UNIVERSITY OF TARTU
Faculty of Science and Technology
Institute of Computer Science
Computer Science Curriculum

Aral Açıkalın

Collecting and Using a Labeled Dataset of NATO Mission Task Symbols to Improve and Benchmark Detection Models

Master's Thesis (30 ECTS)

Supervisor(s): Ardi Tampuu, PhD

Tartu 2023

Collecting and Using a Labeled Dataset of NATO Mission Task Symbols to Improve and Benchmark Detection Models

Abstract:

Neural networks are commonly used for object detection tasks but require immense amounts of data to train. For the task of North Atlantic Treaty Organization (NATO) mission task symbol detection using object detection neural networks, it is not possible to meet the data requirements. Additionally, labeling mission task symbols is very time-consuming and costly.

This thesis aims to collect and label a dataset of NATO mission task symbols, propose a part of it as a benchmark for our solutions and future solutions, and finally propose different methods to use a part of the scarce collected data to improve the performance of our object detection models. YOLOv5 neural network is selected and used to experiment with different ways of using the scarce collected data. As a result, 113 images were collected and labeled. Five performance metrics are proposed for the benchmark. Finally, it was discovered that when dataset size is limited, extracting information from the dataset and using it to generate artificial data improves performance compared to directly introducing the scarce dataset to symbol detection models.

Keywords:

Machine learning, deep learning, computer vision, object detection, symbol detection, image processing, benchmark

CERCS: P170 - Computer science, numerical analysis, systems, control, P176 - Artificial intelligence, T111 Imaging, image processing

NATO lahingutoimingute sümbolite märgendatud andmestiku kogumine ja kasutamine tuvastusmudelite täiustamiseks ja võrdlemiseks

Lühikokkuvõte:

Tehisnärvivõrke kasutatakse laialdaselt objektide tuvastamiseks piltidelt, kuid nende treenimiseks on vaja väga palju andmeid. Põhja-Atlandi Lepingu Organisatsiooni (NATO) lahingutoimingute sümbolite tuvastamiseks objektituvastuse närvivõrkude abil ei ole neid nõudmisi andmehulgale võimalik täita. Lisaks oleks lahingutoimingute sümbolite märgendamine päris andmetel väga aeganõudev ja kulukas.

Selle lõputöö eesmärk on koguda ja märgendada NATO lahingutoimingute sümbolite andmestik, pakkuda osa sellest meie lahenduste ja tulevaste lahenduste võrdlusaluseks ning lisaks pakkuda välja erinevad meetodid, kuidas kasutada osa nappidest kogutud andmetest meie mudelite võimekuse parandamiseks. Valisime objektituvastuseks YOLOv5 närvivõrgu ja kasutasime seda andmete erinevate kasutusviisidega katsetamiseks. Töö tulemusena kogusime ja märgendasime 113 pilti. Välja pakutud võrdlusaluse jaoks valisime välja viis mõõdikut. Lisaks leidsime, et kui andmestiku suurus on piiratud, on

andmekogumist teabe eraldamine ja selle kasutamine kunstlike andmete genereerimiseks efektiivsem võrreldes napi andmestiku otsese kasutuselevõtuga treeningandmepunktidena.

Võtmesõnad:
Masinõpe, sügavõpe, masinnägemine, objektituvastus, sümbolituvastus, pilditöötlus, võrdlusalus

CERCS: P170 - Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria), P176 - Tehisintellekt, T111 Pilditehnika

Contents

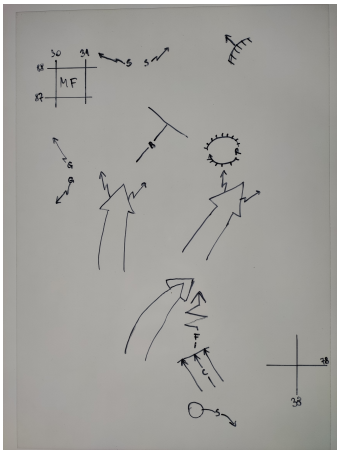
1 Introduction

In military operations, correct assessment of plans and the decision-making speed are very important. Establishing a common language that all users clearly understand in joint military operations is imperative. To achieve this, members of the North Atlantic Treaty Organization (NATO) use a joint symbology [app11] for military plans. For example, the joint symbology allows the military tactical plans drawn by an Estonian to be understandable to a Frenchman.

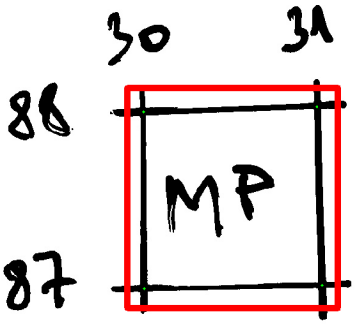
Most of the time, military operations need a lot of information. Different kinds of information, like friendly planned actions, enemy unit locations, etc., are drawn to different transparent films to be overlaid on a map. Putting many films with different content in the right location on a large map is time-consuming but necessary to obtain an overview of the situation. To reduce time consumed on this process, "Kaitseväe olukorra- ja lahinguteadlikkuse süsteem"(KOLT), or in English, "The situational and combat awareness system of the Defense Forces" was created. This system allows plans to be drawn on a digital map, and once the plans have been digitized, they will never need to be aligned with the map again. KOLT also allows analyzing digitilized plans to be more convenient. One can zoom in or out and change the information type displayed with a button click. Even though KOLT helps save time by not having to align plans to the map, the information on the plans must still be entered into the system, which is also time-consuming. For an officer on the front line, drawing plans by hand is still faster than entering them directly into the digital system. That is why, currently, original plans are made on film and they are only digitized in the headquarters using KOLT. To investigate the possibility of automatizing and speeding up the digitalization process, Estonian National Defence College/ Kaitseväe Akadeemia has started a collaboration project with the University of Tartu Institute of Computer Science (ICS) for the automatic detection of a certain type of symbols. This task comes down to acquiring a high-quality image of the plan, determining the transformation matrix between the image's pixels and the digital map's coordinate system, localizing and identifying the symbols, and detecting their poses.

