learning*, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[20] M. A. Nielsen, 'The architecture of neural networks,' in *Neural Networks and Deep Learning*, Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[21] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '5.1 multilayer pereceptrons,' in *Dive Into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[22] M. A. Nielsen, 'Learning with gradient descent,' in *Neural Networks and Deep Learning*, Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[23] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '12.3 gradient descent,' in *Dive Into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[24] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '12.10 adam,' in *Dive Into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[25] Y. Goodfellow, Y. Bengio and A. Courville, '5.2 capacity, overfitting and underfitting,' in *Deep Learning*, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[26] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '3.6 generalization,' in *Dive Into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[27] Y. Goodfellow, Y. Bengio and A. Courville, '5.3 hyperparameters and validation sets,' in *Deep Learning*, MIT Press, 2016. [Online]. Available: https://www.deeplearningbook.org/.

[28] D. Tomar, 'Performance estimation and generalisation beyond the development dataset,' Lecture 7, IN4310 Spring 2023, 2023.

[29] F. Chollet, '8 introduction to deep learning for computer vision,' in *Deep Learning with Python*, 2nd ed., Manning Publications, Dec. 2021, ISBN: 1-61729-686-4.

[30] M. A. Nielsen, 'Introducing convolutional network,' in *Neural Networks and Deep Learning*, Determination Press, 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/.

[31] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '7. convolutional neural networks,' in *Dive into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[32] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '1.3.1.2. classification,' in *Dive into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[33] Y. Amit, P. Felzenszwalb and R. Girshick, 'Object detection,' in *Computer Vision*, Cham: Springer International Publishing, 2020, pp. 1–9, ISBN: 978-3-030-03243-2. DOI: 10.1007/978-3-030-03243-2_660-1. [Online]. Available: http://link.springer.com/10.1007/978-3-030-03243-2_660-1 (visited on 21/09/2023).

[34] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '14.3. object detection and bounding boxes,' in *Dive into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[35] D. Tomar, 'Object detection,' Lecture 12, IN4310 Spring 2023, 20th Apr. 2023. [Online]. Available: https://www.uio.no/studier/emner/matnat/ifi/IN3310/v23/lecture-materials/in4310_2023_slides_object_detection.pdf.

[36] H. Rezatofighi, N. Tsoi, J. Gwak, A. Sadeghian, I. Reid and S. Savarese, *Generalized intersection over union: A metric and a loss for bounding box regression*, 14th Apr. 2019. arXiv: 1902.09630[cs]. [Online]. Available: http://arxiv.org/abs/1902.09630 (visited on 21/09/2023).

[37] A. Zhang, Z. C. Lipton, M. Li and A. J. Smola, '14.4.2. intersection over union (IoU),' in *Dive into Deep Learning*, Cambridge University Press, 2023. [Online]. Available: http://d2l.ai/.

[38] A. Rosebrock. 'Intersection over union (IoU) for object detection,' PyImageSearch. (7th Nov. 2016), [Online]. Available: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/ (visited on 28/09/2023).

[39] Q. Zhang and Y. Liu, 'An improved algorithm for PRI modulation recognition,' in *2017 IEEE International Conference on Signal Processing, Communications and Computing (ICSPCC)*, IEEE, Oct. 2017, pp. 1–5, ISBN: 978-1-5386-3142-3. DOI: 10.1109/ICSPCC.2017.8242587. [Online]. Available: https://ieeexplore.ieee.org/document/8242587 (visited on 15/11/2023).

[40] M. Ahmadi and K. Mohamedpour, 'PRl modulation type recognition using level clustering and autocorrelation,' *American Journal of Signal Processing*, vol. 2, no. 5, pp. 83–91, 1st Dec. 2012, ISSN: 2165-9354. DOI: 10.5923/j.ajsp.20120205.01. [Online]. Available: http://article.sapub.org/10.5923.j.ajsp.20120205.01.html (visited on 05/12/2023).

[41] K.-H. Song, D.-W. Lee, J.-W. Han and B.-K. Park, 'Pulse repetition interval modulation recognition using symbolization,' in *2010 International Conference on Digital Image Computing: Techniques and Applications*, Sydney, Australia: IEEE, Dec. 2010, pp. 540–545, ISBN: 978-1-4244-8816-2. DOI: 10.1109/DICTA.2010.96. [Online]. Available: http://ieeexplore.ieee.org/document/5692617/ (visited on 05/12/2023).

[42] H. P. K. Nguyen, H. Q. Nguyen and D. Ngo, 'Deep learning for pulse repetition interval classification:' in *Proceedings of the 8th International Conference on Pattern Recognition Applications and Methods*, Prague, Czech Republic: SCITEPRESS - Science and Technology Publications, 2019, pp. 313–319, ISBN: 978-989-758-351-3. DOI: 10.5220/0007253203130319. [Online]. Available: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0007253203130319 (visited on 22/08/2023).

[43] A. Svensson, 'Classification of radar emitters based on pulse repetition interval using machine learning,' 2022. [Online]. Available: https://urn.kb.se/resolve?urn=urn:nbn:se:kth:diva-320777.

[44] 'PyTorch docs GaussianBlur,' GaussianBlur. (), [Online]. Available: https://pytorch.org/vision/main/generated/torchvision.transforms.GaussianBlur.html (visited on 23/10/2023).

[45] X. Ying, 'An overview of overfitting and its solutions,' *Journal of Physics: Conference Series*, vol. 1168, p. 022 022, Feb. 2019, ISSN: 1742-6588, 1742-6596. DOI: 10.1088/1742-6596/1168/2/022022. [Online]. Available: https://iopscience.iop.org/article/10.1088/1742-6596/1168/2/022022 (visited on 21/09/2023).

[46] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan and S. Belongie, *Feature pyramid networks for object detection*, 19th Apr. 2017. arXiv: 1612.03144[cs]. [Online]. Available: http://arxiv.org/abs/1612.03144 (visited on 21/09/2023).

[47] A. Krizhevsky, I. Sutskever and G. E. Hinton, 'Imagenet classification with deep convolutional neural networks,' in *Advances in Neural Information Processing Systems*, F. Pereira, C. Burges, L. Bottou and K. Weinberger, Eds., vol. 25, Curran Associates, Inc., 2012. [Online]. Available: https://proceedings.neurips.cc/paper_files/paper/2012/file/c399862d3b9d6b76c8436e924a68c45b-Paper.pdf.

[48] A. Krizhevsky, *One weird trick for parallelizing convolutional neural networks*, 2014. arXiv: 1404.5997 [cs.NE].

[49] 'PyTorch docs AlexNet,' AlexNet. (), [Online]. Available: https://pytorch.org/vision/main/models/generated/torchvision.models.alexnet.html (visited on 23/10/2023).

[50] G. Huang, Z. Liu, L. van der Maaten and K. Q. Weinberger, *Densely connected convolutional networks*, 28th Jan. 2018. arXiv: 1608.06993[cs]. [Online]. Available: http://arxiv.org/abs/1608.06993 (visited on 23/10/2023).

