Utrecht University

Department of Information and Computing Science

Applied Data Science Master Thesis

Illegal Object Detection in Neutron Images from Dynaxion Scanning System using Faster R-CNN

First examiner
prof. dr. G.T. Barkema
G.T.Barkema@uu.nl

Second examiner
dr. M.J.S. Brinkhuis
m.j.s.brinkhuis@uu.nl

Candidate
Ane Dicks Herrán
a.dicksherran@students.uu.nl

In cooperation with
Dynaxion - Koen van den Brandt
koen.vandenbrandt@dynaxion.nl
Juan Moreno Diez
j.morenodiez@students.uu.nl

July 5th, 2023

Disclamer

The data used in this study is entirely simulated, ensuring the exclusion of any privacy concerns associated with real-world personal information. Furthermore, the model developed within this research is designed to be privacy insensitive, solely focused on distinguishing between the legality of objects within the images. This deliberate approach was adopted as a precautionary measure to mitigate potential privacy issues in the future.

Abstract

Nowadays, inspection of baggage for threat objects (such as explosives) using images is of interest as it can contribute towards more effective and advanced security screening systems. As neutron imaging presents certain advantages over traditional X-Ray imaging, Dynaxion is a company located in Eindhoven that designed a screening system based on neutron generation. Moreover, the checking of baggage is partially based on manual detection. While this can be efficient, it is also costly and leaves window open for human error. Therefore, this study aims at optimizing a Faster R-CNN that can accurately detect and classify illegal objects contained in neutron images that were generated from Dynaxion. For this, six Faster R-CNN models were trained based on different techniques of feature extraction (ResNet50 and MobileNet-v3), optimizers (Adam or SGD), learning rates, batch sizes, number of epochs and weight decays. In conclusion, the best detection was achieved by the combination of the SGD optimizer and the ResNet50 feature extractor, achieving an overall F1 score of 0.85. Sample images on a test dataset demonstrate the model's accurate object detection capabilities.

Contents

1. Introduction

1.1 Motivation and context

Millions of suitcases and parcels move around the world every day. While in every country one of the biggest priorities is security, current screening safety methods cannot provide complete certainty as to what is inside; whether explosives, drugs weapons or simply a pile of clothes. This uncertainty around what is actually being transported presents a real risk to safety, health and society [1].

The current scanning systems in use rely on traditional X-ray imaging. While X-Ray imaging has been proven to be effective, it also presents its limitations [2]. Neutron imaging, on the other hand, offers certain advantages. They have a greater ability to penetrate dense and shielded materials more effectively than X-Rays, allowing to better detect substances that may intentionally be occluded within an object. Additionally, neutrons have different interaction properties with matter compared to X-Rays, as they are more sensitive to light elements like hydrogen and nitrogen [3]. This is advantageous because illegal substances often contain hydrogen-rich or nitrogen-rich compounds [4]. By considering these advantages, it can be concluded that neutron imaging can provide additional information to help detect objects and their materials more accurately.

In line with these insights, Dynaxion is a company based at the High Tech Campus in Eindhoven that is developing a scanning system using neutrons to determine the contents inside parcels and suitcases. Their system is based on particle acceleration and neutron generation technology. As parcels pass through the scanning system, the particle accelerator creates a neutron beam towards the parcel and irradiates it. During this process, the neutrons that pass through the material of the object without interacting or being absorbed, known as transmitted neutrons, can be quantified or measured. This process is called neutron transmission and allows the detection of neutrons in a position-sensitive way. By measuring the intensity of transmitted neutrons and analyzing their energy distribution, researchers can gather information about the material's composition, density, and interaction properties with neutrons [5].

Neutron images are created based on the interactions of neutrons with the material. Dark spots in the resulting neutron images arise from regions of the material that strongly interact with neutrons and where the material exhibits high neutron absorption, resulting in fewer neutrons being transmitted and measured behind those regions. Conversely, regions that appear lighter in the neutron image indicate areas where the material has a lower interaction with neutrons. These regions allow a larger proportion of neutrons to pass through without substantial absorption, resulting in a higher number of transmitted neutrons being detected. Therefore, in neutron transmission imaging, the contrast between light and dark spots provides information about the material's interaction with neutrons.

Neutron imaging provides images of what is inside the parcel but there is still the need for those images to be interpreted. Automation is nowadays mainly based on scanning luggage or packages into scanners and manually determine whether there is any potential thread present [6]. Nevertheless, one main drawback with this strategy is that, besides being costly, it also leaves window open for human error [7]. Therefore, automatic object-detection in neutron images is a promising approach in which nuclear physics and Artificial Intelligence (AI) algorithms can be combined to provide an accurate and automated security screening solution.

1.2 AI for object recognition

Object recognition is a fundamental task in computer vision that aims at identifying and categorizing objects within images or video sequences. It plays a crucial role in various applications, such as autonomous driving, surveillance systems and robotics [8]. In the context of object recognition, classification of neutron images can be a difficult task because images can be occluded by different objects and cannot be recognised simply once scanned [9].

Over the years, numerous methods have been proposed for object recognition. Traditional approaches relied on hand-designed features, such as Scale-Invariant Feature Transform (SIFT) or Histogram of Oriented Gradients (HOG), that were coupled with machine learning algorithms like Support Vector Machines (SVM) or Random Forests. While these methods achieved moderate success, they struggle to handle complex scenes involving scale variations or object occlusions [10]. More recently, deep learning-based approaches, such as You Only Look Once (YOLO), Faster Region-based Convolutional Neural Network (R-CNN) and Single Shot Detectors (SSD) have emerged as a powerful tool for automatically learning discriminative features from pixel data. They are an extended approach that overcome the limitation of traditional Convolutional Neural Networks (CNNs) when it comes to effectively managing object detection scenarios involving with multiple objects in an image [11].

Specifically, Faster R-CNN addresses this limitation by introducing a region proposal network (RPN) that generates potential object regions, known as bounding box proposals, within an image. These proposals are then classified and refined to obtain the final object detection [12]. This enables Faster R-CNN to handle object detection and localization in complex scenarios involving occlusions and multiple objects, making it suitable for object recognition in neutron images.

To demonstrate the effectiveness of Faster R-CNN, several studies have been conducted where Faster R-CNN was chosen as the method for the object recognition task. Sa et al. showed that one can achieve much better performance using Faster R-CNN compared to traditional sliding window detection method using hand-crafted features. In their work, they achieved average precision of 0.905 (as compared to 0.061 average precision using traditional approaches) when fine tuning the network using 974 lumbar X-Ray images [13]. A different study from Koçi et al. performed object recognition for baggage inspection for threat objects. When comparing Single Shot Detector (SSD), R-FCN and Faster R-CNN; the best detection was achieved by the combination of Faster R-CNN and ResNet feature extractor, achieving an accuracy of 87.58% [14].

1.3 Objectives and significance of the study

Image-based detection of objects in neutron images is not yet so common and is therefore an interesting approach. Studies have shown that AI algorithms have the ability to provide a promising solution for the control of illicit objects in parcels or suitcases undergoing the scanning process. Therefore, it is of great interest to analyse the data from the Dynaxion scanning system to train and optimize a Faster R-CNN model that can accurately determine the spatial extent of the objects in the neutron images as well as the certainty level of whether it is a legal or illegal object.

The significance of this study lies in its potential to contribute to the development of more effective and advanced security screening systems. Moreover, training a Faster R-CNN model could enhance the accuracy of object detection in neutron images, enabling better differentiation between legal and illegal objects. The successful implementation of this model could have broader implications beyond object detection in neutron images. It could potentially be applied to other fields, such as medical imaging, where accurate object analysis is fundamental. Ultimately, this study could also contribute to enhancing overall safety and security in various domains. This includes areas such as transportation or border control, where the detection of illicit objects is crucial for public safety and the prevention of illegal activities.

2. Neural Network Architecture

Having mentioned the effectiveness and applicability of Faster R-CNN, it is important to gain understanding of the algorithm's inner workings and delve into its individual components to make use of its full potential.

2.1 Faster R-CNN

Faster R-CNN is said to be efficient as it introduces a unified architecture that integrates region proposal generation and object detection into a single framework. The structure of Faster R-CNN can be divided into four main components: Feature extractor, Region proposal network (RPN), Region of Interest (RoI) and classifier. A scheme of the network architecture can be seen in Figure 2.1 [15].

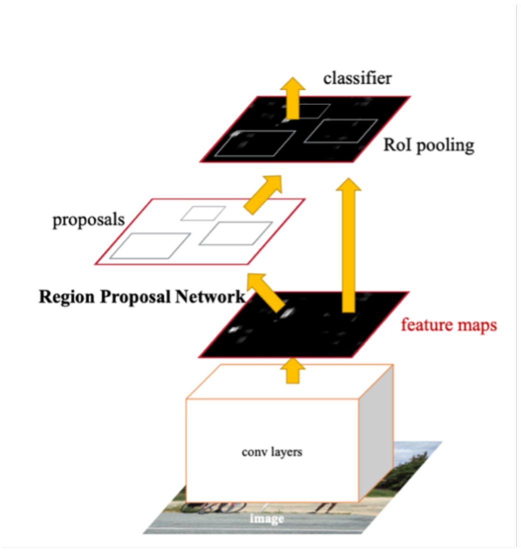


Figure 2.1: The different parts of a Faster R-CNN architecture [15]

Convolutional layers: feature extraction

Faster R-CNN starts with a backbone convolutional neural network (CNN) to extract meaningful features from the input image. These feature extraction networks are typically pre-trained on large-scale image classification datasets like ImageNet.