The ICS research team decided to tackle this problem by acquiring a high-quality image of the plan with the help of a lightboard, detecting the location markers on the plan, aligning the image with the digital map, localizing and identifying the symbols with the YOLO (You Only Look Once) object detection neural network, and finally detect the poses of the symbols which is a subject of a concurrent thesis (this process can be seen in Figure 1.). However, modern object detection approaches need thousands of data points per object class. For example, creators of a recent YOLO version recommend using more than 10,000 instances of each object type in the data [ea21]. Labeling thousands of images and tens of thousands of symbols is very labor-intensive. In our experience, it took one day to label approximately 24 images/156 symbols. Some professional labeling

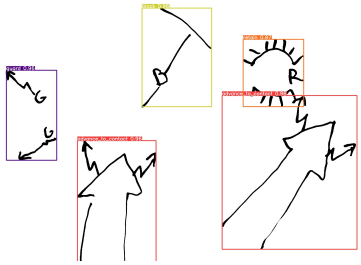
services[1] charge up to 5 euros per hour, and freelancers offer services with prices starting from 4 euros and going up to 15 euros[2], but even with these services, quality is not guaranteed. In addition, labeling military plans is not a trivial task. Identifying symbols, in some cases, is very challenging, even for a human. The challenging and sensitive nature of the military plans makes trusting a third party to label military plans unreliable. Moreover, there are not that many plans the military academy can share with us.



(a) First, a photograph of the film is acquired by placing it on lightboard.



(b) Then, the image is binarized and the location markers are identified. Intersection points of the marker lines are determined. The transformation matrix between pixel and geographical space is calculated using the real geographical locations for the points.



(c) Finally, YOLO network identifies symbols on the binarized image. The geographical locations for these symbols are determined using the previously calculated transformation matrix.

Figure 1. Proposed plan of digitizing military plans.

To achieve sufficient data for training object detection networks, a generator was created by the ICS team. However, it is unable to produce real-like data, and it only imitates real data to a certain degree. To know how good models trained on this generated data are on real data, one needs a labeled test dataset of real data, which is the first contribution of the thesis. Secondly, to help generalize the models to the real data, it may be useful to use a part of the scarce labeled data in the training process to expose the model to its distribution. The second contribution of this thesis is testing obvious methods and inventing non-obvious ways of using relatively few real examples to increase the

[1]Abelling labeling service. https://abelling.ai/ Accessed on 25.04.2023
[2]Search conducted on https://www.fiverr.com/ on 25.04.2023

robustness of our models on real data. The proposed dataset, alongside baseline models and evaluation metrics, forms a benchmark. With the labeled real dataset, baseline model code, and the data generator made publicly available, other scientists from other countries (hopefully NATO member countries) can create similar and improved solutions.

1.1 Contributions

This thesis aims to:

- Collect and label a dataset of hand-drawn military plans for detecting NATO mission task symbols.
- A methodology to integrate a part of the collected dataset into object detection algorithm training.
- Propose metrics and a subset of the collected dataset as a benchmark.

1.2 Outline

The thesis is organized as follows: Section 2 provides background information on benchmarks, object detection, data generation approaches and discusses related work on symbol detection. Section 3 presents the methods used here for collecting data and creating a benchmark. It also covers how to use an object detection algorithm and different ways of incorporating real data to improve generalization to its distribution. Section 4 presents the collected dataset as one of the results of this thesis and proposes a part of it as a benchmark. In addition, the conducted experiment results are presented and discussed. Section 5 concludes the thesis by summarizing the findings and their implications and proposing future directions for research.

2 Background

In this section, the object detection task is explained, then a literature review on symbol detection and techniques for dealing with scarce data are presented. Additionally, background on benchmarking is covered.

2.1 Object Detection

Object detection is a computer vision task that involves identifying and localizing objects within an image or a video stream. The goal of object detection is to detect all instances of objects of interest within an image and to provide a bounding box around each object along with a classification label indicating the category of the object (see Figure 2) [AFG20, LMB+14].

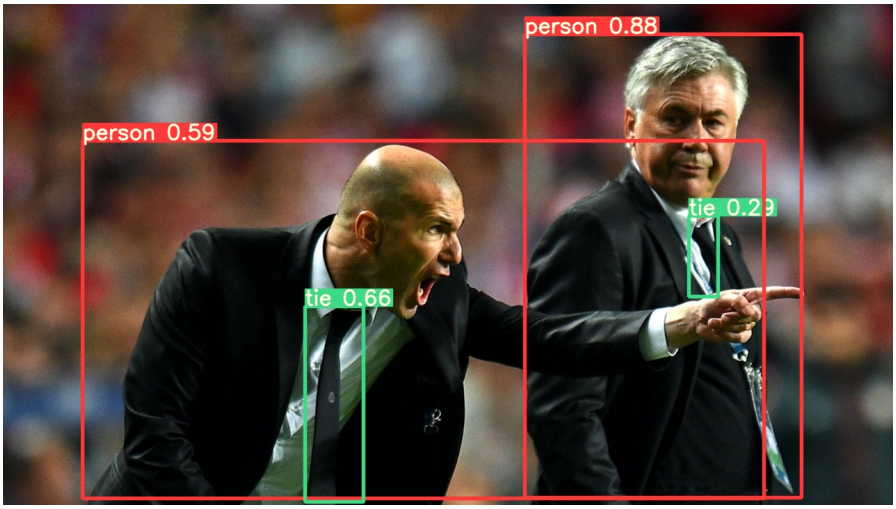


Figure 2. YOLOv5 output visualized. Output includes object class and bounding box information.[ea21]

.