Appendix A

Diagram of Detection Models

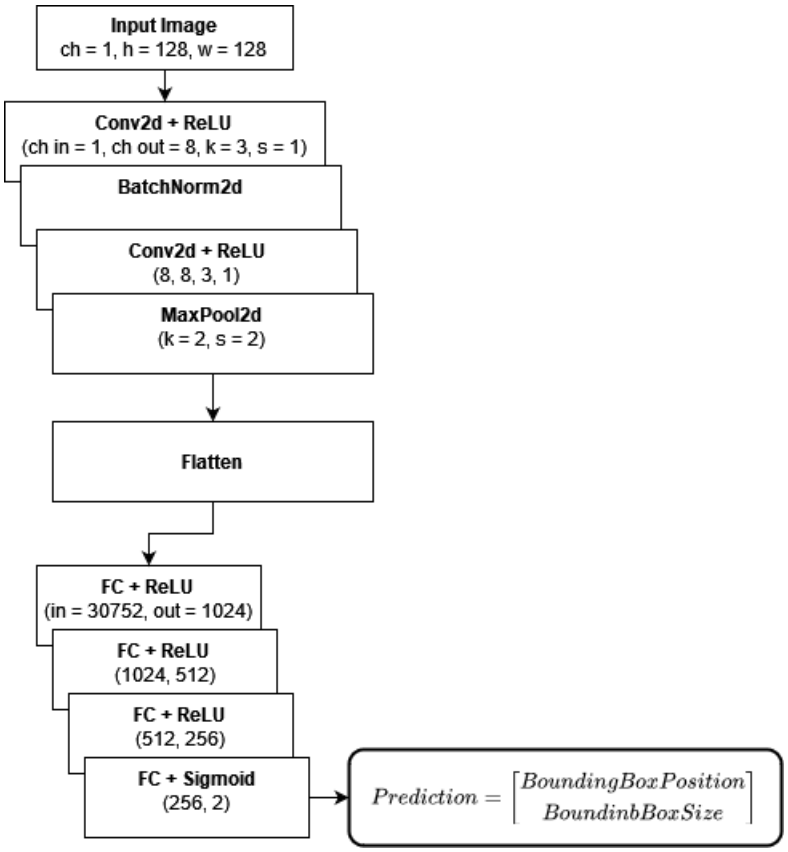


Figure A.1: Diagram of Conv1 Model

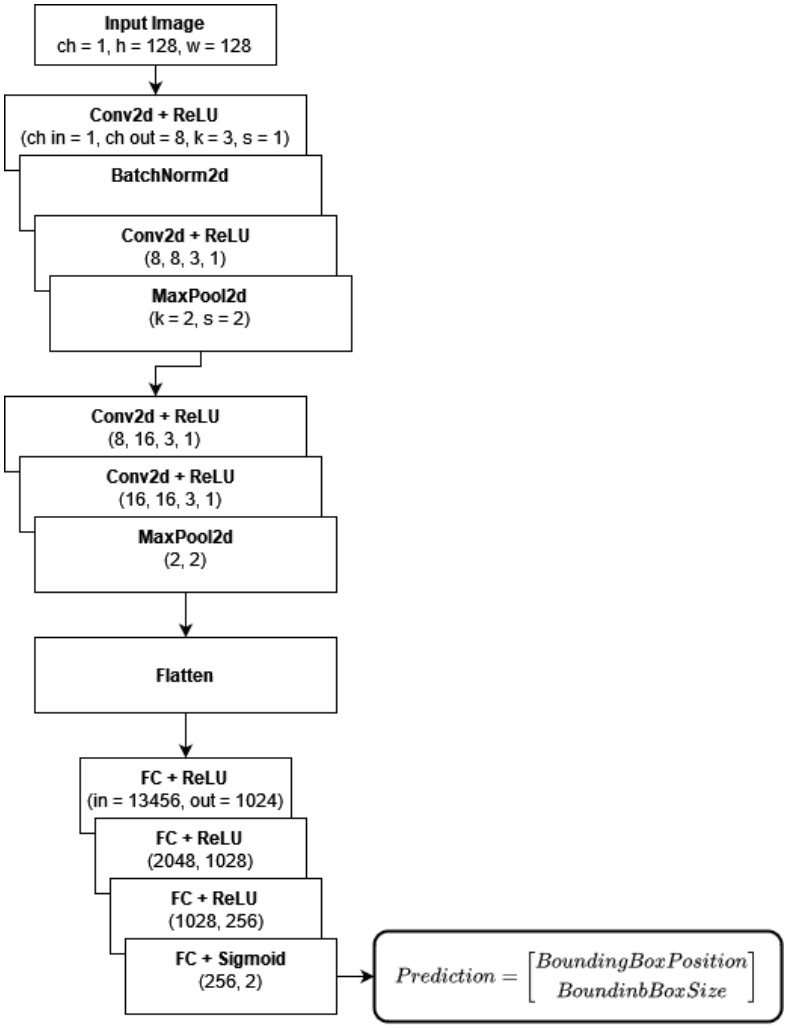


Figure A.2: Diagram of Conv2 Model

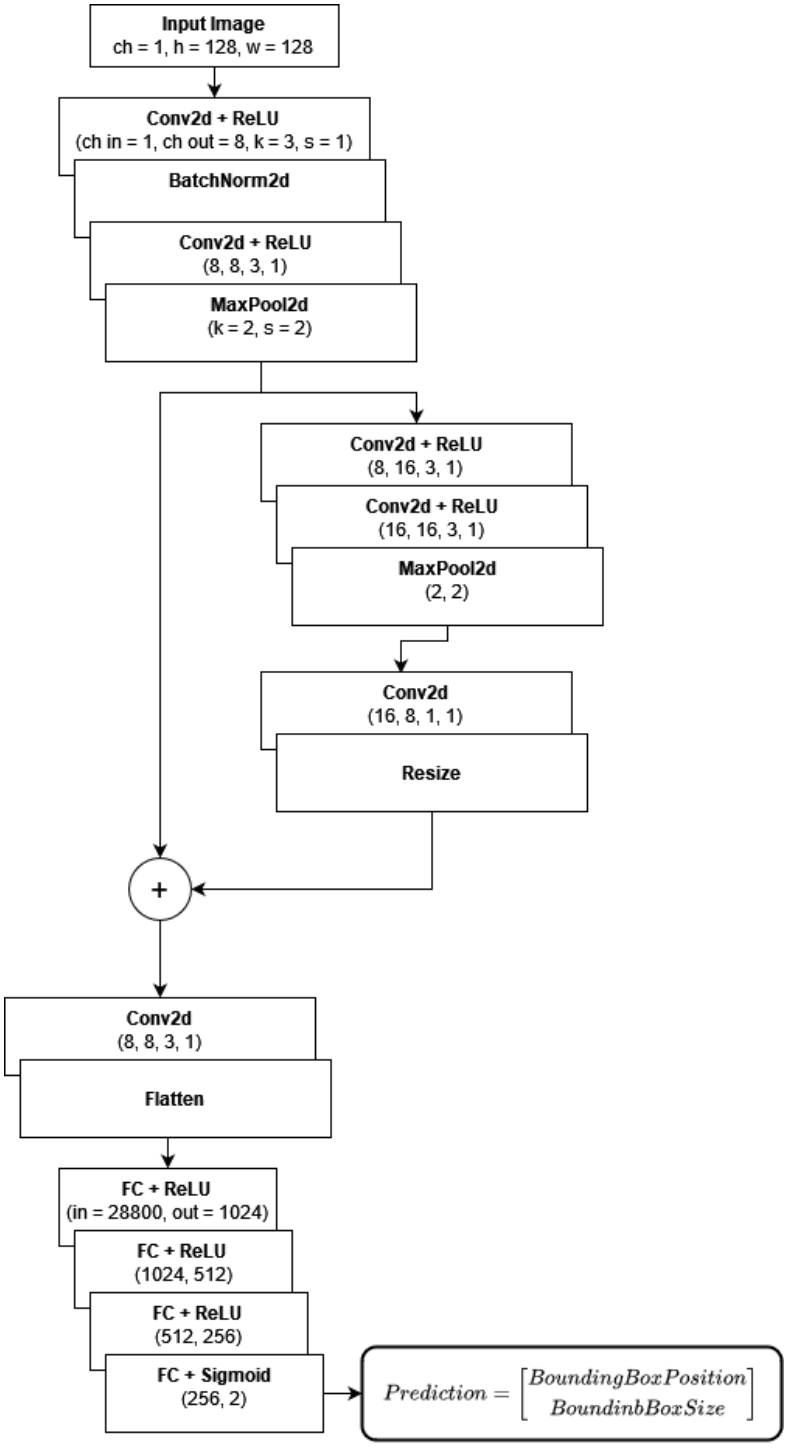


Figure A.3: Diagram of Conv2 Model with feature fusion