The obtained feature maps are used for subsequent processing. CNNs like ResNet or VGGNet have demonstrated satisfactory performance, surpassing traditional methods [16].

Region proposal network (RPN)

Subsequently, the obtained features from the convolutional layers are used to generate region proposals, which are potential bounding boxes that may contain objects. It achieves this by sliding a window of size n x n over each feature map. For each sliding window position, a set of k anchors are generated of different sizes and aspect ratios, but all having the same center point (Figure 2.2) [15]. In a typical Faster R-CNN implementation, there are k = 9 anchor boxes of 3 different scales and 3 different aspect ratios [17].

At each position, the RPN makes two kinds of predictions. First, it predicts whether an object might be present (labelled as *foreground class*) or might not be present in the proposed area (labelled as *background class*). Second, it predicts the coordinates of the proposed bounding boxes.

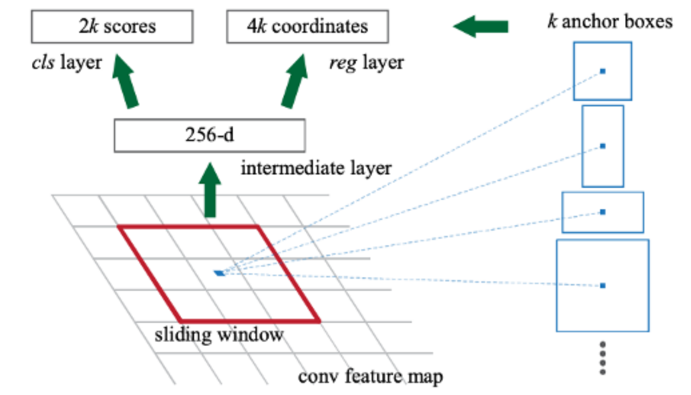


Figure 2.2: The regional proposal network (RPN) of a Faster R-CNN where different anchor boxes are used at every sliding window position

In order to select where an object is predicted to be present and which anchor box is representative for the ground-truth bounding box, the Intersection over Union (IoU) between each anchor box and the target bounding box is calculated [18]. For this, there are 4 conventional rules to select the positive and negative anchor boxes:

Rule 1. Anchor boxes are marked as positive if the Intersection over Union (IoU) with the ground-truth bounding box is greater than 0.7.

Rule 2. All anchor boxes that have an IoU with the ground-truth bounding box less than 0.3 are marked as negative.

Rule 3. If no anchor box has an IoU greater than 0.7, the anchor boxes with the largest IoU (if the IoU is greater than 0.5) are selected.

Rule 4. Anchor boxes that were not marked as either positive or negative do not contribute to the training of the model.

**Positive anchor boxes is where an object might be present. Negative anchor boxes is where an object might not be present.*

Figure 2.3 shows an example of different anchor boxes with their corresponding IoU [19]. Finally, the RPN generates a set of region proposals by adjusting the anchor boxes based on the two aforementioned predictions.

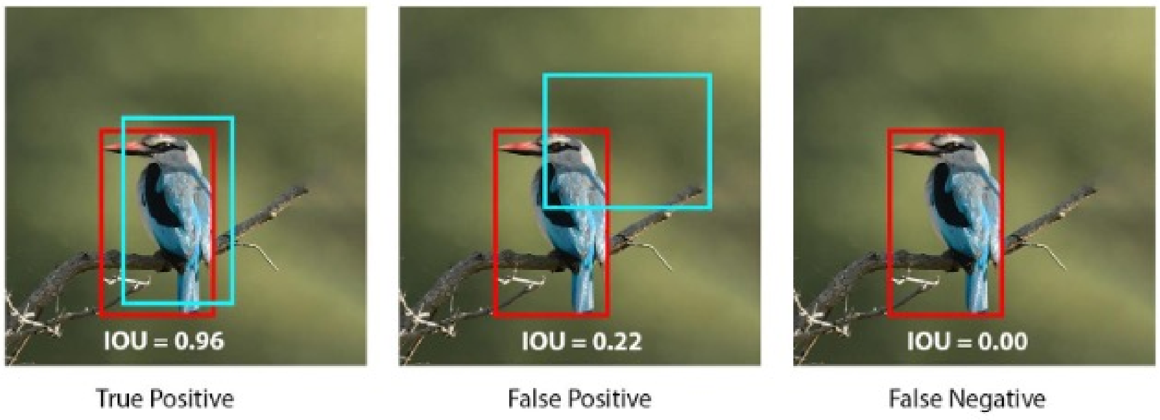


Figure 2.3: Example of different Intersection over Union (IoU) values in object detection [19]

Region of Interest (RoI) pooling

After generating region proposals, the aim of RoI pooling is to reduce all the feature maps obtained from each region proposal to the same size to enable classification and bounding box regression. This is achieved by flattening each of the feature maps, where each multi-dimensional feature map is converted into a 1-D vector array.

Classification

The final step of Faster R-CNN involves making use of the fixed-sized feature maps obtained from RoI pooling to perform object classification and predict refined bounding box coordinates for more accurate object localization. The feature maps are fed into a set of fully-connected layers to predict the probability of each object class being present in the region proposal. For this, the SoftMax function produces a probability distribution for each region. The class with the highest probability is considered the predicted class for a given region [20]. Additionally, the fully-connected layers also regress the coordinates of the bounding box, refining their positions and sized to obtain a more accurate localization of the object.

After classification and bounding box regression, the regions undergo non-maximum suppression to remove redundancy. This process selects the most confident detections and suppresses regions that have a significant overlap [21]. The final output of Faster R-CNN is a set of bounding boxes along with their corresponding class labels and confident scores, representing the detected objects in the input image.

2.2 Model evaluation and optimization

After building the model, it is important to establish an effective evaluation process to assess its performance and further optimize it.

2.2.1 Performance metrics

Loss function

In the context of AI, loss is a measure that quantifies the discrepancy between the predicted output of a model and the true output. It is used as a reference measure of how well a model is performing during training and can serve as a guide for model optimization. The choice of the loss function depends on the specific task. Examples of loss functions are MSE, Binary Cross-Entropy or Smooth L1 [22].

Binary Cross-Entropy is commonly used in binary classification problems, i.e. then the classification problem involves two classes. It quantifies the dissimilarity between the predicted probability distribution and the true binary labels (for example, legal and illegal) and is illustrated in Eq. (2.1) [23].

$$L_{\text{BCE}} = -(y_{\text{true}} \log(y_{\text{pred}}) + (1 - y_{\text{true}}) \log(1 - y_{\text{pred}})) \tag{2.1}$$

where y_{true} represents the true binary labels (0 or 1) and y_{pred} represents the predicted probabilities.

Another example is Smooth L1 loss, which is a function commonly used in regression tasks, such as predicting bounding box coordinates. It is a combination between the L1 and L2 loss and it adjusts the loss value based on the error difference [23]. The formula is as follows:

$$S_{\text{L1}} = \begin{cases} 0.5x^2, & \text{if } |x| < 1 \\ |x| - 0.5, & \text{otherwise} \end{cases} \tag{2.2}$$

Where x represents the difference between the predicted and ground truth value. Smooth L1 loss behaves like L1 loss or L2 loss, depending on the conditions.

If the error between the predictions and ground-truth is less than 1, then the function behaves like L2 loss. However, if the error is greater than 1, then it behaves like the L1 loss. One of the advantages is that it presents the advantages of both L1 and L2 loss function, and it is less sensitive to outliers than L2 loss. Outliers or extreme data points can occur due to various reasons such as measurement errors, noise, or anomalies. They have the potential to significantly affect the training process and model performance, as they can disproportionately influence the loss calculation. Therefore, being less sensitive to outliers is useful as it helps prevent the model from being overly influenced by anomalous or erroneous data points, leading to more reliable and robust performance.

In an object detection task, the total loss function would be a combination between the loss of classification and localization as illustrated in Eq. (2.3).

$$L = L_{cls} + L_{bbox} \tag{2.3}$$

The ultimate goal when training a model is to minimize the chosen loss function. This is typically achieved through optimization algorithms that adjust the model's parameters to minimize the discrepancy between the predicted and true values.

Precision and recall

Precision and recall can be used to evaluate an object detection model. They both range from 0 to 1. On one hand, precision, also known as positive predicted value, measures the percentage of the positive predictions that are truly positive as illustrated in Eq. (2.4). A higher precision indicates a lower rate of false positives, which means the model is more precise in identifying positive instances [24].

$$PR = \frac{tp}{tp + fp} \tag{2.4}$$

Where *tp* = true positives and *fp* = false positives.

On the other hand, recall, also known as sensitivity or true positive rate, measures the fraction of positives that are correctly identified as illustrated in Eq. (2.5). A higher recall indicates a lower rate of false negatives, which means the model is better at capturing positive instances [24]. In the case of illegal object detection, a high recall value would indicate that the model is effectively capturing most of the illegal objects in the dataset.

$$RE = \frac{tp}{tp + fn} \tag{2.5}$$

Where *fn* = false negatives.

The choice between recall and precision is dependent on the specific focus area. For instance, there might be a lower risk for the model to classify something legal as illegal than misclassifying something that is actually illegal. In such situations, opting for recall as a metric for evaluating the model would be more practical. However, it is not necessary to make an exclusive choice, as F1-score is a metric that incorporates both recall and precision [25].