Object detection is an important task in computer vision, with many applications such as surveillance, autonomous driving, and robotics [ZCS+23]. The development of accurate and efficient object detection models is an active area of research with increasing interest, and benchmark datasets such as COCO [LMB+14], and Pascal VOC [EVGW+09] are widely used to evaluate the performance of different models and techniques [ZCS+23].

Neural networks have been helpful in tackling object detection datasets. Girshick et al. [GDDM14] proposed Region with Convolutional Neural Network features(RCNN) for object detection and semantic segmentation tasks, and from there, many different methods were created and used. YOLO [RDGF15] is another neural network designed

for object detection tasks. YOLO outperformed RCNNs in terms of accuracy and speed [ZCS+23]. For this thesis, we used YOLOv5, the fifth iteration of the architecture. The choice of using the YOLO neural network and the fifth iteration of it was a team decision.

The spatial correctness of a predicted bounding box around the object is counted with the help of the intersection over union (IoU) metric. This metric measures the degree of overlap between the ground truth bounding box and the predicted bounding box. The formula for IoU can be seen on the Figure 3. Depending on the task, different spatial precision can be needed, so a threshold for IoU is selected. Only if the IoU between the object and the detection exceeds this threshold, the prediction is counted as correct, i.e. as true positive[PNDS20]. Figure 4 shows a simple example of this.

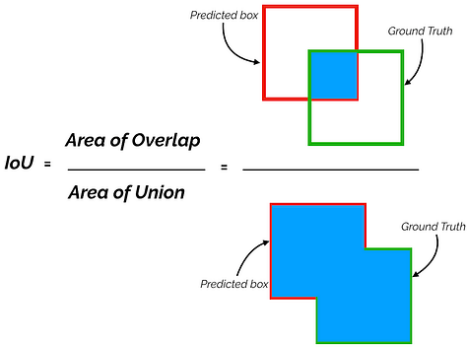


Figure 3. Formula of IoU visualized[3].

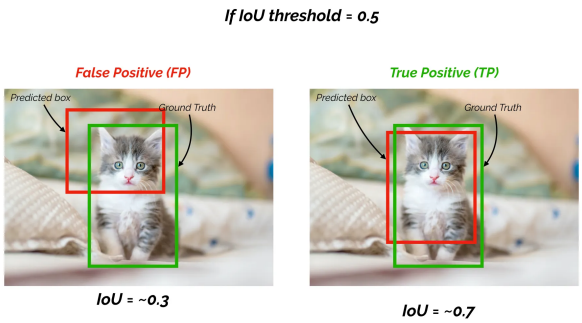


Figure 4. IoU threshold of 0.5 determining the correctness of the detection[3].

When the IoU value of the predicted bounding box with the ground truth bounding box exceeds the threshold set, and the predicted class matches with the ground truth, then that detection is counted as true positive(TP). If any of these conditions is not satisfied, that detection is counted as a false positive(FP). When an object is not detected at all, or the detected object class does not match with the ground truth, or the predicted bounding box and the ground truth cannot be matched with sufficient IoU, then a false negative(FN) is counted.

The performance of object detection models is typically evaluated using various metrics such as precision, recall, and average precision (AP). Precision measures the fraction of correctly detected objects out of all detected objects for a class , while recall measures the fraction of correctly detected objects out of all actual objects in the image for a class. Equations 1 and 1 show the formulas of precision and recall. In object detection, average precision (AP) is defined as the area under the curve of the precision-recall plot. The formula for AP can be seen in Equation 3

[3]Figures taken from https://towardsdatascience.com/map-mean-average-precision-might-confuse-you-5956f1bfa9e2 Accessed on 04.05.2023

$$Precision(P) = \frac{TP}{FP + TP} \tag{1}$$

$$Recall(R) = \frac{TP}{FN + TP} \tag{2}$$

$$AveragePrecision(AP) = \int_0^1 P(R)\, dR \tag{3}$$

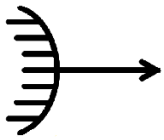
The AP metric was originally introduced in Pascal Visual Object Classes (Pascal VOC) challenge [EVGW+09]and was proposed to be used with the IoU threshold of 0.5. The AP@0.5 (AP at 0.5 IoU threshold) was very commonly used. To get an overview of the model, they also used mean average precision(mAP), which is the mean of APs over all classes, resulting in the still commonly used metric referred to as mAP@0.5. Additionally, the COCO challenge introduced mAP@0.5-0.95, which means they also averaged the APs at different IoU thresholds between 0.5 and 0.95 [ZCS+23]. This metric is stricter compared to mAP@0.5 for the accuracy of the localization. Today, mAP@0.5-0.95 is a very commonly used metric in object detection tasks.

2.2 Symbol Detection

Here, the data we use is very different from natural images. Our task is to detect and localize symbols, more specifically NATO mission task symbols, drawn on a white background, whereas typical object detection tasks deal with photos of real-world objects. For instance, popular object detection datasets like COCO focus on detecting everyday objects in the real world, such as persons, vehicles, and animals, rather than handwritten symbols. Therefore, this sub-section looks more specifically into symbol detection on data types with symbols drawn on a white background.

Elyan et al. [EJAG20] researched detecting symbols on 2D engineering drawings that contain shapes, symbols, lines, and text drawn on a white background. They used YOLO to detect a set of these engineering symbols. They reported 94.9% accuracy on their test set. However, handwritten symbols are more challenging to detect because they are more variable than printed, standard engineering symbols. Figure 5 shows an excellent example of the difference between hand and digitally drawn or printed symbols. Many factors, like different writing instruments, used surface, and haste while drawing, may cause the variation between hand and digitally drawn symbols. A symbol drawn by someone may differ significantly from another person's drawing of the same symbol [PSP+19]. Other work similar to symbol detection is done by Pizarro et al. [PHSS22] and by Ziran et al. [ZM18] on 2D floor plans that contain text and shapes representing furniture on white background. Pizarro et al. reported that 30% of their data was not drawn compliant with a standard rule, which made detection difficult. Ziran et al. reported 0.31 mAP@0.5 on a scarce test set which was diverse and non-standard. The

low performance on floor plans compared to engineering drawings exemplifies clearly the additional difficulty that hand-writing introduces.