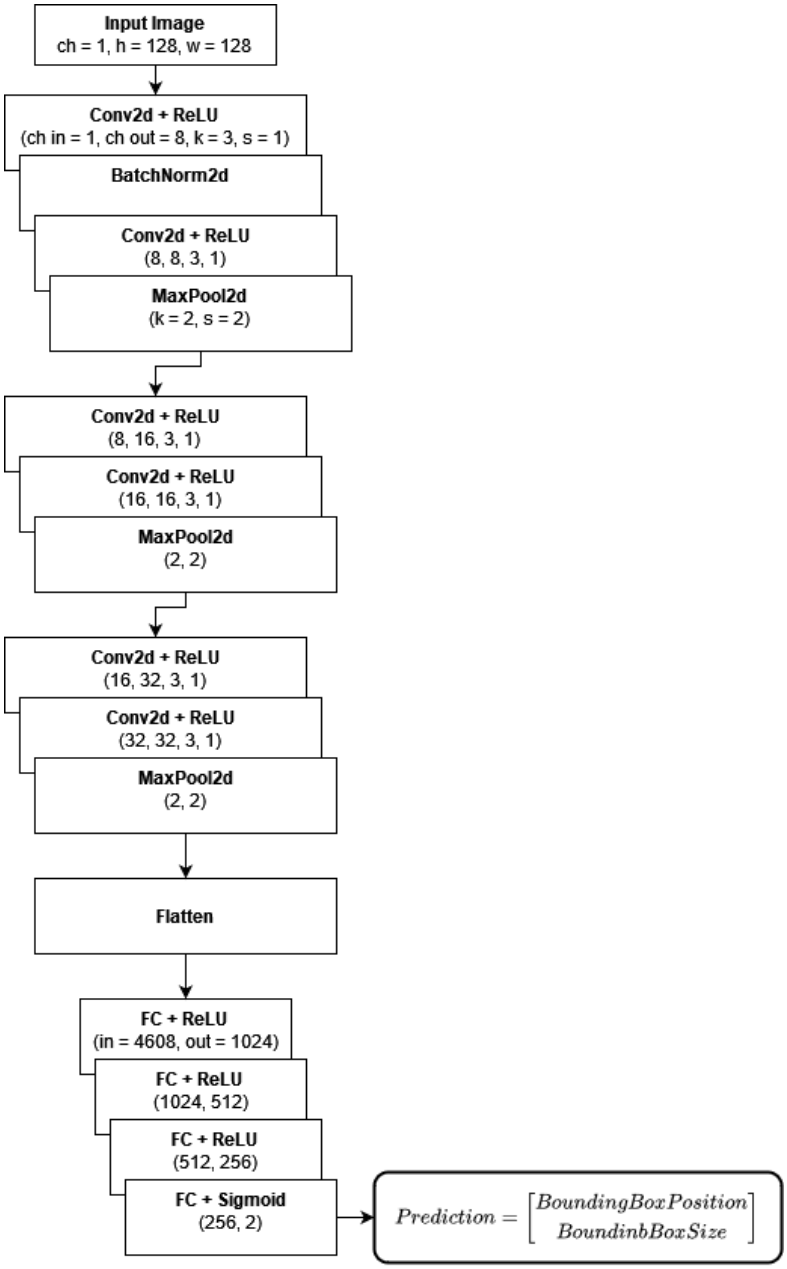


Figure A.4: Diagram of Conv3 Model

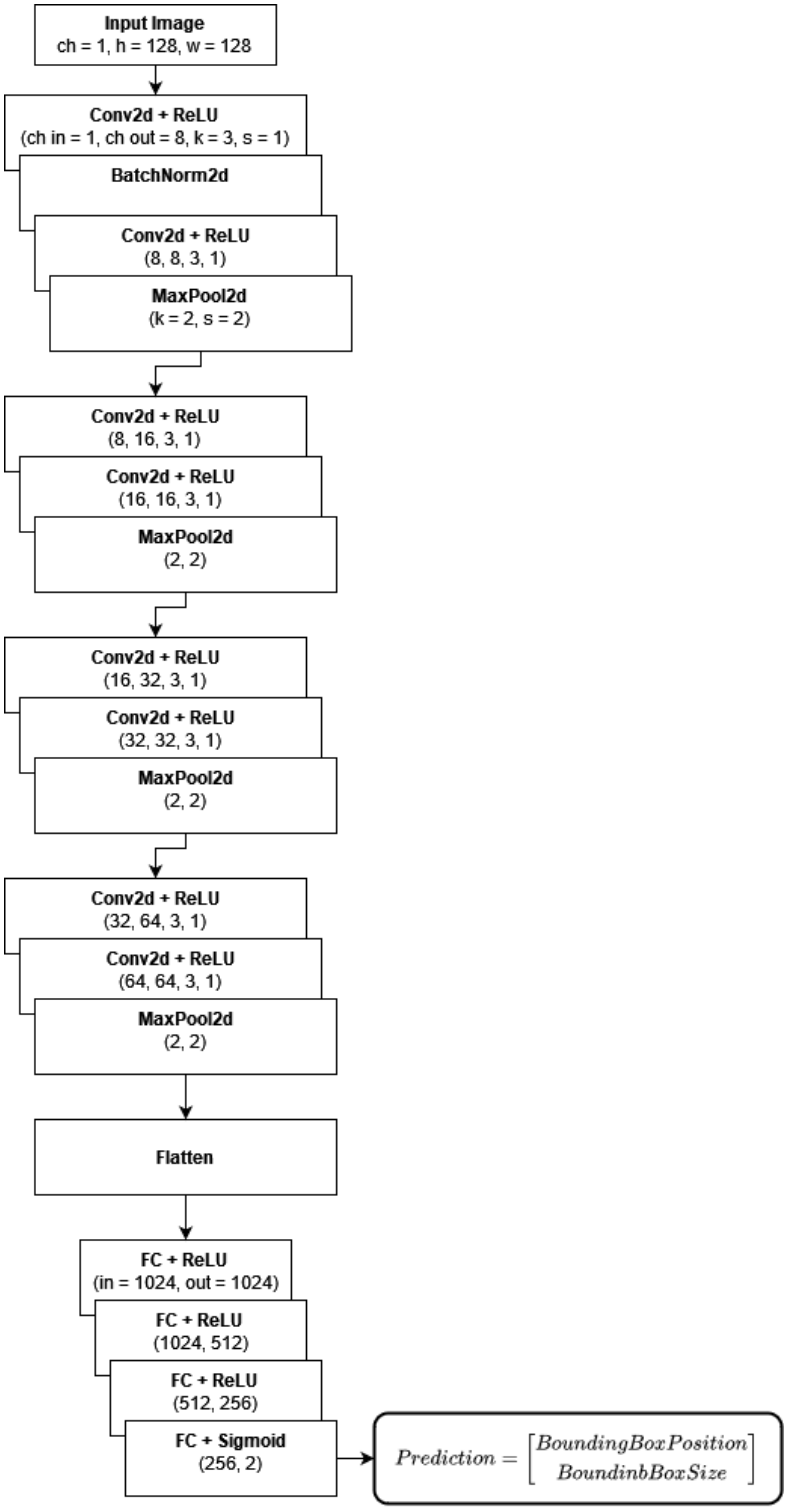


Figure A.5: Diagram of Conv4 Model

Appendix B

Classwise Metric Distribution of Detection Models on Validation Set

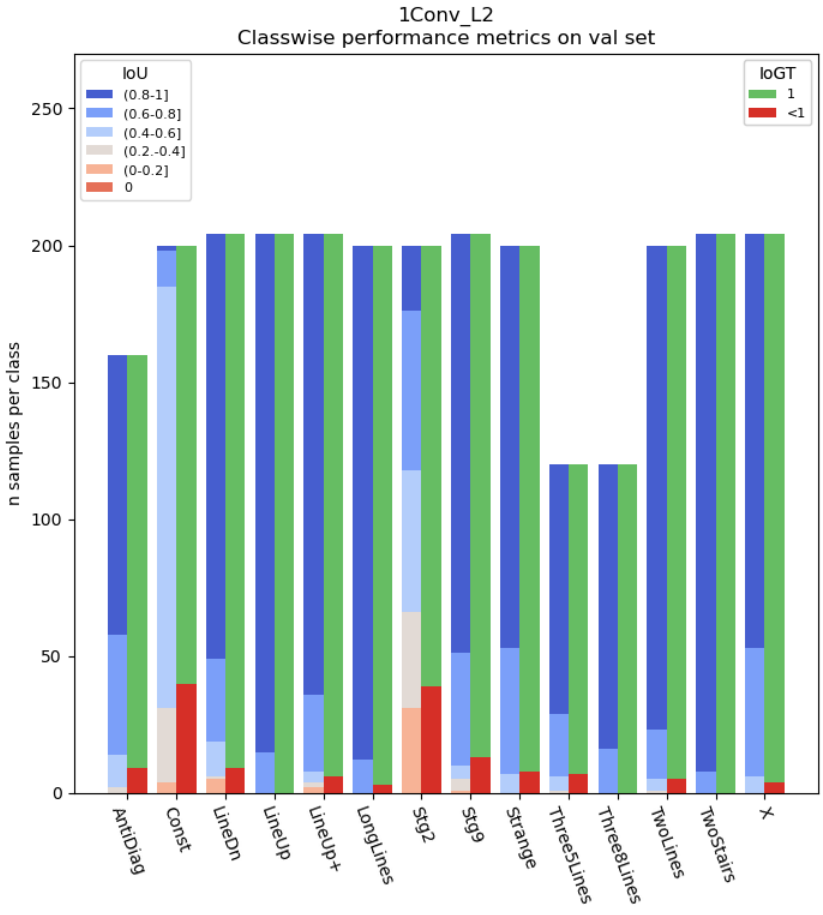