F1 score

F1-score combines precision and recall into a single measure, providing a balanced evaluation of the model's overall effectiveness. F1-score is defined by the following equation:

$$F1 = 2x\frac{precisionxrecall}{precision + recall} \tag{2.6}$$

The value ranges from 0 to 1, where a higher value indicates better performance [25]. Moreover, F1 score is particularly useful when evaluating the model's performance on imbalanced datasets, where the number of positive and negative instances may vary significantly [26]. Taking this into account, the F1 score seems like a suitable estimate for defining the overall accuracy of a Faster R-CNN model.

2.2.2 Optimization techniques

Model optimization plays an essential role in training deep neural networks. Generally, there are two metrics used to evaluate the efficiency of optimizer: speed of convergence and generalization [27].

Stochastic gradient descent (SGD) is commonly used for training deep neural networks. It is an algorithm that starts at a random point on a function and travels down its slope in steps until it reaches the lowest point of that function. In SGD, instead of using the entire dataset for each iteration, only a small batch is randomly selected to calculate the gradient and update the model parameters. However, the randomness introduced by using batches of data can make the optimization process more noisy, leading to fluctuations in the loss function during training. To mitigate this, techniques like learning rate scheduling and momentum are often employed. The advantage of SGD is its computational efficiency, compared to traditional Gradient Descent or other methods that require processing the entire dataset [28].

Adam (Adaptive Moment Estimation) is another example of an optimization algorithm. It is an extended version of SGD that combines two SGD extensions-Root Mean Square Propagation (RMSProp) and Adaptive Gradient Algorithm (AdaGrad) to provide an adaptive learning rate and efficient parameter updates. One of its advantages is that it is widely used due to its fast convergence [29]. However, it has been shown that adaptive optimization methods such as Adam or RMSprop tend to perform well in the initial steps of training but tend to generalize poorly. Adam tends to converge faster, while SGD often converges to more optimal solutions [30]

In practice, the choice between SGD and Adam depends on various factors, including the specific optimization problem, the size of the dataset, the architecture of the model, and the available computational resources. It is often recommended to experiment with different optimizers and hyperparameters to find the best combination for a particular task. Additionally, techniques such as learning rate scheduling and early stopping can also be applied to further enhance the convergence and generalization properties of both SGD and Adam.

3. Neutron Image Dataset: Analysis and Insights

3.1 Description of the data

The dataset comprised a total of 2358 images, each captured from a distinct angle using the Dynaxion scanning software. Each image had a corresponding annotation file containing the following information about each object contained in the image: object name, material name, material legality, material category, object legality, bounding box coordinates (xmin, xmax, width and height). Figure 3.1 depicts an example neutron image with a bounding box surrounding each object (headphones, mobile phone, folded clothes, a book and a knife). Bounding boxes were needed to delineate the spatial extent of the object and provide a visual reference for object recognition.

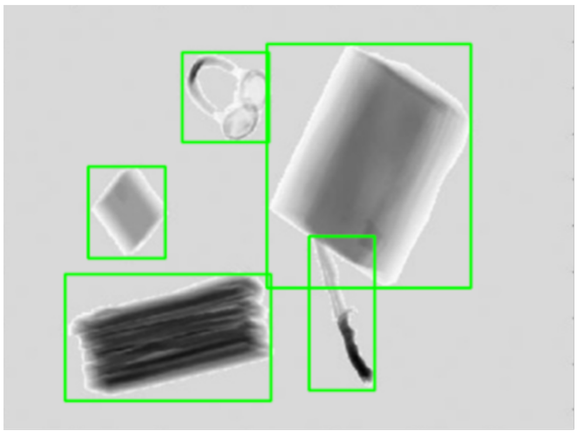


Figure 3.1: Example neutron image with bounding box surrounding object

The dataset encompassed 9162 objects present in the neutron images. Table 3.1 provides an overview of the object distribution within the dataset. Moreover, the different objects were distinguished based on their size and, hence, their area was measured as the number of pixels. There were 295 objects with an area smaller than 32^2 and were considered small objects (3.2 %); there were 56 with an area between 32^2 and 96^2 and were considered medium objects (0.6 %); and there were 8,811 (96.2 %) objects with an area greater than 96^2 and were considered to be large objects.

Table 3.1: The distribution of objects in the dataset

Object class	Count of Objects
MilitaryKnife*	1980
HandgunColt*	1530
FoldedClothes	1170
headphone	720
shoes	504
cola	468
sunglasses	468
digitalCamera	468
ChocolateBar	468
Huawei_P50_Pocket	450
Book	396
MaleShoes	270
Books	270

Moreover, objects marked with an asterisk (*) indicate target threat objects. There were 5652 legal objects (approx. 62%) and 3510 illegal objects (38%) across the entire dataset, as shown in Table 3.2. This means there is a slight class imbalance.

Table 3.2: Legality class balance

Legality class	Count
Legal	5652
Illegal	3510

The neutron images were obtained in different sizes and in nine different scales: 475 × 600, 475 × 775, 475 × 850, 475 × 825, 475 × 700, 475 × 700, 475 × 825, 475 × 850 and 475 x 775 pixels. This is because each image was taken from one of the different angles: 0° until 340°, in steps of 20°. The variation in sizes and scales of the neutron images is a direct result of the different angles at which they were taken. As the angle of view changes, the spatial coverage of the object within the image frame also changes.

3.2 Data preparation

Image resizing

The main objective of this phase of processing is to prepare an image that will provide the detection algorithm with the highest possible accuracy during training. To ensure consistency in the training and testing process, all images were uniformly scaled to a target size of 800 × 800 pixels. The choice of this specific size was arbitrary, with the primary goal being to achieve a square shape for the images, which would allow Faster R-CNN to further adjust their size based on the specific requirements of the feature extractor network being used. For instance, ResNet50 typically operates on images of size 224 × 224 pixels [31].

Furthermore, to facilitate accurate analysis and comparison of object positions and sizes within the images, the bounding box coordinates were normalized. This normalization process accounts for variations in image dimensions and ensures that the bounding box coordinates are relative to the target size of the image.

Training, validation and test split

Secondly, the dataset was randomly shuffled and split into a training set, validation set and test set to evaluate the model's performance. For this, the image and annotation file pairs were randomly divided into a training, a validation and a test set, maintaining a 7:1:2 ratio. Therefore, the training set comprised 1651 images, the validation set comprised 236 images and the test set contained 471 images.

Custom dataset

The subsequent step in the data processing involved preparing the training and validation data for the Faster R-CNN architecture. This was done using a custom dataset class, which would load the data in the format required for Faster R-CNN and retrieve the necessary information and map the object class names to their respective labels.

From the annotation files, the following information was extracted for each image: the coordinates of the N bounding boxes in [x0, y0, x1, y1] format contained in each image, the labels for each bounding box [1 or 2], a unique image identifier and the area of the bounding box. The label 0 represented the background class. Labelling was employed to annotate the ground truth within the dataset. Moreover, the images were normalized to ensure consistent input for the Faster R-CNN model. The image normalization involved dividing the pixel values by 255 to normalize it to the range [0,1].

In addition, and in accordance with the requirements of the Faster R-CNN model, the annotation files and images were stored in the same folder.

Data augmentation

Moreover, data transformation was applied to the training set to augment the set and improve the model's generalization. During training, the images were flipped horizontally. Moreover, motion blur effect, brightness, contrast and colour jitter were applied to the images. Each of the transformation techniques mentioned was applied to each image with a probability of 0.5. These transformations were applied during the training process, rather than generating additional augmented data beforehand. In this way, the model sees different versions of the same image during each training iteration.

4. Method

4.1 Faster R-CNN training

A PyTorch based framework was used to train and evaluate the Faster R-CNN model. All these implementations are available in open-source GitHub repository [32]. In order to get the most optimized final model, six models with different combinations of hyperparameter values were tested, allowing for a comparison of their performance and behaviour during training. Each of the model's architectures can be seen in Table 4.1.

Table 4.1: Overview of the six model trainings and configuration parameters

Model	**Backbone network**	**Optimizer**	**LR**	**Batch size**	**Number of epochs**	**Weight decay**
1	ResNet50	Adam	0.001	16	50	0.01
2	ResNet50	SGD	0.001	16	50	0.01
3	MobileNet-v3	Adam	0.001	16	50	0.01
4	MobileNet-v3	SGD	0.001	16	50	0.01
5	ResNet50	SGD	0.01	32	25	0.1
6	ResNet50	SGD	0.1	64	25	0.01

The different hyperparameters included the feature extractor network, optimizer, learning rate, batch size, number of epochs and weight decay. ResNet-50 and MobileNet-v3 were chosen as the backbone network for feature extraction. ResNet-50 is a deeper network that may capture more intricate features but requires higher computational resources, while MobileNet-v3 is optimized for computational efficiency while maintaining reasonable accuracy [33]. The choice between these two provided a trade-off between model complexity, computational resources, and performance.

Additionally, two different optimizers, namely Adam and Stochastic Gradient Descent (SGD), were selected for the training process. In some scenarios, Adam has demonstrated better optimization performance compared to SGD. However, recent research indicates that Adam may result in poorer generalization performance compared to SGD [29]. Given these findings and time constraints, it was of interest to select these two optimization methods. Moreover, momentum of 0.9 was applied to the models trained with SGD optimizer. This was done because SGD with momentum is a method which helps accelerate gradients vectors in the right directions, as the current update is highly influenced by the previous update, thus leading to faster converging and escaping the possible local minima. In addition, if momentum is applied, local minima be escaped and model can reach to the global minima. Literature has shown that a momentum value of 0.9 is commonly considered a good choice [34].