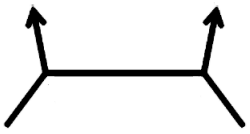


(a) Digitally drawn ambush symbol.

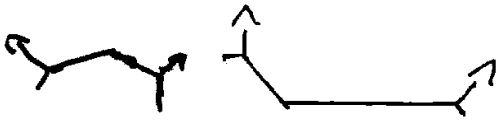
(b) Hand-drawn ambush symbol from the labeled data set from this thesis.

Figure 5. Hand drawn symbol. Comparison of hand and digitally drawn symbols.

Today, it is common in the military academy to draw military plans onto transparent plastic films using markers. The transparency of the plastic films later allows them to be overlaid on the maps. To create military plans, officers use the NATO Joint Military symbology and use the symbols defined there [app11]. Mission task symbols are a subtype of the symbol group control measure symbols. Control measure symbols aim to provide operational information [app11]. All the symbols are hand-drawn, sometimes with the use of a ruler, and may have irregular shapes (see Figure 6). In addition, their poses and rotations might vary.



(a) Digitally drawn support by fire symbol template



(b) Hand drawn "support by fire" symbol which has irregular shape

Figure 6. Irregular symbol shapes. Comparison between symbol template and irregular symbol shapes are shown.

The quantity of information on the film varies, and a variety of other symbols beyond mission task symbols can be present, some of which can be similar in shape and size to the symbols we wish to detect. In conclusion, the task is more complicated than many commonly studied symbol detection tasks as the handwritings vary strongly, different poses and all rotations can exist, and symbols are not placed in standard locations as in handwritten text where letters follow each other in a line.

2.3 Dealing with Scarce Data

Obtaining high-quality data can be difficult in certain fields due to privacy concerns, limited access, or the high cost of labeling data. Data augmentations aim to extract more information from the original dataset and increase variation. For images, flipping and rotating the data is a widely known method to increase the dataset size [SK19]. In [JSA+20], authors augment their dataset of limited size by first creating horizontally flipped copies and then rotating all images with increments of 20 degrees to increase the dataset size significantly. However, not all types of data can be augmented by flipping. For example, text would lose its meaning when flipped, and because of that, it cannot be flipped [LHY21]. In object detection, robustness to object position is vital because objects can appear in any part of an image. Translation augmentations can reduce position bias by translating the image to the left, right, top, or bottom[SK19]. There are many types of other augmentation techniques aiming to reduce different kinds of biases that the original dataset can cause, many of which are discussed by Shorten et al.[SK19]

Another solution to combat data scarcity is to use artificial data, which can be generated using different methods. For example, Thambawita et al. [TSS+22] used Generative Adversarial Networks (GANs) to generate synthetic data for medical image segmentation, which can be used to reduce the time and cost associated with data acquisition. GANs are a machine learning architecture that can generate new data by learning from an existing unlabeled dataset [GPAM+14]. In autonomous driving, Ghosh et al. [GBC16] also use GANs to generate synthetic data to cover a wider range of situations and conditions not present in their natural dataset. Artificial data can be a valuable tool for addressing data scarcity or lack of variation, allowing researchers to generate new larger datasets that can improve the performance of machine learning models.

For tackling the task of symbol detection, we opted to use generating artificial data by a rule-based generator, which places augmented and rotated symbols on a white canvas. Further augmentation is performed during training by the YOLOv5's default training pipeline. Exact details are discussed in Section 3.

2.4 Benchmark

A benchmark is a standard or a reference point against which other solutions can be measured or compared. Scientifically, it is a set of tests, metrics, or procedures used to evaluate the performance of different systems, algorithms, or machine learning models.

In artificial intelligence(AI) and machine learning, a benchmark is used to standardize a set of tasks, metrics, and data so that everyone can evaluate and compare the performance of different models and algorithms on an equal basis. The metrics and data vary from task to task. Researchers are creating and using benchmarks for the tasks in their respective fields. Some examples of these benchmarks are;

- Image classification - ImageNet [DDS+09]
- Object detection - Common Objects in Context (COCO) [LMB+14]
- Semantic segmentation - Cityscapes dataset [COR+16]
- Speech recognition - Libri Light benchmark [KRZ+19]

Creating a good benchmark can be challenging. A benchmark should have a clear objective defined, a task representative of real-world problems. Appropriate metrics should be selected; the metrics should be relevant, reliable, and interpretable to capture the performance of the solutions using them. A benchmark should include a representative dataset, representing the real-world problem the benchmark is trying to address.

In object detection, various metrics like precision, recall, mAP@0.5, and mAP@0.5-0.95 may be reported because one metric cannot capture the performance alone. Most commonly in recent benchmarks, mAP@0.5-0.95 is chosen as a main metric [ZCS+23]. A benchmark should consider the goal of the task and the end user when choosing the metric or metrics. For example, if the goal/end-user likely requires high classification and localization precisions, the benchmark metrics should reflect this. As a second example, if the end-user needs good performance in all object classes, average metrics do not suffice and the minimum over classes should be reported. In case the user intention is unclear, the benchmarks may report a variety of metrics.

3 Methods

This thesis aims to improve symbol detection models' performance using scarce real data and propose a benchmark to evaluate these models. Therefore this section describes the methods used to train object detection neural networks. The generator used is described, and real data collection and labeling methods are explained, then methods used to sort the data are described. Additionally, methods for using the scarce collected dataset to improve symbol detection models are presented. Finally, the selected metrics for evaluation and, experiment setup are described.