Figure B.1: Classwise metric distribution of 1Conv trained with L2 Loss

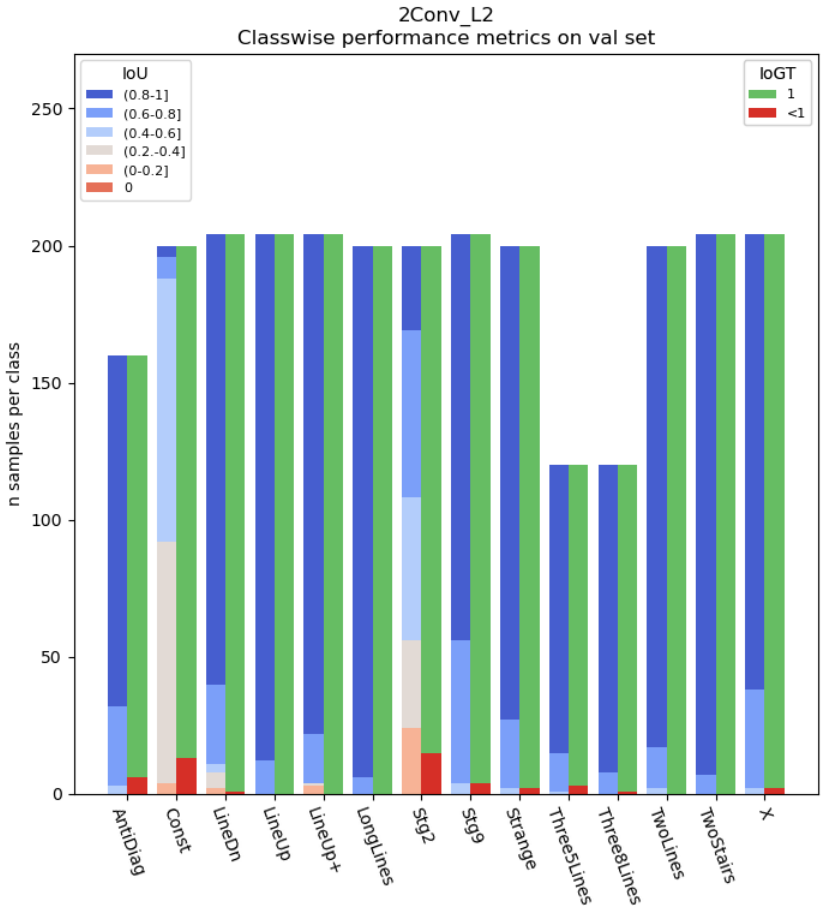


Figure B.2: Classwise metric distribution of 2Conv trained with L2 Loss

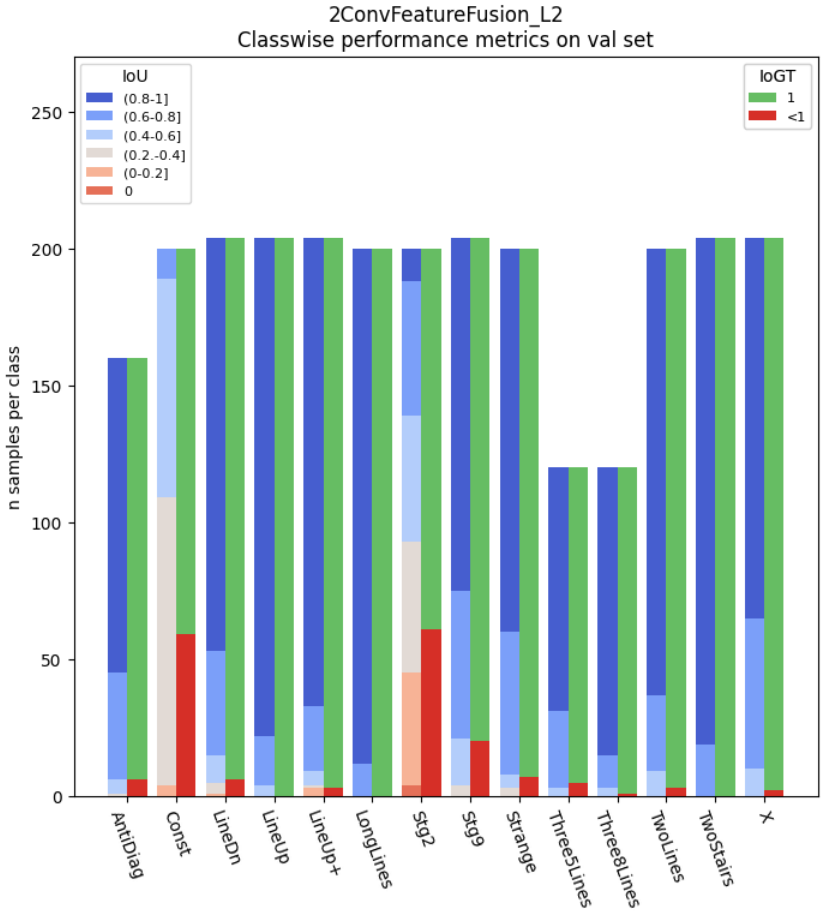


Figure B.3: Classwise metric distribution of 2Conv with feature fusion trained with L2 Loss