As a side note, the training could not be performed using a GPU-accelerated environment to improve computational efficiency for the training.

4.2 Model validation

The trained Faster R-CNN models were evaluated on the validation dataset to assess its performance and generalization ability to detect and localize objects accurately in neutron images. For this, the model's total loss was calculated by summing the four individual losses from the two main components: the RPN and the classifier. Each individual loss would contribute equally to the total loss. The loss functions used for each component can be seen in Table 4.2. The classification loss uses Binary cross entropy loss function and the regression loss uses the Smooth L1 Loss function.

Table 4.2: Loss functions used for model validation

Component	Task	Loss function
RPN	Foreground or background class prediction	Binary cross-entropy
	Candidate object region proposal	Smooth L1
Classifier	Class label prediction	Binary cross-entropy
	Bounding box prediction	Smooth L1

The training loss serves to optimize and evaluate the model's performance during the training process. However, it is important to note that the training loss alone may not offer a comprehensive understanding of the model's generalization capability to unseen data. Therefore, the evaluation of each model also involved calculating the mean average precision (mAP) and the mean average recall (mAR). They were both calculated and averaged over 10 different IoU threshold values from 0.5 to 0.95 with a step size of 0.05. This is because having a single threshold to assess the detection model might not completely reflect the localization performance as it could induce bias in the evaluation metric.

These metrics were obtained as an overall average across all objects and separately for three distinct object sizes: small (area smaller than 32^2), medium (area between 32^2 and 96^2) and large (area larger than 96^2). In the end, the F1-score was derived from the mAP and mAR (using Eq. 2.6) as the final metric for assessing the overall performance of the models, given that it combines both precision and recall into a single measure.

4.3 Model evaluation

Finally, after performing hyperparameter tuning, the final and improved model was evaluated using the unseen test dataset. For this, inference was performed on the test dataset with a detection IoU threshold of 0.7. Model inference involved loading the final model with its weights to predict each object class together with the corresponding bounding box for each image in the test dataset. All the images with the predictions and their object label confidence scores were saved. Finally, a confusion matrix using the correct and predicted labels was generated.

5. Results

The primary emphasis of the results is on giving an overview of the obtained results, including loss values, F1 scores and overall model performance. The first subsection focuses on evaluation metrics on the validation dataset and the second subsection focuses on the inference results on the test dataset, specifically using the model that obtained the best overall results from the validation phase for predicting on new images.

5.1 Evaluation metrics

Validation loss

Table 5.1 presents the total loss and the individual losses for each model after the completion of training. Models 1, 2, 3 and 4 completed 50 epochs of training, whereas models 5 and 6 completed 25 epochs. This was due to limited time constraints. The sum of all four individual losses result in the total loss, as seen in Eq. (2.3). According to the table, model 2 demonstrates the lowest loss among all models for the final classification (0.022) and bounding box regression tasks (0.031) in the Faster R-CNN. Although model 1 yields the lowest loss for the RPN classification (0.012) and RPN bounding box regressor task (0.005), model 2 still achieves a lower total loss (0.079).

Table 5.1: Average loss values and loss values for each component of Faster R-CNN the end of the training process (epoch 50 for models 1, 2, 3 and 4; and epoch 25 for models 5 and 6)

Model	Total loss	Loss classifier	Loss BB regressor	Loss RPN classifier	Loss RPN bbox regressor
Model 1	0.302	0.113	0.172	**0.012**	**0.005**
Model 2	**0.079**	**0.022**	**0.031**	0.016	0.010
Model 3	0.299	0.116	0.161	0.016	0.006
Model 4	0.138	0.041	0.062	0.023	0.012
Model 5	0.606	0.181	0.133	0.224	0.068
Model 6	0.326	0.097	0.146	0.052	0.031

**Values in bold correspond to the lowest loss for each component.*

In addition, the loss values for each model were recorded throughout all epochs (*Appendix A*). The loss graphs do not show any clear difference in pattern across the models. The loss trajectories for all models appear to follow a similar trend as they all decrease exponentially and eventually reach stability.

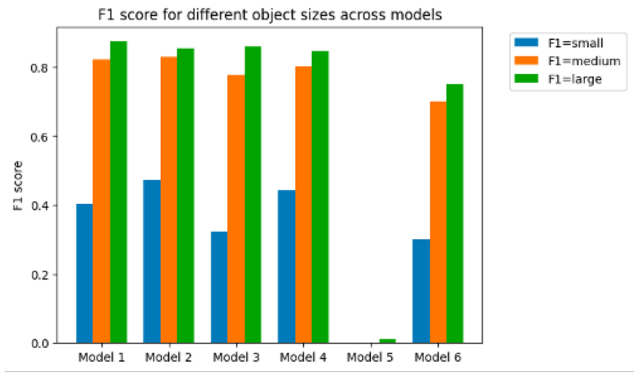
F1 score

From the mAP and mAR, the F1 score was calculated using Eq. (2.6) throughout the training processes. Table 5.2 shows the F1 scores across various object sizes (small, medium and large) and the overall F1 score for each of the models at the last epoch of the training. In addition, Figure 5.1 (a) is a visual representation of the F1 score at the end of the training for the different object sizes and Figure 5.1 (b) is a visual representation of the overall F1 score throughout the training processes.

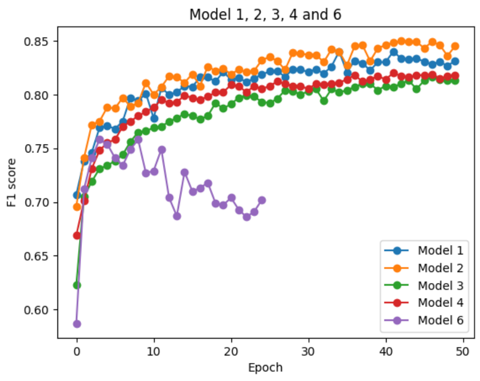
Table 5.2: F1 scores for small, medium and large objects as well as the overall F1 score at the end of the training process (epoch 50 for models 1, 2, 3 and 4; and epoch 25 for models 5 and 6)

Model	F1-score small	F1-score medium	F1-score large	Average F1-score
Model 1	0.40	0.82	0.87	0.83
Model 2	0.47	**0.85**	**0.89**	**0.85**
Model 3	0.37	0.81	0.88	0.81
Model 4	**0.49**	0.82	0.86	0.82
Model 5	0.00	0.00	0.01	0.01
Model 6	0.30	0.70	0.75	0.70

**Values in bold correspond to the largest F1 score for each component.*



(a)



(b)

Figure 5.1: F1 scores for (a) small (blue), medium (orange) and large (green) objects; and (b) model 1 (blue), model 2 (orange), model 3 (green), model 4 (red) and model 6 (purple) across the training history.

From the table and from Figure 5.1, it can be observed that models 1, 2, 3 and 4 yield similar results, as the difference in performance is not very big. Specifically, model 4 achieves the highest F1-score (0.49) for the detection of small objects. On the other hand, model 2 has the largest F1 score for medium (0.85) and large objects (0.89) as well as the largest F1 average (0.85). Besides these, models 5 and 6 exhibit worse performance results. Model 5 shows F1 scores of 0 or approximately 0 for all categories and model 6 has the second-lowest average F1-score among all models. Additionally, the F1 score in model 6 does not increase like models 1,2 3 and 4, but it decreases. Model 6 is not included in the F1 graphs, as the obtained values were very close to 0. Moreover, as seen in fig. 5.1 (a) and in (*Appendix B*), the F1 score is highest for large objects and lowest for small objects across the models.

5.2 Inference on test dataset

Confusion matrix

Table 5.2 represents the confusion matrix obtained when performing model inference on the test dataset using model 2. It shows that there are 282 instances of the 'Legal' class that were correctly predicted as 'Legal' (true negatives). Moreover, 210 instances of the 'Illegal' class were correctly predicted as 'Illegal' (true positives). Therefore, the model did not misclassify any object in the entire test dataset.

Prediction sample images

Table 5.3: Confusion matrix on test dataset (492 images)

True values / Predicted values	**Positive (Illegal)**	**Negative (Legal)**
Positive (Illegal)	210	0
Negative (Legal)	0	282

The complete set of 492 prediction images after choosing model 2 for predictions can be found in the open-source GitHub repository [32]. Figure 5.2 shows 6 out of them used to analyse and visually assess the model's performance. A confidence score threshold of 0.7 or higher was selected to ensure that the predictions excluded classifications where the confidence score was lower. As seen in Figure 5.2, the model's predictions were precise in identifying the objects and their respective bounding boxes, even in cases where the objects were partially hidden and had a different assigned legality label (Figure 5.2 (a), (b) and (f)). Moreover, figure 5.2 (a) includes a military knife positioned at the lower left corner that is barely visible to the human eye. However, the model successfully identified and accurately predicted the presence of the knife.

The prediction error mainly resulted from bounding box regression, rather than classification. This can be seen in Figure 5.2 (c), where the model interpreted the two military knives as being the same object and, hence, drawing one bounding box around them. However, this was not always the case. For instance, in Figure 5.2 (d), the model successfully depicted the objects as two knives.

Additionally, Figure 5.2 (e) illustrates an example of an image where the model successfully detected a very small object. In this case, the object corresponds to a pair of men's shoes. Initially, there was a presumption of error, but upon examining the corresponding annotation file, the prediction made by the model was indeed accurate. Finally, Figure 5.2 (f) is another example of correctly classified object.