3.1 You Only Look Once (YOLO)

You Only Look Once (YOLO) is an object detection neural network architecture heavily influenced by the human ability to glance at an image and instantly know what objects are in it, where they are, and how they interact. Humans can do this process very fast and accurately, and the creators of YOLO aimed to achieve fast and accurate detection of objects in an image [RDGF15].

YOLO uses convolutional neural networks(CNNs) to detect and locate objects within an image. It takes an image as input and processes it through a series of convolutional layers to extract features, then these features are used to generate a set of bounding boxes and corresponding class probabilities as output, which indicate the location and class of the objects within the image. Later versions of YOLO architecture can detect more than 9000 object categories and still run in real-time [RF16].

YOLO is a supervised learning object detection neural network. Because of this, it needs labeled data. The labels for objects should include the bounding box information for the object. The creators of YOLO implemented their scripts in a way that requires labels as class index, the center x and y coordinates, and the width and height of the bounding box in pixels. The labels for objects should be in the form of the following;

$$< objectclass >< x >< y >< width >< height >$$

Except for the object class index, all of the bounding box information must be relative to the image and normalized with the width and height of the image (see Figure 7) [PNDS20]. This way, the bounding boxes can still be used when the images are resized. However, it is important to note that the actual YOLO implementation used in this thesis does not use this labeling notation as input or output. The YOLOv5 implementation uses non-normalized coordinates of the top left and bottom right corners of the bounding boxes instead of normalized center x and y coordinates and width and height. The conversion is done internally by the YOLO scripts.

For this thesis, we used YOLOv5, the fifth iteration of the architecture [ea21]. The choice of using the YOLO neural network and the fifth iteration of it was a team decision.

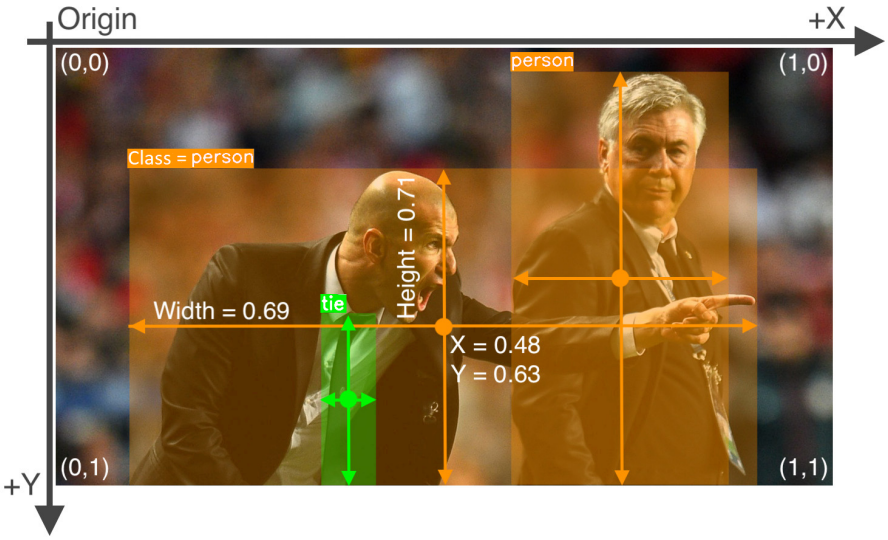


Figure 7. Normalized input labels of YOLOv5 visualized. Objects are labeled in the form of class, X, Y, Width, Height [ea21]

.

For training a YOLOv5 model, fully labeled datasets are required. There should be at least two different datasets: training and validation sets. The training set is used to train the model and the validation set is used to validate the training after every epoch. After completing an epoch, YOLOv5 saves the current weights of the model by creating a checkpoint. This checkpoint is overridden after every epoch. Additionally, it saves the best checkpoint by comparing the mAP@0.5 and mAP@0.5-95 scores of the validation set on the last created checkpoint and the previous best checkpoint. The user needs to set the number of epochs the model should train. However, if the validation set scores do not improve for a set number of epochs, the training will early stop and not complete the number of epochs set..

According to the recommendations provided by the creators of YOLOv5, a training dataset should ideally consist of more than 1,500 images and 10,000 object instances per class [ea21]. For instance, if there are three object classes, the training dataset should have 30,000 objects labeled in approximately 4,500 images. The dataset must accurately represent the target environment where the solution will be deployed, reflecting real-world scenarios where the object detection model will be used. It should also maintain consistency, ensuring that all instances of all classes in all images are labeled, and the bounding boxes of the objects should closely enclose the objects. Additionally, the creators of YOLOv5 suggest including background images in the dataset to reduce possible false positive detections [ea21]. Background images refer to images without any objects, which can help the model better understand the absence of objects in certain

situations, thus enhancing its performance in real-world deployment scenarios.

3.2 Generator

The main data for training YOLO for our mission task symbol detection task comes from a generator specifically designed by Mihkel Lepson to generate NATO mission task symbols on an empty canvas alongside other required military markings like texts and other kinds of symbols that can appear on the studied plastic films.

First, Mihkel drew sets of NATO mission task symbols with different drawing styles (see Figure 8). The generator uses these symbol sets as templates to generate data. The templates are binarized, so the leftmost, rightmost, topmost, and bottommost black pixels of a rotated template image always determine the bounding box of the symbol.