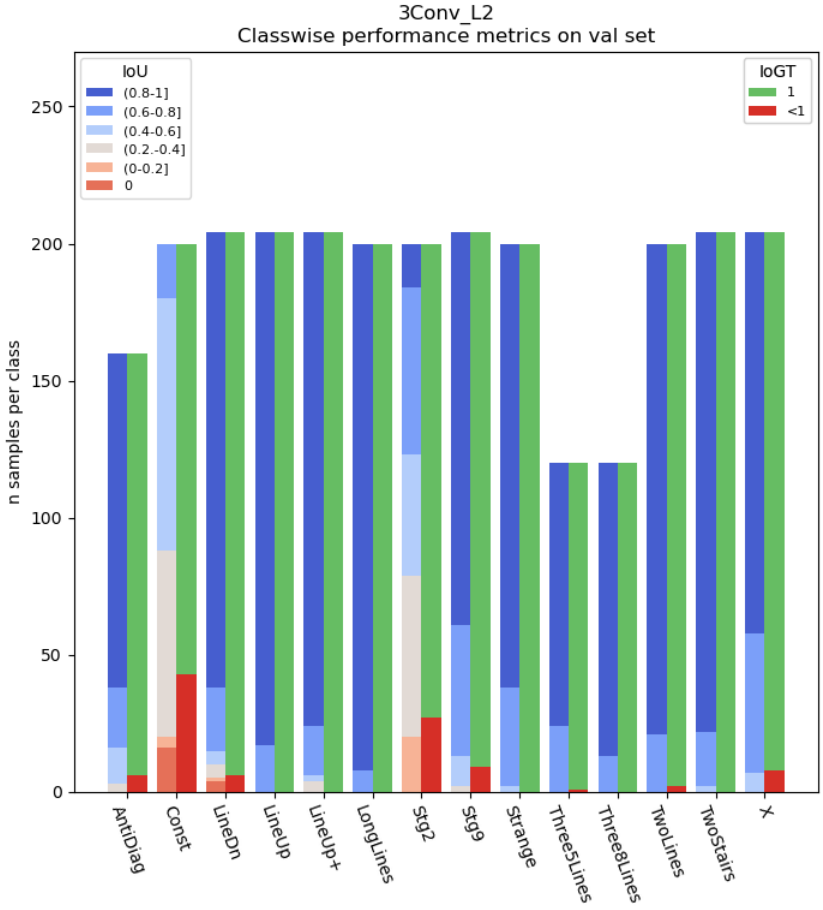


Figure B.4: Classwise metric distribution of 3Conv trained with L2 Loss

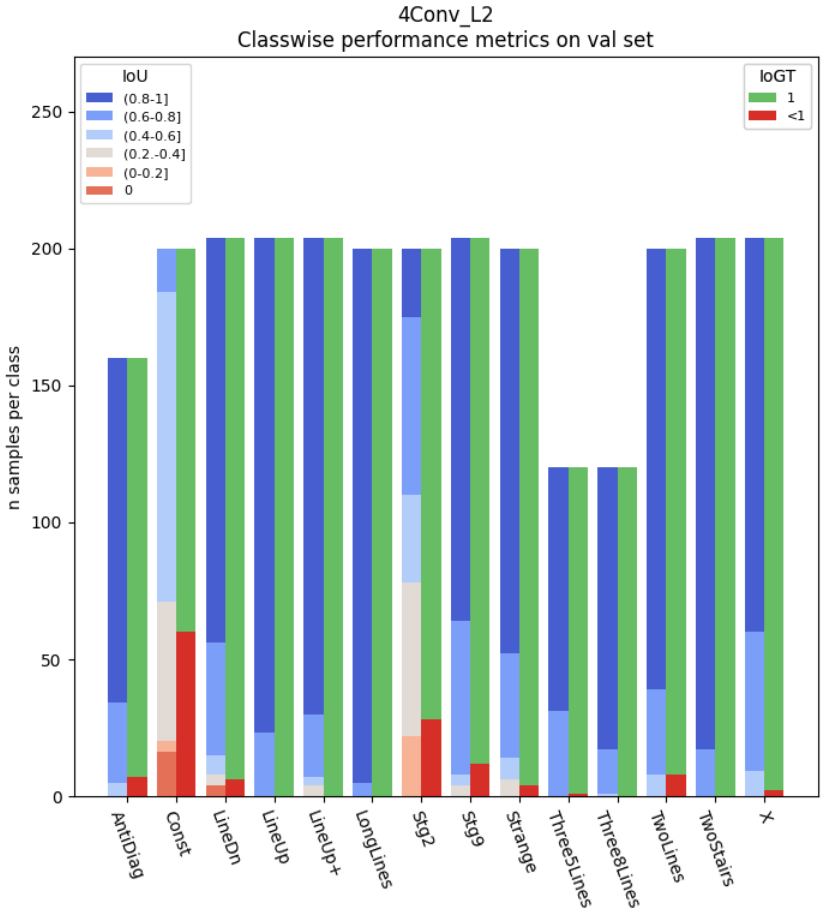


Figure B.5: Classwise metric distribution of 4Conv trained with L2 Loss