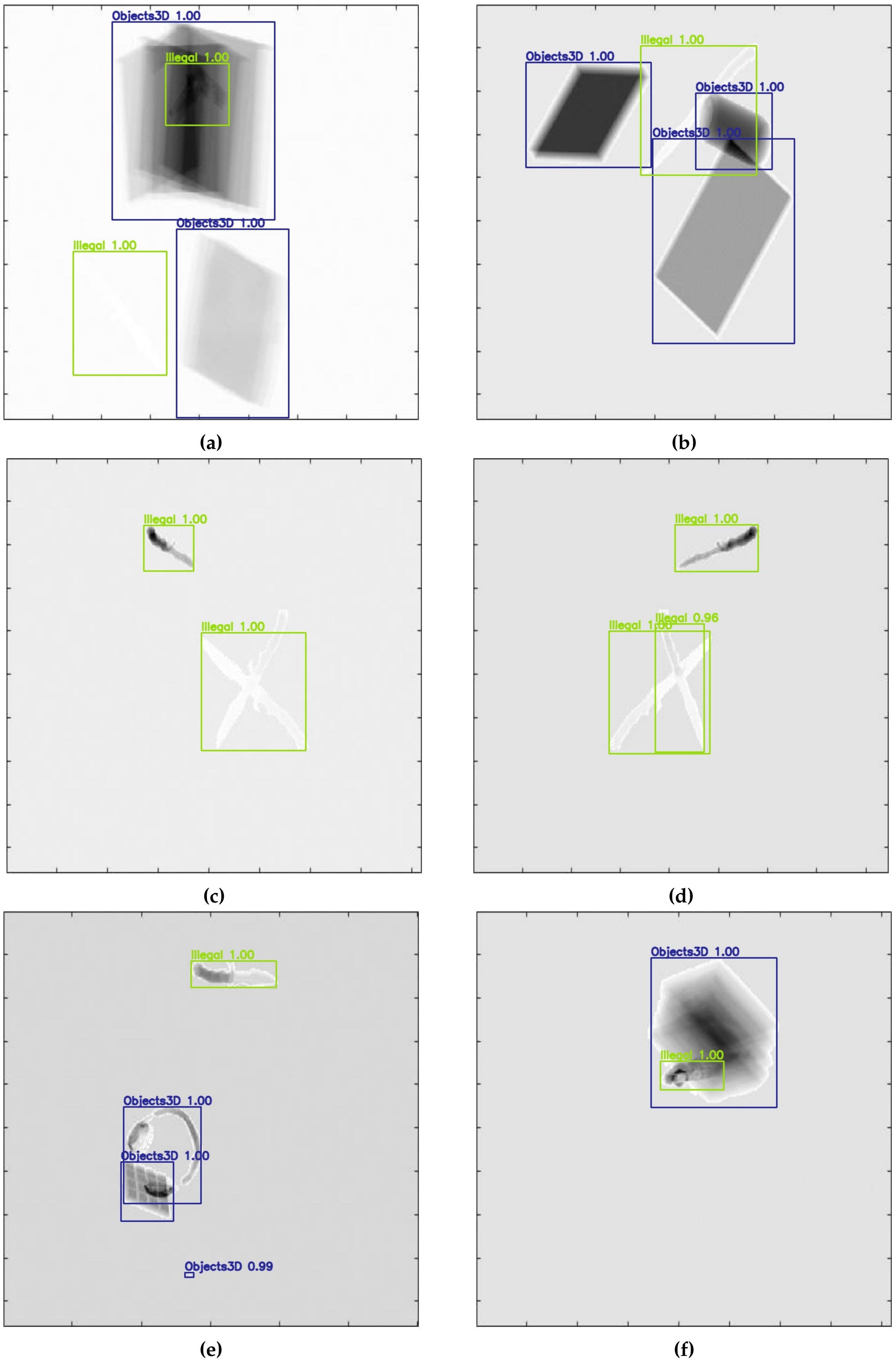


Figure 5.2: Prediction images on the test dataset with the corresponding bounding box, class label and confidence score for object class

6. Discussion

The aim of this study was to compare six different Faster R-CNN models in order to find the network architecture that would lead to the more optimal results for the task of object detection in neutron images. Therefore, the primary emphasis of this section is to compare the different models, as the objective was to optimize a model using different network configurations, and analyse potential sources of error.

6.1 Evaluation metrics

Validation loss

The validation loss graphs (*Appendix A*) show that the models exhibit similar behaviour as the loss decreased exponentially. Moreover, it also demonstrates that the models reached a plateau in their learning and that further improvements in loss reduction may require more sophisticated techniques or adjustments to the training approach. This stability implies that the models consistently maintain their performance throughout the training process without significant fluctuations or abrupt changes in loss values. Model 2 produced the minimum total loss of 0.079, as compared to 0.606 which corresponds to the highest total loss of model 5. This indicates a significant improvement in performance for Model 2. As compared to the other models, model 5 yielded acceptable loss values for the RPN component of the Faster R-CNN. However, it exhibited significantly higher loss values for the final classification and regression tasks. Hence, model 5 serves as an illustration that even with a relatively low validation loss, as seen in Table 5.2, that the model's performance can still be unsatisfactory.

F1 score

Since a low loss value during training does not necessarily guarantee a high F1 score or good performance on the evaluation or test data, it is also important to take F1 into consideration. From the F1 scores in Table 5.2, it can be concluded that the average F1 score between model 1, 2, 3 and 4 was similar; the lowest value being 0.81 from model 3 and the highest being 0.85 from model 2. Model 6 yielded an average F1 score of 0.70, indicating that it performs less effectively compared to the other models in predicting object boundaries accurately. Model 5 performed the worst with an average F1 score of 0, which suggests that this model has very low accuracy and performs poorly in object detection of neutron images.

Moreover, as can be seen in Figure 5.1 (a), the F1 score is highest for large objects (best value of 0.89) and medium objects (best value of 0.85) but lower for small objects (best value of 0.49). This suggests that the models are generally more effective in accurately detecting and classifying larger objects compared to smaller ones. One reason for this is that there are more large objects (96.2%) than small objects (3.2%) in the complete dataset. This imbalance in the dataset with a significantly higher proportion of large objects compared to small objects, can impact the model's performance. The model may become biased towards the majority class (large objects) and perform less effectively on the minority class (small objects). Another reason could be that the features used by the feature extractor network may be more suited to capturing the visual patterns of large objects. If the algorithm relies on specific features that are more prominent in larger objects, it may struggle to effectively detect smaller objects that exhibit different or less prominent visual patterns. In terms of neutron transmission imaging, small objects tend to transmit a smaller fraction of the incident neutron flux compared to larger objects. This is because the interaction between neutrons and small objects is less pronounced due to their smaller size and lower density, which can make the detection of small objects more challenging in neutron transmission imaging.

Regarding the F1 score progress throughout the training, as shown in Figure 5.1 (b), models 1, 2, 3 and 4 resulted in increased F1 score, whereas the F1 score for model 6 showed decreasing behavior. This might be because of the high learning rate assigned to the training of this model. This might have caused the model's parameter updates to be too large and therefore not converging to an optimal solution.

From the loss values and the F1 scores obtained, it can be concluded that model 2 demonstrated superior overall performance, by achieving minimal total loss and maximal average F1 score and F1 score for medium and large objects. Despite model 4 achieving the highest F1 score of 0.49, model 2 still obtained a score of 0.47, which is relatively close in value.

6.2 Model hyperparameters

Model 1, 2, 3 and 4 shared nearly identical configurations, including a learning rate of 0.001, a batch size of 16, 50 epochs, and a weight decay of 0.01. The only difference between them was the feature extractor network and the optimizer used to adjust the network weights. Model 1 and 2 both employed ResNet50, whereas model 3 and 4 employed MobileNet-v3. Moreover, model 1 and 3 had Adam as optimizer; and model 2 and 4 had SGD as optimizer. This was done so that they could be compared to determine which of the two feature extractors and optimizers would be a more suitable choice for the specific configuration being studied.

From the validation loss (Table 5.1) and the F1 scores (Table 5.2), it can be concluded that SGD yielded lower loss values and larger F1 scores than Adam. Unfortunately, speed of convergence was not recorded. Hence, no conclusions can be drawn about whether Adam or SGD was faster. A study conducted by Mahmoud et al. compared different optimizers for object detection in remote sensing images. It was concluded that Adam yielded higher accuracy than SGD [35]. However, it was also the case that no momentum was applied to SGD which could have lead to a possible local minima in the loss function. However, the fact that SGD performed better than Adam in this study is in line with other recent studies that show that Adam often leads to worse generalization performance than SGD [29].

Moreover, as to the feature extractor network, ResNet50 yielded better results than MobileNet-v3 when comparing the total loss values (Table 5.1) and average F1 scores (Table 5.2). This could be because ResNet50 is a deeper network with a larger number of layers than MobileNet-v3, and might therefore perform better for large datasets and complex architectures [36]. This aligns with a study conducted by Storey et al. where they compared several R-CNN models to detect possible signs of disease in images containing plant leaves. ResNet50 yielded an accuracy of 80.5% and MobileNet-v3 an accuracy of 68.3%.