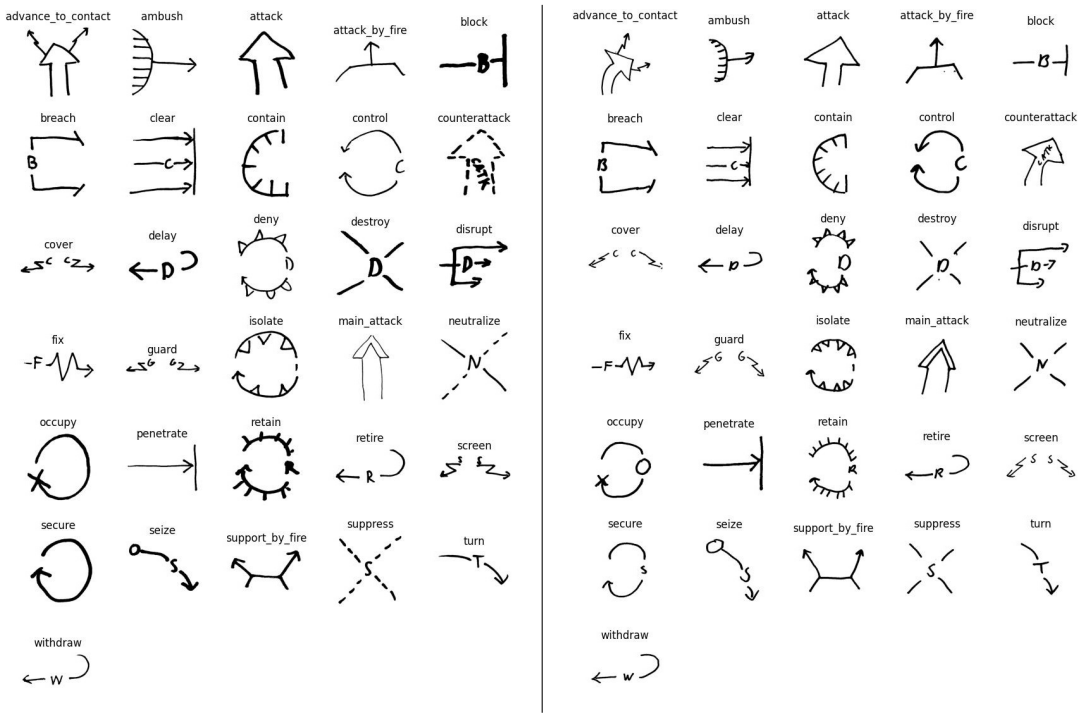


Figure 8. Two sets of template symbols used by the generator.

To create an example input, the generator randomly picks mission task symbols with uniform distribution, picks a random realization of these symbols in the aforementioned set of templates, randomly rotates and distorts the picked symbol, and places it in random location on the canvas. The type and bounding box information of the symbol is recorded in the label file. Other elements such as phase lines, map location markers and certain texts are added to make the generated "film" look closer to the real data.

The generator will create images at a resolution of 4624 by 3468 because it is the resolution at which the templates for symbols were digitized. After creating the image, the generator can resize the image to any desired dimension.

The biggest limitation of this generator is that it cannot place the symbols to achieve an overall plan layout that would make military sense (see Figures 9, 10). It is unfeasible to define the rules that can create realistic military plans. Therefore, the symbol placement distribution of the generated and the real data will always differ. On the other hand, a model that can detect non-sensically randomly placed symbols, should generalize reasonably well to any systematic spatial pattern of symbols, i.e. the real distribution is contained within the random placement.

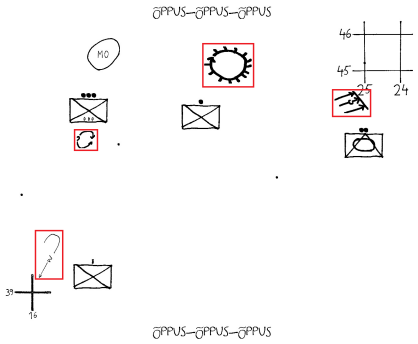


Figure 9. Random symbol placements in generated data. (Symbols are marked in red boxes.)

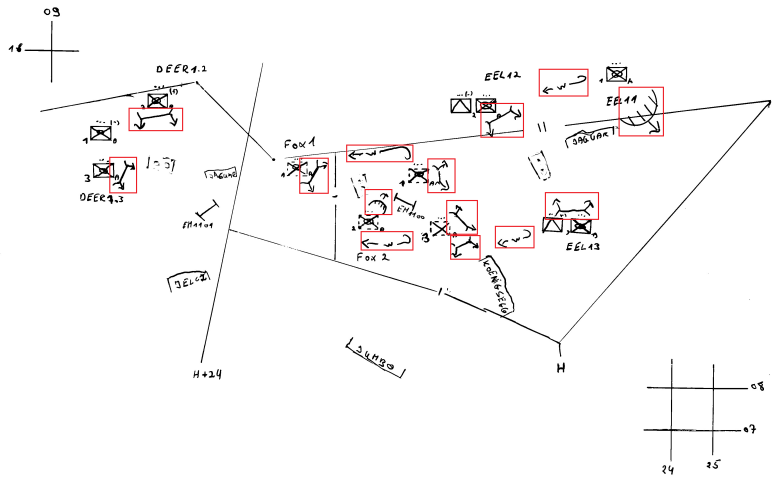


Figure 10. Symbol placements in real data. (Symbols are marked in red boxes.) The symbols are placed in a way that makes military sense.

3.3 Collecting Labeled Real Data

To create a benchmark, a dataset needs to be collected. However, due to the uniqueness of the data, it needs to be first digitized and then labeled. The existing labeling tools are insufficient to create a fully labeled dataset. Although pose detection for the symbols is not a part of this thesis, it is required for digitizing the military plans completely. For object detection labeling the identity and bounding boxes of objects would be enough. However, the rotations of the NATO mission task symbols must also be labeled to create a fully labeled dataset.