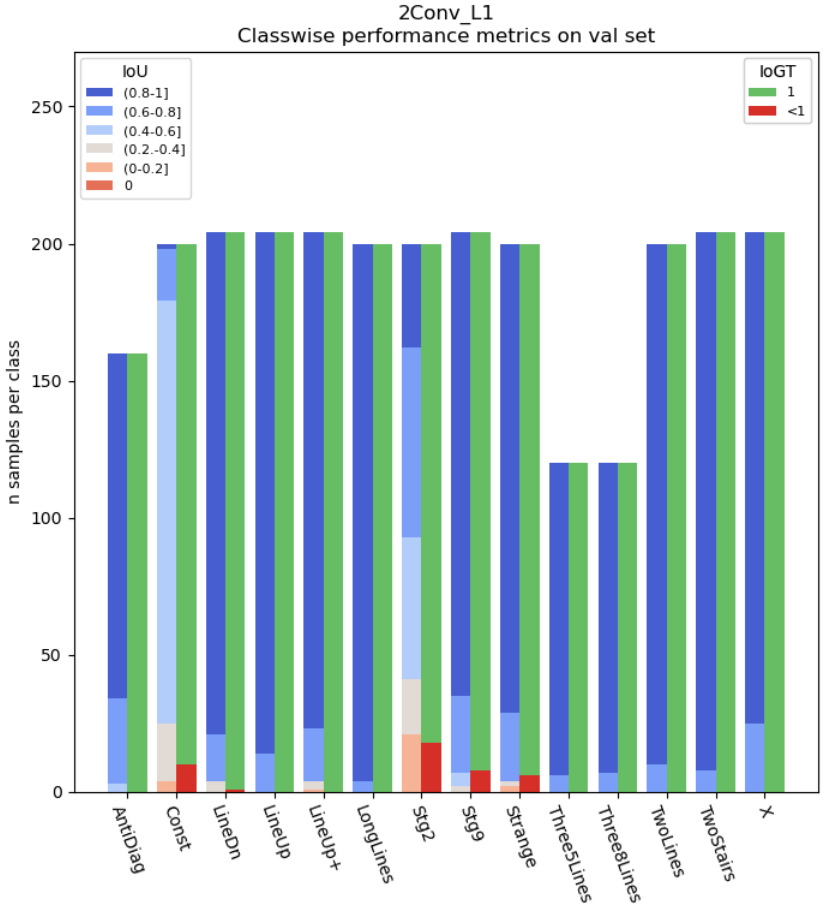


Figure B.6: Classwise metric distribution of 2Conv trained with L1 Loss

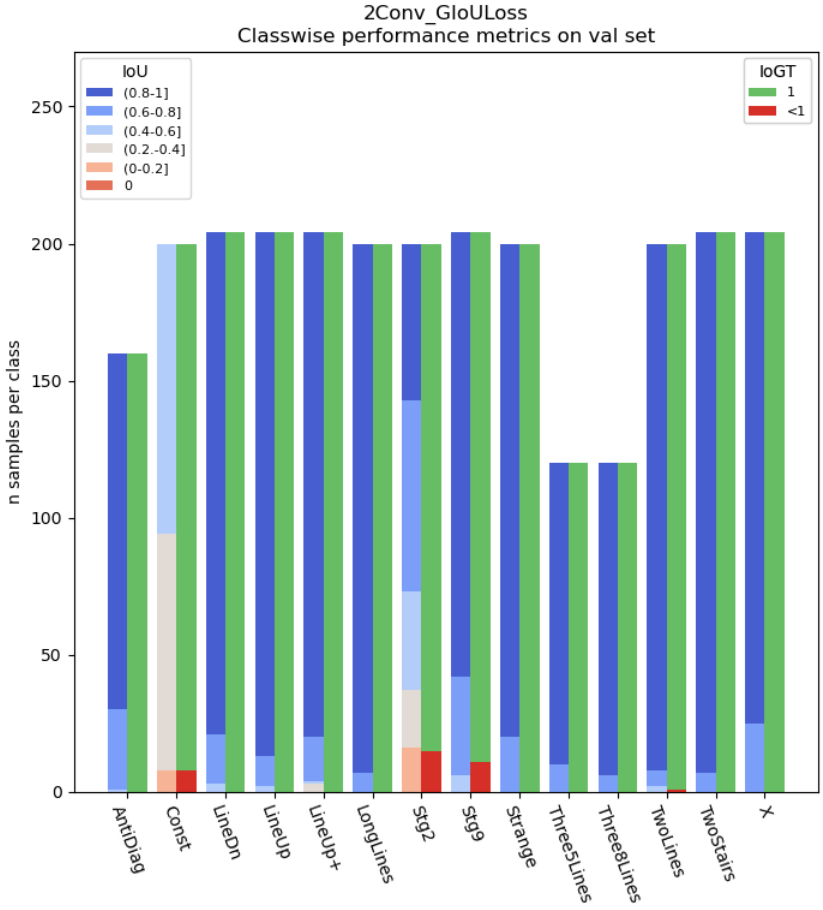
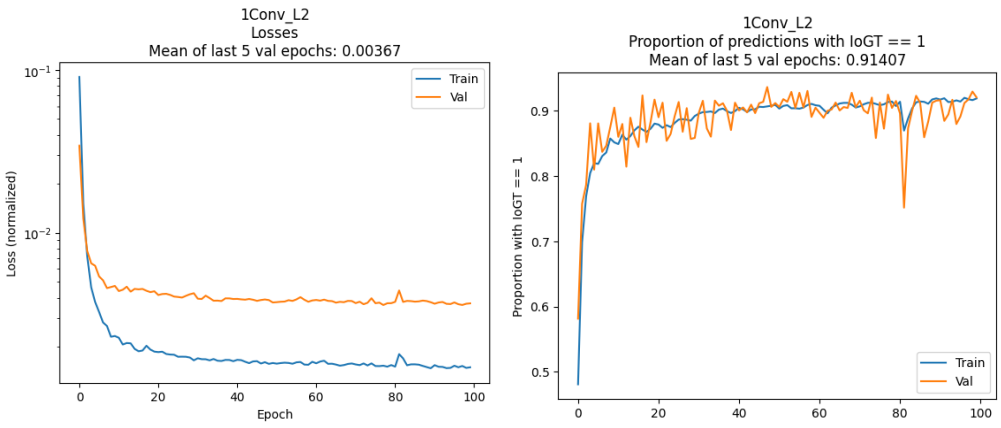