Having discussed this, model 5 and model 6 were adjusted based on the better performance of ResNet50 backbone network and SGD optimizer. For these models, the other hyperparameters were further adjusted (learning rate, batch size, number of epochs and weight decay). From Table 5.1, it can be seen that the loss values obtained with these two models were larger, as compared to the other models. Furthermore, Table 5.2 reveals that model 5 and model 6 resulted in lower F1 scores, with model 5 demonstrating significantly lower F1 scores than the other models, as all values were approximately 0. The poor performance of model 5 is suspected to be due to the large weight decay. It has been shown that, in the case of excessive weight decay, even with prolonged training, the model can fail to achieve a satisfactory level of fitting [37].

Furthermore, there exists an association between the learning rate and batch size, as evidenced by previous findings [38]. In cases where the learning rates are set to high values, employing a larger batch size can yield to superior performance compared to using smaller learning rates. Nevertheless, it appears that opting for a large batch size and high learning rate for model 6 did not yield significant differences in outcomes. The model's performance, while performing better than model 5, exhibited a lower level of effectiveness compared to the remaining models. The study also showed that lowering the learning rate and decreasing the batch size will allow the network to train better [38]. Taking into account that the low learning rate and small batch size of model 1, 2, 3 and 4 yielded the best results, it is recommended for future research to use these models as a baseline or initial starting point.

Additionally, model 5 and 6 were trained only for 25 epochs due to limited time constraints. However, it can be seen from the F1 score in Figure 5.1 (b) and the loss behaviour throughout the training (*Appendix A*) that, even with an extended training duration of 50 epochs, these models would not be expected to outperform the other models. Considering that the primary objective of this study is to identify the most optimal model, these two models were excluded from the pool of potential candidates during the selection process.

Overall, the results highlight the desirable performance of models 1, 2, 3 and 4 achieving good results, particularly for large objects. Model 2 yielded the best performance by making use of ResNet50, SGD, a low learning rate and weight decay, a small batch size, and a relatively large number of epochs. Meanwhile, models 5 and 6 exhibit comparatively worse performance, potentially indicating areas for further improvement.

6.3 Inference on test dataset

As shown in Figure 5.2, the model generally performed accurately, correctly classifying and enclosing each object with its respective bounding box, even when objects were occluded by other objects. Moreover, the confusion matrix on the test dataset showed that the legality labels from all bounding boxes were correctly classified. Even when model's small object F1 score was relatively low (0.49), the model generally correctly classified very small objects when visually inspecting the images.

In some cases, the model predicted two objects as being one and therefore, drawing one single bounding box around them. This happened because in the last step of a Faster R-CNN, the regions undergo maximum-suppression by which multiple bounding boxes with a large IoU get suppressed to remove redundancy. By lowering the threshold, the model might be more tolerant of overlapping bounding boxes and less likely to merge separate objects into a single box during the maximum-suppression step. Another alternative would be making use of Soft-NMS, which, instead of completely removing bounding boxes that overlap significantly, the suppression process gradually reduces their scores or weights [39]. In this way, it would reduce the chances of merging them into a single bounding box. In any case, it would not be a big problem as this was only the case when two objects with the same legality label overlapped. In a real-world scenario, the focus is primarily on identifying illegal objects. If two illegal objects located very closely were mistakenly treated as a single object, it would not be a major problem, as further inspection of the parcel or suitcase would be required regardless.

Moreover, it would have been useful to calculate mAP and mAR on the test data in addition to the mAP mAR values on the validation data. However, the obtained values can be considered to be a good approximation to how the model performs as the validation dataset is not used for training and is therefore new unseen data for the model at the evaluation step.

After establishing a specific model and evaluating its performance and generalizability, it is worth thinking about how the model would perform in a real-world scenario. First of all, the model's performance would probably not be the same as it has only been trained on a limited dataset comprising 12 distinct object classes. In reality, there can be a significantly larger variety of object classes. As an alternative, a hypothetical test case was explored. Due to the lack of concrete data on the exact number of illegal items shipped in parcels, it is difficult to provide precise figures. However, the test case assumed a conservative estimate of 1% illicit materials being shipped. Although this number is purely speculative, it helps to highlight the potential impact and risks associated with illegal shipments when using the model derived within this study. Model 2 results in an mAP of 0.83 and mAR of 0.86. Assuming a large number of objects, let's say there are 10,000 objects in total. With 1% of them being illegal, there are 100 illegal objects. Recall indicates that the model can correctly identify approximately 86% of the illegal objects. Therefore, the model would be expected to detect around 86 out of the 100 illegal objects. On the other hand, precision suggests that out of the objects classified as illegal by the model, around 83% are true positives. So out of the objects flagged as illegal by the model, approximately 83% or 83 objects would be truly illegal. Although the model's performance metrics indicate favorable results, they should not be regarded as a complete substitute for manual detection of illegal objects, especially if the risks taken when missing an illegal object are high.

6.4 Limitations and future research

The aim of this study was to optimize a Faster R-CNN model capable of effectively detecting objects in neutron images. This objective was successfully accomplished; however, the study also identified certain limitations and proposed areas for further improvement. First, the use of macOS as operating system posed a limitation as it was not completely compatible with Python PyTorch, resulting in the usage of CPU instead of GPU for model training. Consequently, this significantly slowed down the training process, with each training taking approximately 20-40 hours. Therefore, to overcome the slow training process, it is recommended to explore compatibility with GPU platforms, such as by transitioning to a different operating system or finding alternative solutions to enable GPU utilization. This could significantly reduce the training time and facilitate the comparison of additional network configurations.

Moreover, the accuracy of the model may be influenced by the fact that it was tested on only 13 different types of objects (Table 3.1). In the real world, there exists a vast variety of object classes, which were not included in the evaluation. This limitation suggests the need for future work to enhance the model's applicability in real-world scenarios by testing the method with a more extensive and diverse range of threat objects.

As another limitation, a class imbalance between object sizes was observed, as the dataset contained 3.2% small objects and 96.2% large objects. This is clearly a notable difference, which might have been the reason for such low F1 score values for small objects. Moreover, a slight class imbalance was observed in the dataset, with the legal class constituting 62% of the samples, while the illegal class accounted for 38% (Table 3.2). This imbalance in class distribution could have also influenced the model's performance and, therefore, future work should consider employing weighted class balancing techniques. By assigning higher weights to the underrepresented classes during training, the model can learn to better handle imbalanced data and improve its performance on the minority classes.

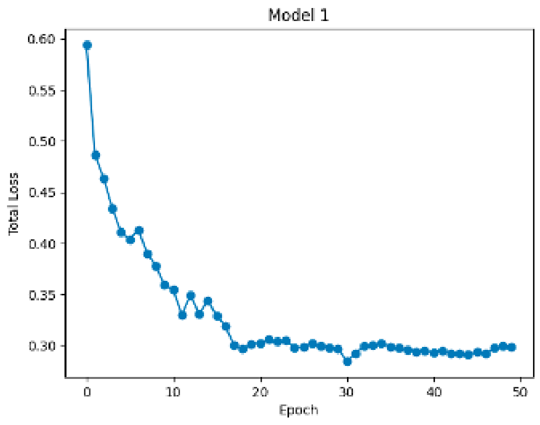
Finally, it is also important to explore advanced techniques, such as different types of data augmentation and perform additional attempts to fine-tune the hyperparameters, such as learning rate, batch size, and weight decay. This altogether could lead to improved model performance.

6.5 Conclusion

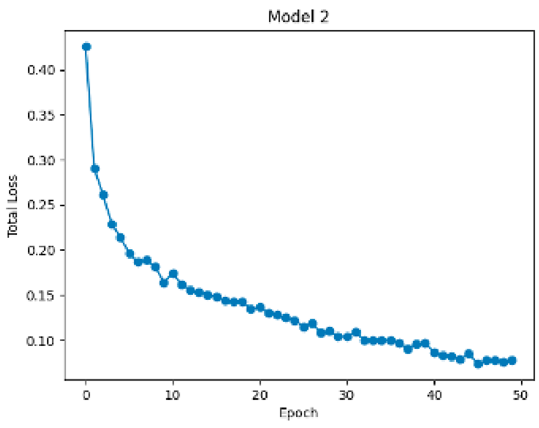
The transportation of suitcases and parcels poses security risks due to uncertainties regarding their contents. Therefore, imaging techniques can be useful to detect suspicious content in parcels or suitcases. As compared to traditional X-Rays, neutron imaging offers advantages as they penetrate dense and shielded materials more effectively and they are more sensitive to light elements, such as hydrogen or nitrogen, which are often contained in illegal substances. Moreover, as these images need to be interpreted and manual detection is costly and leaves window to human error, the use of AI algorithms has become of great interest to enhance security screening. Object recognition, particularly using Faster R-CNN, has shown promising results in previous studies involving complex scenarios with multiple objects. Therefore, it was of great interest to make use of the neutron images generated at Dynaxion's scanning system.

This study aims to train and optimize a Faster R-CNN model for accurate object detection in neutron images. For this, six models with different combinations of hyper-parameters were tested and compared. Their network architecture included different feature extractor networks, optimizers, learning rates, batch sizes, number of epochs, and weight decay values. The models were trained using ResNet-50 and MobileNet-v3 as backbone networks for feature extraction; and Adam and Stochastic Gradient Descent (SGD) were chosen as optimizers. Model validation was performed on a separate validation dataset, evaluating the model's total loss and calculating mean average precision (mAP) and mean average recall (mAR) at various IoU thresholds. The F1-score was derived from mAP and mAR as the final metric for overall performance assessment. The final model was evaluated on an unseen test dataset using a detection IoU threshold of 0.7. The choice of feature extractor network and optimizer affected the model's performance, with ResNet50 and SGD yielding better results. From the six models, the Faster R-CNN with ResNet50 and SGD demonstrated the lowest validation loss (0.079) and achieved the highest F1 scores for medium (0.85) and large objects (0.89) as well as the highest average F1 score (0.85). Sample images are provided to visually assess the model's performance, where model generally detected objects accurately, even in cases of occlusion or partial visibility. While the model showed satisfactory performance, it is recommended for future research to further expand the complexity of the dataset and model architecture to an even more realistic scenario.