3.3.1 Digitizing the Data

The data we get from the military academy is in its physical form, on plastic films, and hand-drawn. It needs to be digitized by photographing or scanning to process the data further. Acquiring images by photographing is chosen over scanning because of the reasons of convenience for the collaboration partners. Scanners are not very user-friendly for remote locations when the plan size is bigger than the A4 size. Different methods for acquiring images were analyzed thoroughly in a report co-authored by the author of this thesis and submitted to the collaboration partners, and acquiring images by photographing was preferred. Therefore, for this thesis, we used the method of photographing.

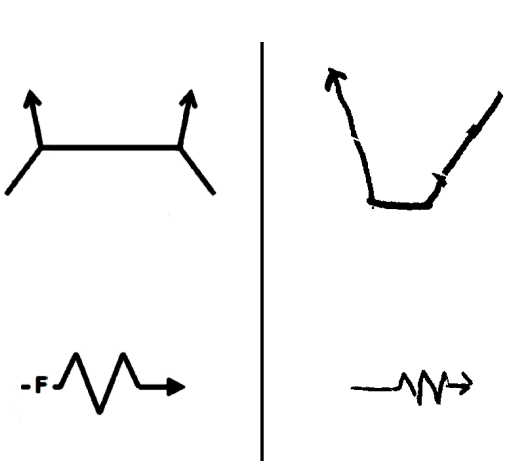
Photographing military plans is not always straightforward. The provided data is not always preserved in the best conditions. Some films may be folded or rolled, making it hard to lay them on the ground flat. Getting the best quality photographs required fixing these issues, which is a time-consuming task.

To ensure high-quality data, plastic films should be photographed in a way that minimizes reflections, as they are highly reflective by nature. In addition, the obtained images should be binarized because the models trained expect binarized input. Thresholding techniques can be used for this, but achieving perfect binarization can be challenging due to the difficulty of uniformly lighting the film. To address this issue, we utilized a lightboard to illuminate the films from underneath evenly and photographed them in a dark room without direct lighting on the films. This method allows for applying OTSU thresholding [Ots79], which yields the desired binarized version of the photograph. The author of this thesis previously implemented the process of photographing and binarizing the data during an internship with the research team.

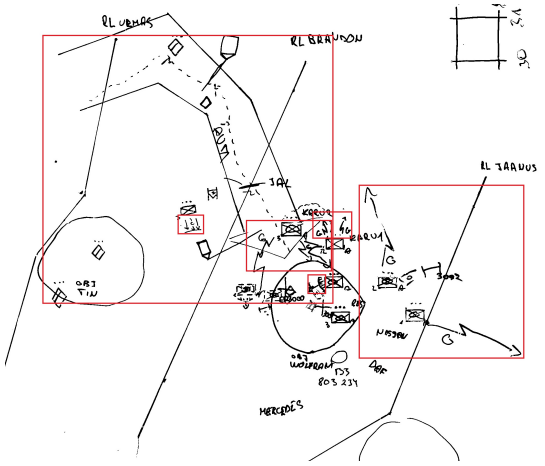
3.3.2 Labeling and Sorting the Data

Labeling data, in general, is a difficult and time-consuming task, but labeling the data we collected is more difficult that labeling natural images. For example, determining common objects like cars or animals is not much of a challenge because we see them in our daily lives and are accustomed to them, however, determining the symbol type on a

military plan is difficult. The difficulty comes from many different factors. Hand-drawn symbols may have missing elements, like arrowheads or characters (see Figure 11a). Some symbols may be drawn very similarly to other symbols. Overlapping symbols may make it hard to recognize a symbol (see Figure 11b). In these cases, confirmation from a person who is more knowledgeable of these symbols and understands the idea of the overall plan, may be required to label them correctly. All this increases the time and effort it takes to label military plans and makes it difficult to outsource them.



(a) Examples of hand-drawn symbols which have missing elements can be seen on the right and what they should look like can be seen on the left.



(b) Military plan with densely placed symbols. Symbols are marked with red boxes

Figure 11. Symbols that are hard to identify in the real data.

In our pipeline of data labeling, we start by labeling the identity of the symbols and their bounding boxes. We used a tool called Labelimg [Tzu15]. With the help of this tool, we labeled every symbol in every photographed and binarized image available to us with one-pixel precision. This process can be seen in Figure 12. As the output, each image is associated with a file containing the symbols' identities, locations, and sizes.

The final subtask of digitizing NATO mission task symbols is detecting their poses. After detecting the object, it is also necessary to detect its rotation/pose. The mission task symbols can be drawn in plans in any rotation. Although we have not worked on detecting the rotations of the symbols in this thesis, rotations still need to be labeled to form a dataset allowing to solve the problem fully.

Because of this problem's uniqueness, no tool was found to help label the rotations of objects and we needed to create this tool ourselves as part of this thesis. This tool overlays a template of the symbol on top of the symbol to be rotation-labeled using the identity and bounding box information available for the Labelimg-labeled images.

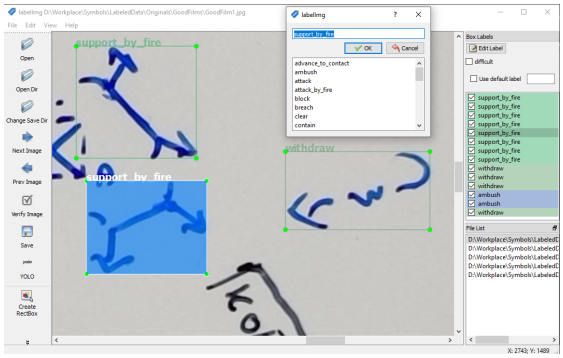


Figure 12. "Labelimg" tool. It is used to label bounding boxes of symbols.

The user can then rotate the template image starting from zero rotation until its rotation matches the symbol to be labeled (see Figure 14). For all symbols, the zero default rotation must be defined. For some symbols, this choice can be made arbitrarily or is obvious. For the symbols that can have irregular shapes/trajectories (see Figure 13), the zero rotation is defined as the rotation that results in the symbol “starting” (e.g., the tail of the arrow) from the bottom center of the bounding box.