Figure B.7: Classwise metric distribution of 2Conv trained with GIoU Loss

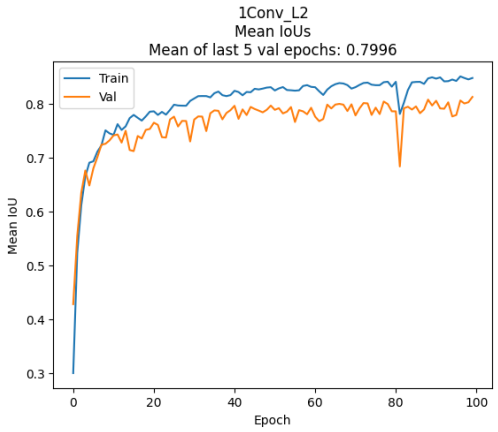
Appendix C

Training and Validation Curves of the Detection Models



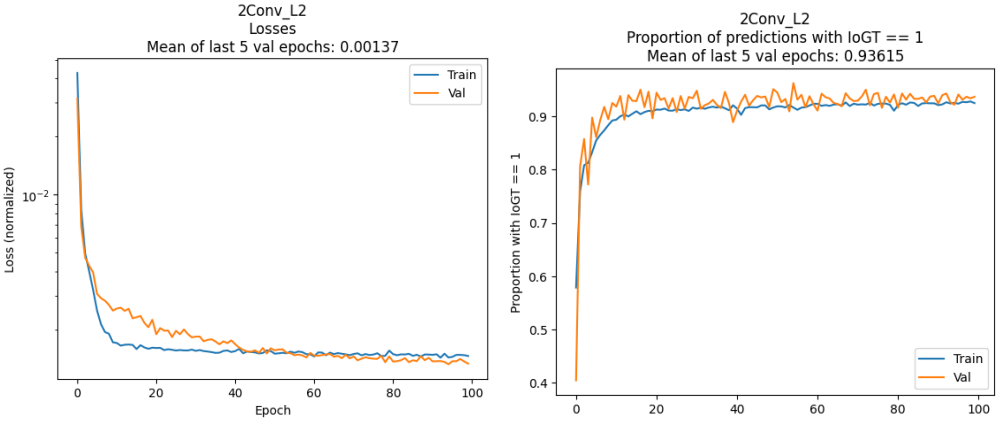
(a) Loss Curves

(b) IoGT Score Curves



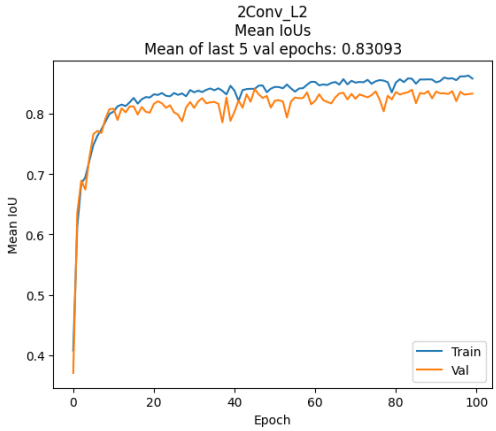
(c) IoU Curves

Figure C.1: Training and validation curves of Conv1, trained with L2 loss



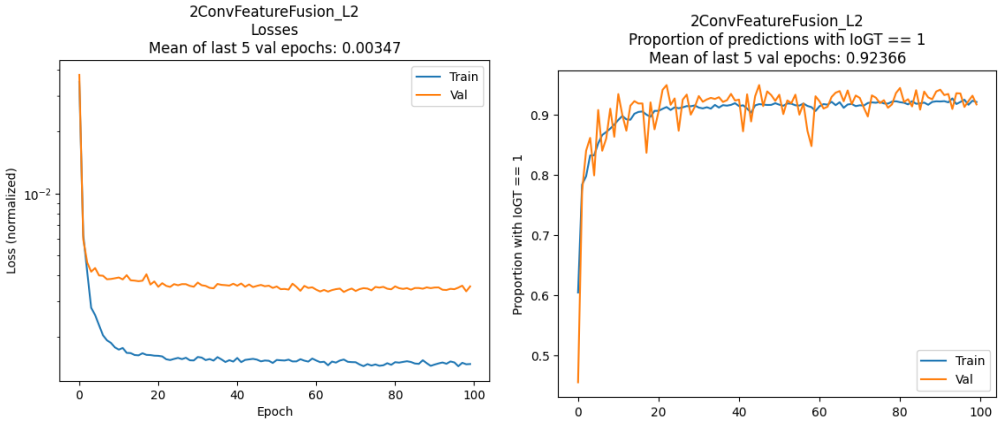
(a) Loss Curves

(b) IoGT Score Curves



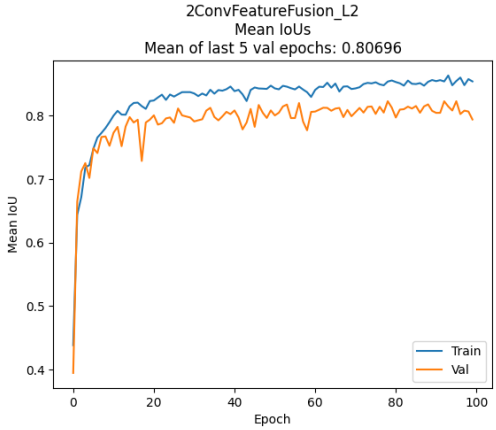
(c) IoU Curves

Figure C.2: Training and validation curves of Conv2, trained with L2 loss



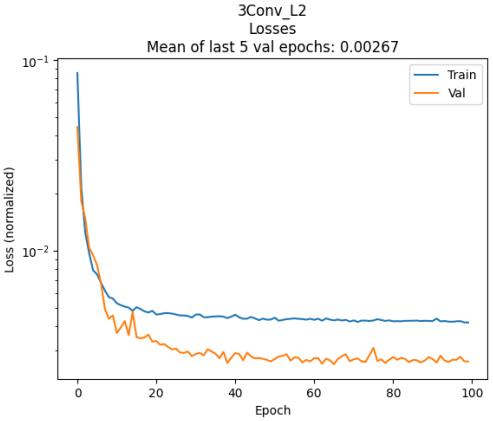
(a) Loss Curves

(b) IoGT Score Curves

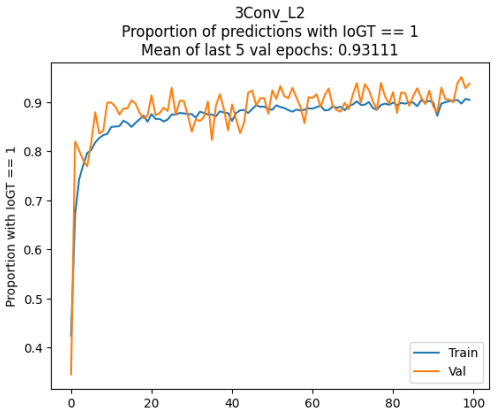


(c) IoU Curves

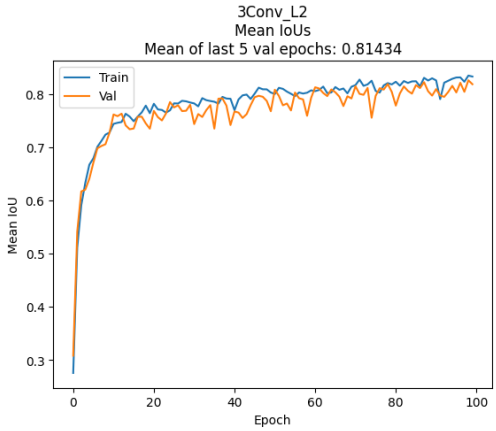
Figure C.3: Training and validation curves of Conv2 with feature fusion, trained with L2 loss



(a) Loss Curves

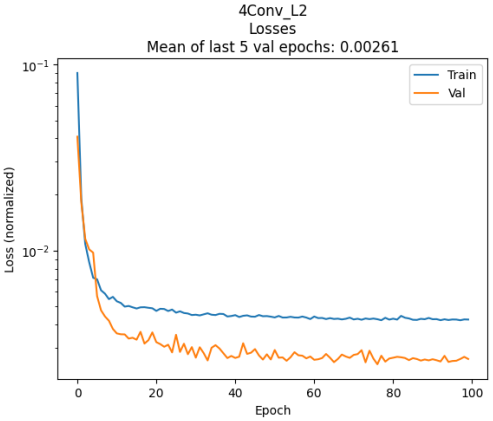


(b) IoGT Score Curves

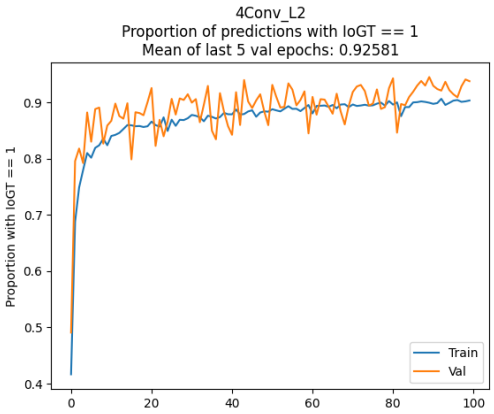


(c) IoU Curves

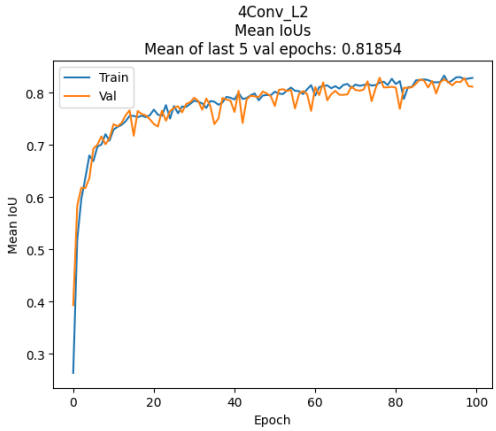
Figure C.4: Training and validation curves of Conv3, trained with L2 loss