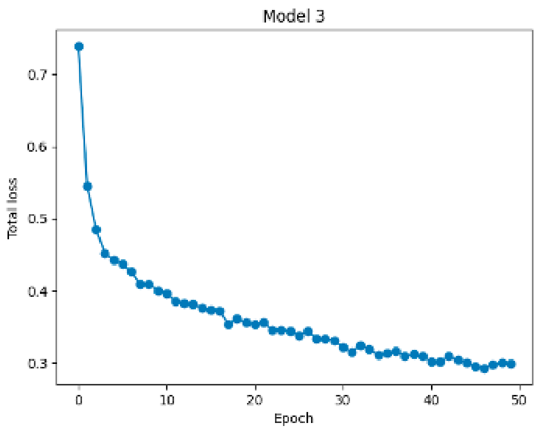
Appendix A



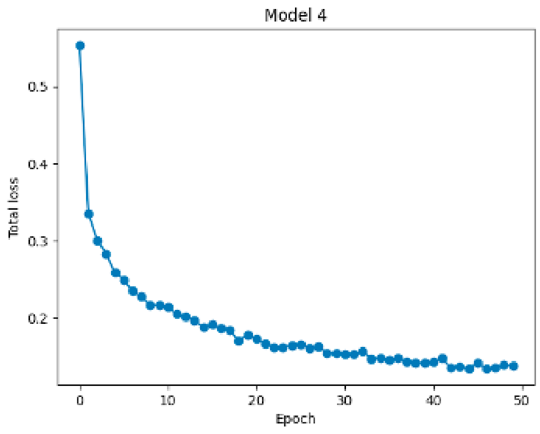
(a)



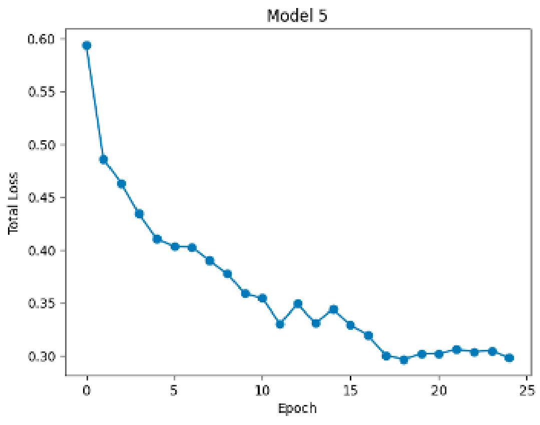
(b)



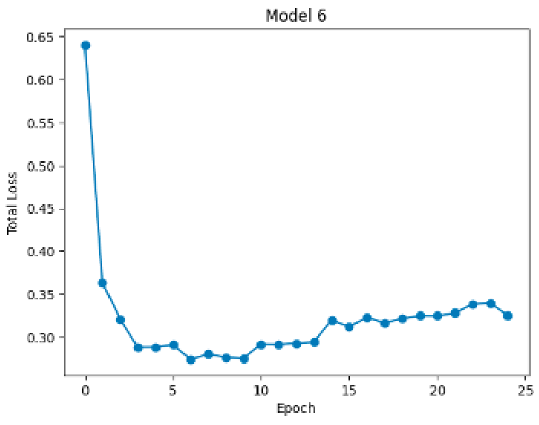
(c)



(d)



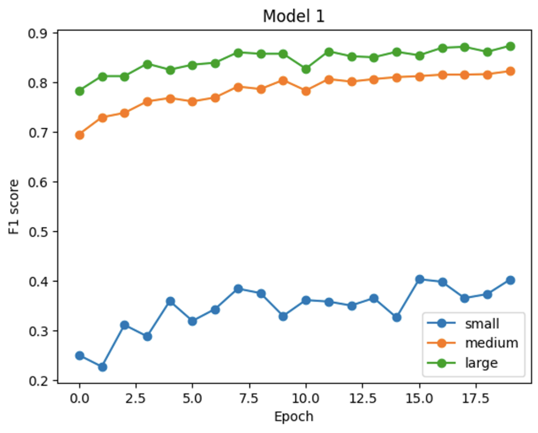
(e)



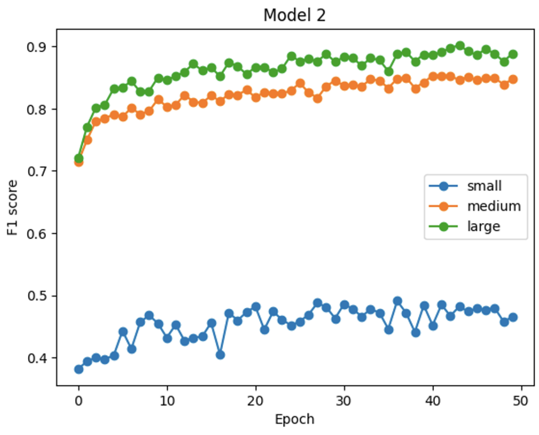
(f)

Figure 6.1: Validation loss graph for each model throughout the duration of the training: 50 epochs (model 1, 2, 3 and 4) and 25 epochs (model 5 and 6)

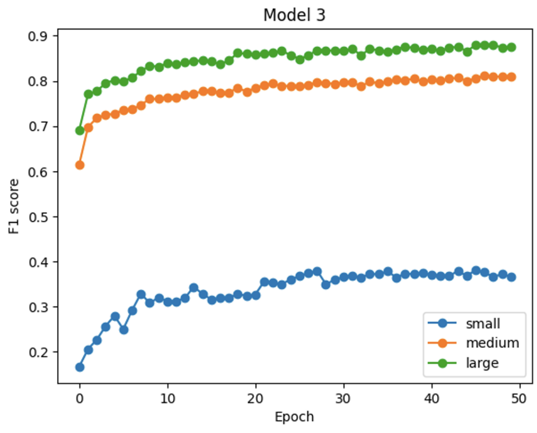
Appendix B



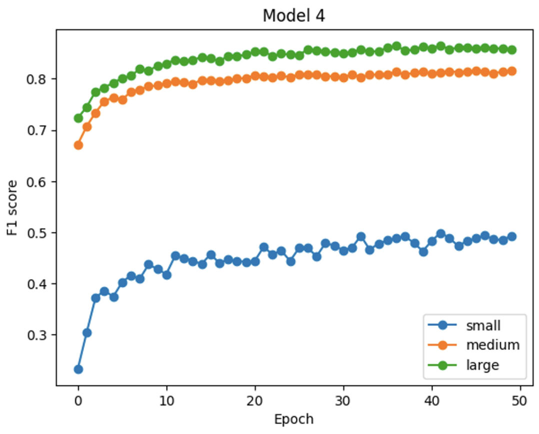
(a)



(b)



(c)



(d)

Figure 6.2: F1 score for each object size (small, medium, large) for each model throughout the duration of the training: 50 epochs (model 1, 2, 3 and 4)

Bibliography

[1] *Airline Passenger Security Screening*. National Academies Press, Jun. 1996. DOI: 10.17226/5116. [Online]. Available: https://doi.org/10.17226/5116.

[2] X. Ou, X. Chen, X. Xu, *et al.*, "Recent development in x-ray imaging technology: Future and challenges," *Research*, vol. 2021, Jan. 2021. DOI: 10.34133/2021/9892152. [Online]. Available: https://doi.org/10.34133/2021/9892152.

[3] S. Chen, M. Bourham, and A. Rabiei, "Attenuation efficiency of x-ray and comparison to gamma ray and neutrons in composite metal foams," *Radiation Physics and Chemistry*, vol. 117, pp. 12–22, Dec. 2015. DOI: 10.1016/j.radphyschem.2015.07.003. [Online]. Available: https://doi.org/10.1016/j.radphyschem.2015.07.003.

[4] A. Buffler, "Contraband detection with fast neutrons," *Radiation Physics and Chemistry*, vol. 71, no. 3-4, pp. 853–861, Oct. 2004. DOI: 10.1016/j.radphyschem.2004.04.110. [Online]. Available: https://doi.org/10.1016/j.radphyschem.2004.04.110.

[5] N. Kardjilov, I. Manke, R. Woracek, A. Hilger, and J. Banhart, "Advances in neutron imaging," *Materials Today*, vol. 21, no. 6, pp. 652–672, Jul. 2018. DOI: 10.1016/j.mattod.2018.03.001. [Online]. Available: https://doi.org/10.1016/j.mattod.2018.03.001.

[6] Y. Huang, X. Fu, and Y. Zeng, "Anchor-free weapon detection for x-ray baggage security images," *IEEE Access*, vol. 10, pp. 97 843–97 855, 2022. DOI: 10.1109/ACCESS.2022.3205593.

[7] F. Nachreiner, P. Nickel, and I. Meyer, "Human factors in process control systems: The design of human–machine interfaces," *Safety Science*, vol. 44, no. 1, pp. 5–26, Jan. 2006. DOI: 10.1016/j.ssci.2005.09.003. [Online]. Available: https://doi.org/10.1016/j.ssci.2005.09.003.