(a) Loss Curves

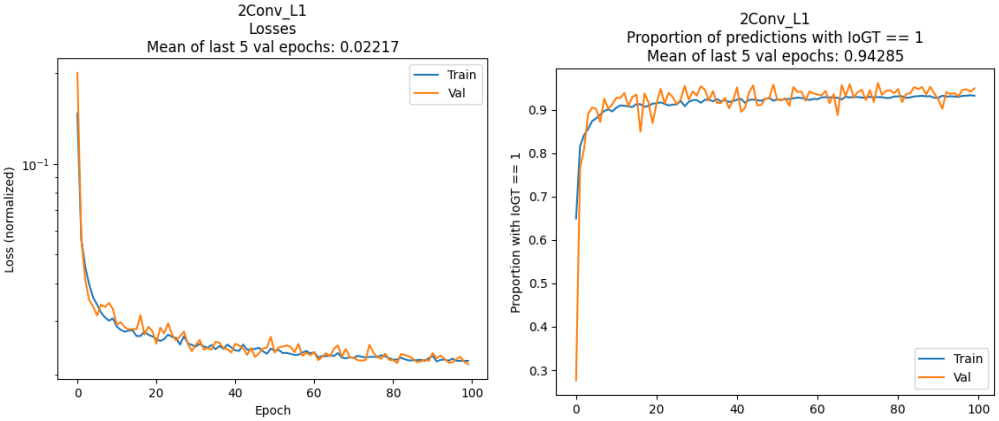


(b) IoGT Score Curves



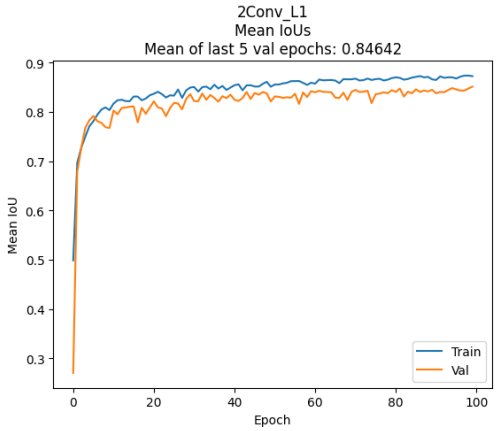
(c) IoU Curves

Figure C.5: Training and validation curves of Conv4, trained with L2 loss



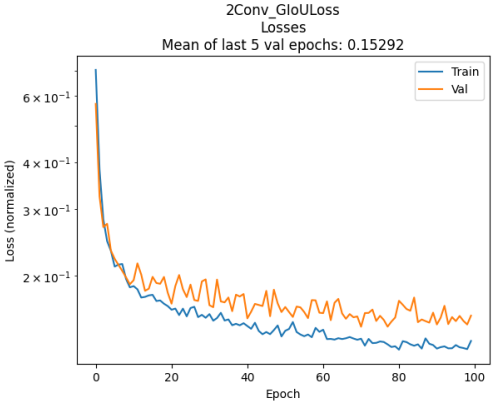
(a) Loss Curves

(b) IoGT Score Curves

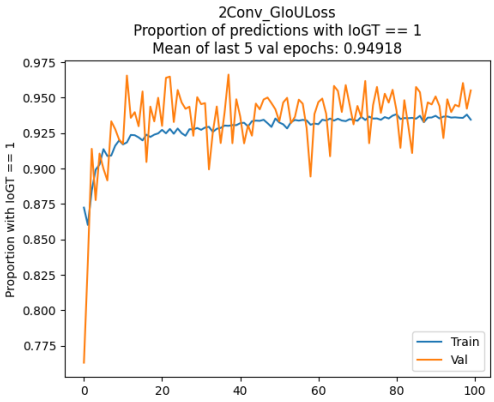


(c) IoU Curves

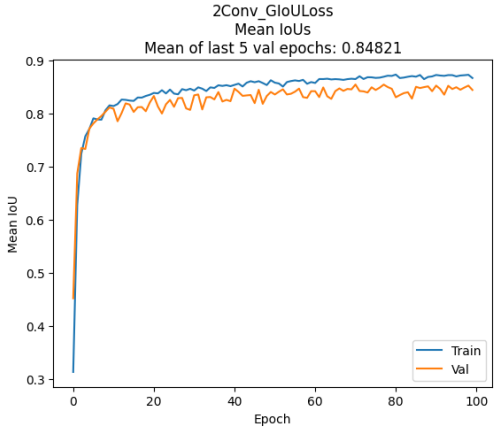
Figure C.6: Training and validation curves of Conv2, trained with L1 loss



(a) Loss Curves



(b) IoGT Score Curves

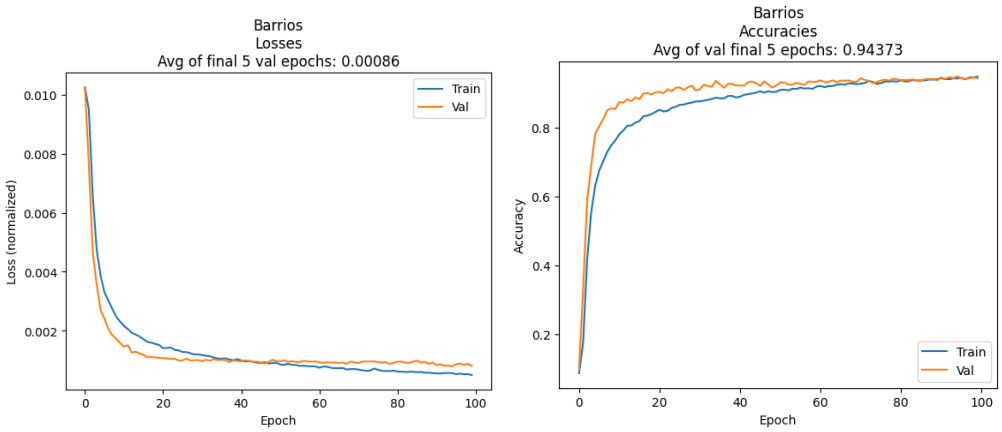


(c) IoU Curves

Figure C.7: Training and validation curves of Conv2, trained with GIoU loss

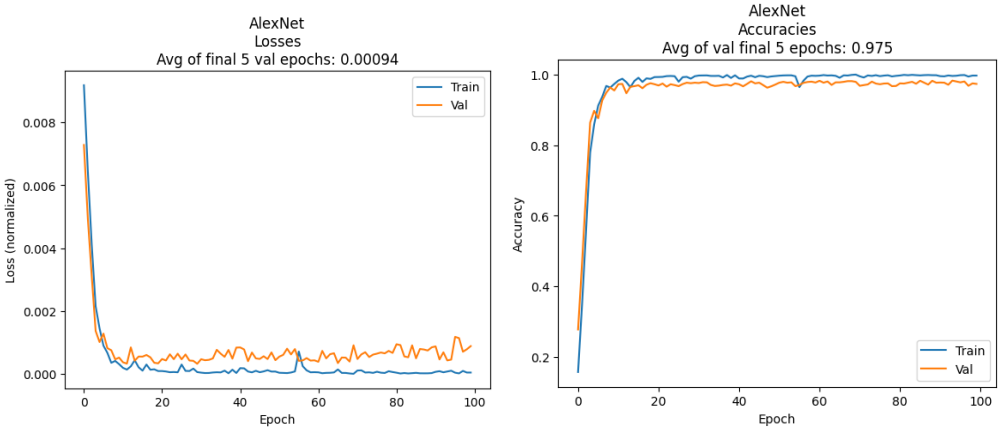
Appendix D

Training and Validation Curves of the Classification Models



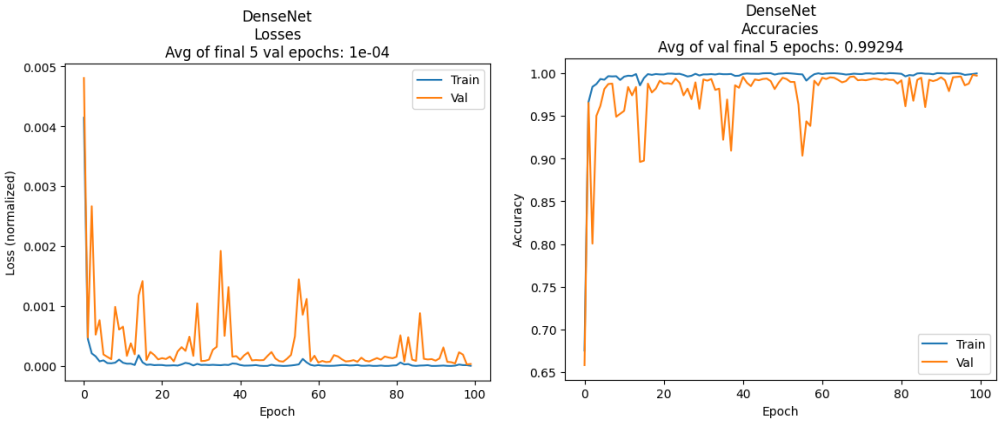
(a) Loss Curves (b) Acuracy

Figure D.1: Training and validation curves of Barrios' model



(a) Loss Curves (b) Acuracy

Figure D.2: Training and validation curves of AlexNet



(a) Loss Curves (b) Acuracy

Figure D.3: Training and validation curves of DenseNet