[8] J. Kaur and W. Singh, "Tools, techniques, datasets and application areas for object detection in an image: A review," *Multimedia Tools and Applications*, vol. 81, no. 27, pp. 38 297–38 351, Apr. 2022. DOI: 10.1007/s11042-022-13153-y. [Online]. Available: https://doi.org/10.1007/s11042-022-13153-y.

[9] D. Mery, V. Riffo, I. Zuccar, and C. Pieringer, "Automated x-ray object recognition using an efficient search algorithm in multiple views," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, Jun. 2013.

[10] Z. Yahya, S. Imane, H. Hicham, A. Ghassane, and E. B.-I. Safia, "Applied imagery pattern recognition for photovoltaic modules' inspection: A review on methods, challenges and future development," *Sustainable Energy Technologies and Assessments*, vol. 52, p. 102 071, Aug. 2022. DOI: 10.1016/j.seta.2022.102071. [Online]. Available: https://doi.org/10.1016/j.seta.2022.102071.

[11] Z.-Q. Zhao, P. Zheng, S.-T. Xu, and X. Wu, "Object detection with deep learning: A review," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 30, no. 11, pp. 3212–3232, 2019. DOI: 10.1109/TNNLS.2018.2876865.

[12] L. Zhu, Z. Xie, L. Liu, B. Tao, and W. Tao, "IoU-uniform r-CNN: Breaking through the limitations of RPN," *Pattern Recognition*, vol. 112, p. 107 816, Apr. 2021. DOI: 10.1016/j.patcog.2021.107816. [Online]. Available: https://doi.org/10.1016/j.patcog.2021.107816.

[13] R. Sa, W. Owens, R. Wiegand, *et al.*, "Intervertebral disc detection in x-ray images using faster r-CNN," in *2017 39th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC)*, IEEE, Jul. 2017. DOI: 10.1109/embc.2017.8036887. [Online]. Available: https://doi.org/10.1109/embc.2017.8036887.

[14] J. Koci, A. O. Topal, and M. Ali, "Threat object detection in x-ray images using SSD, r-FCN and faster r-CNN," in *2020 International Conference on Computing, Networking, Telecommunications &amp Engineering Sciences Applications (CoNTESA)*, IEEE, Dec. 2020. DOI: 10.1109/contesa50436.2020.9302863. [Online]. Available: https://doi.org/10.1109/contesa50436.2020.9302863.

[15] S. Ren, K. He, R. Girshick, and J. Sun, *Faster r-cnn: Towards real-time object detection with region proposal networks*, 2016. arXiv: 1506.01497 [cs.CV].

[16] A. B. Amjoud and M. Amrouch, "Convolutional neural networks backbones for object detection," in *Lecture Notes in Computer Science*, Springer International Publishing, 2020, pp. 282–289. DOI: 10.1007/978-3-030-51935-3_30. [Online]. Available: https://doi.org/10.1007/978-3-030-51935-3_30.

[17] T. Wen, H. Wu, Y. Du, and C. Huang, "Faster r-cnn with improved anchor box for cell recognition," *Mathematical Biosciences and Engineering*, vol. 17, no. 6, pp. 7772–7786, 2020. DOI: 10.3934/mbe.2020395. [Online]. Available: https://doi.org/10.3934/mbe.2020395.

[18] M. M. Rahman, Y. Tan, J. Xue, L. Shao, and K. Lu, "3d object detection: Learning 3d bounding boxes from scaled down 2d bounding boxes in RGB-d images," *Information Sciences*, vol. 476, pp. 147–158, Feb. 2019. DOI: 10.1016/j.ins.2018.09.040. [Online]. Available: https://doi.org/10.1016/j.ins.2018.09.040.

[19] Kukil. "Intersection over union (iou) in object detection segmentation." (2022), [Online]. Available: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/.

[20] K. Faust, Q. Xie, D. Han, *et al.*, "Visualizing histopathologic deep learning classification and anomaly detection using nonlinear feature space dimensionality reduction," *BMC Bioinformatics*, vol. 19, no. 1, May 2018. DOI: 10.1186/s12859-018-2184-4. [Online]. Available: https://doi.org/10.1186/s12859-018-2184-4.

[21] A. H. Al-Badri, N. A. Ismail, K. Al-Dulaimi, G. A. Salman, and M. S. H. Salam, "Adaptive non-maximum suppression for improving performance of rumex detection," *Expert Systems with Applications*, vol. 219, p. 119 634, Jun. 2023. DOI: 10.1016/j.eswa.2023.119634. [Online]. Available: https://doi.org/10.1016/j.eswa.2023.119634.

[22] B. Wilson, J. Hoffman, and J. Morgenstern, *Predictive inequity in object detection*, 2019. arXiv: 1902.11097 [cs.CV].

[23] C.-j. Li, Z. Qu, S.-y. Wang, and L. Liu, "A method of cross-layer fusion multi-object detection and recognition based on improved faster r-CNN model in complex traffic environment," *Pattern Recognition Letters*, vol. 145, pp. 127–134, May 2021. DOI: 10.1016/j.patrec.2021.02.003. [Online]. Available: https://doi.org/10.1016/j.patrec.2021.02.003.

[24] T. Saito and M. Rehmsmeier, "The precision-recall plot is more informative than the ROC plot when evaluating binary classifiers on imbalanced datasets," *PLOS ONE*, vol. 10, no. 3, G. Brock, Ed., e0118432, Mar. 2015. DOI: 10.1371/journal.pone.0118432. [Online]. Available: https://doi.org/10.1371/journal.pone.0118432.

[25] R. Yacouby and D. Axman, "Probabilistic extension of precision, recall, and f1 score for more thorough evaluation of classification models," in *Proceedings of the First Workshop on Evaluation and Comparison of NLP Systems*, Online: Association for Computational Linguistics, Nov. 2020, pp. 79–91. DOI: 10.18653/v1/2020.eval4nlp-1.9. [Online]. Available: https://aclanthology.org/2020.eval4nlp-1.9.

[26] M. Grandini, E. Bagli, and G. Visani, *Metrics for multi-class classification: An overview*, 2020. arXiv: 2008.05756 [stat.ML].

[27] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, no. 4, pp. 312–315, Dec. 2020. DOI: 10.1016/j.icte.2020.04.010. [Online]. Available: https://doi.org/10.1016/j.icte.2020.04.010.

[28] N. Ketkar, "Stochastic gradient descent," in *Deep Learning with Python*, Apress, 2017, pp. 113–132. DOI: 10.1007/978-1-4842-2766-4_8. [Online]. Available: https://doi.org/10.1007/978-1-4842-2766-4_8.

[29] Z. Zhang, "Improved adam optimizer for deep neural networks," in *2018 IEEE/ACM 26th International Symposium on Quality of Service (IWQoS)*, 2018, pp. 1–2. DOI: 10.1109/IWQoS.2018.8624183.

[30] P. Zhou, J. Feng, C. Ma, C. Xiong, S. Hoi, and W. E, *Towards theoretically understanding why sgd generalizes better than adam in deep learning*, 2021. arXiv: 2010.05627 [cs.LG].

[31] Z. Zahisham, C. P. Lee, and K. M. Lim, "Food recognition with resnet-50," in *2020 IEEE 2nd International Conference on Artificial Intelligence in Engineering and Technology (IICAIET)*, 2020, pp. 1–5. DOI: 10.1109/IICAIET49801.2020.9257825.

[32] A. Dicks Herrán, *Faster R-CNN in neutron images*, version 2.0.4, 2023. DOI: 10.5281/zenodo.1234. [Online]. Available: https://github.com/anedicksh/Faster-R-CNN-neutron-images.

[33] T. Sadad, A. Rehman, A. Munir, *et al.*, "Brain tumor detection and multi-classification using advanced deep learning techniques," *Microscopy Research and Technique*, vol. 84, no. 6, pp. 1296–1308, Jan. 2021. DOI: 10.1002/jemt.23688. [Online]. Available: https://doi.org/10.1002/jemt.23688.

[34] N. S. Keskar and R. Socher, "Improving generalization performance by switching from adam to SGD," *CoRR*, vol. abs/1712.07628, 2017. arXiv: `1712.07628`. [Online]. Available: `http://arxiv.org/abs/1712.07628`.

[35] A. Mahmoud, S. Mohamed, R. El-Khoribi, H. AbdelSalam, and and, "Object detection using adaptive mask RCNN in optical remote sensing images," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 1, pp. 65–76, Feb. 2020. DOI: `10.22266/ijies2020.0229.07`. [Online]. Available: `https://doi.org/10.22266/ijies2020.0229.07`.

[36] Z. Shen and M. Savvides, "MEAL V2: boosting vanilla resnet-50 to 80%+ top-1 accuracy on imagenet without tricks," *CoRR*, vol. abs/2009.08453, 2020. arXiv: `2009.08453`. [Online]. Available: `https://arxiv.org/abs/2009.08453`.

[37] L. N. Smith, "A disciplined approach to neural network hyper-parameters: Part 1 - learning rate, batch size, momentum, and weight decay," *CoRR*, vol. abs/1803.09820, 2018. arXiv: `1803.09820`. [Online]. Available: `http://arxiv.org/abs/1803.09820`.

[38] I. Kandel and M. Castelli, "The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset," *ICT Express*, vol. 6, pp. 312–315, 2020.

[39] N. Bodla, B. Singh, R. Chellappa, and L. S. Davis, "Improving object detection with one line of code," *CoRR*, vol. abs/1704.04503, 2017. arXiv: `1704.04503`. [Online]. Available: `http://arxiv.org/abs/1704.04503`.