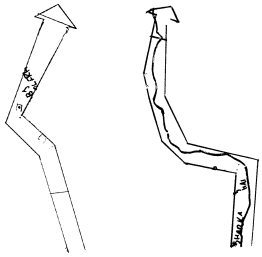


Figure 13. "Attack" symbols with complex trajectories.

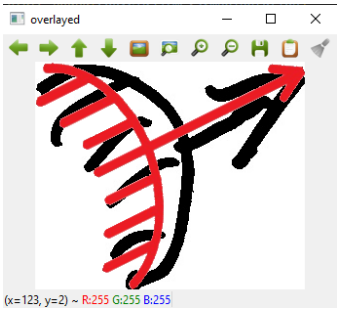


Figure 14. Rotation labeling tool. The red template symbol rotation is matched with the underlying symbol to label the rotation.

Defining rotation is insufficient to capture the entire pose for symbols like “attack”(see

Figure 13). Labeling the entire trajectory will be extremely time-consuming and will likely require developing further tools. This will not fit into the time frame of this thesis. Detection of these complex poses is part of a concurrent thesis, and the solutions are not yet fixed.

The identity, location, and rotation labeled data acquired were divided into two sets. One is intended for training symbol detection models, and the other is to benchmark them. The data was sorted by the quality of its contents. Higher-quality data was used as the test set, and lower-quality data was used as the training set. The following contributing factors determined the quality:

1. Readability of the symbols in the data, influenced by the density of other symbols and handwriting

2. The visibility of the lines that create the symbols.

The readability of symbols in the data refers to the clarity of the symbol shapes without any clutter or distortion within or near them. Additionally, the strength of the lines used to create the symbols is important, as lines that are inconsistent or have gaps can negatively impact the ability of the model to detect the symbol. For example, in the case of drawings on plastic films, which do not absorb ink as paper does, lines may appear empty or have uneven coverage, leading to lower-quality semi-transparent lines. A line with consistent darkness and width is considered to be of higher quality. An illustration of a low-quality line is shown in Figure 15. With this selection method, we reflect the military's own concurrent goal of producing films with less unnecessary content (adhering to so-called minimal military requirements) and with more up-to-standard handwriting with prescribed 0.6mm markers. Supposedly, during future deployment, the handwritten films will be more likely similar to our test set, not the training part of the digitized data.

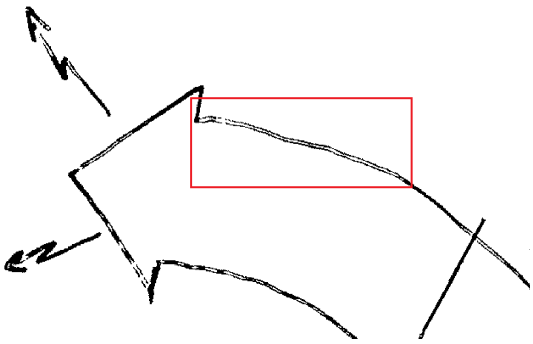
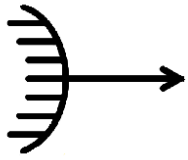


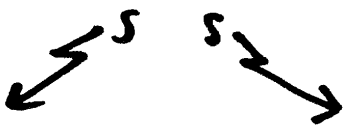
Figure 15. Low-quality line in a symbol. The symbol is extracted from the collected real data and the low-quality line is marked with a red box.

Additionally, some symbols were erased from the images in the test split due to being highly irregular and not up to military standards, i.e. our collaboration partner assumes they will not produce such low quality in the future and does not expect solutions to detect these symbols. These symbols were either missing a vital part of the symbol or drawn very irregularly. Examples of erased symbols can be seen in Figure 16.

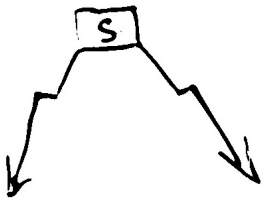


(a) Correctly drawn "ambush" symbol.

(b) The "Ambush" symbol is drawn with the missing arrow.



(c) Correctly drawn "screen" symbol.



(d) The "Screen" symbol drawn in an irregular way.

Figure 16. Erased symbols from the test split. Examples of erased irregular symbols from the test split and their correct counterparts are shown.

Another vital factor for splitting the data was the symbol instance distribution. Because the data is limited and does not include all symbol classes, creating a split with perfectly equal distribution is impossible. However, the split was created to have similar distributions in training and test sets. In summary, the train and test split is done considering data quality and data balance.

3.4 Bridging the Gap Between Generated and Real Data

Using a generator to provide data for training the YOLO object detection model has proven to be a necessary and fruitful approach. Its main limitation of not capturing the real data distribution can be remedied by using our collected real data in different ways during the training. Combining real and generated data has been used in different fields like autonomous driving [GBC16]. For symbol detection, more specifically, mission task symbol detection, we propose a number of different ways of using scarce real data to improve symbol detection performance on the test set.

3.4.1 Using Real Data as is

The most straightforward and trivial way of using the data collected is using the labeled images as data points in training. The number of labeled collected real images is insufficient on its own, and it is only feasible to combine it with generated data. In particular, after augmenting, we have 312 images. These 312 images are added to 39,688. Though the labeled collected real data is scarce, including it with augmentations in training will introduce its distribution to the model, which should help make the model be more robust than when using only generated data.

3.4.2 Using Real Backgrounds

The creators of YOLOv5 recommend including background images in training to lower the number of false positives [ea21]. An image classifies as a background image when it does not have any objects to be detected. Some of the data provided by the military academy fits this description by not having any mission task symbols that need detecting (see Figure 17). However, these no-symbol images are very few, and we need more images like this.

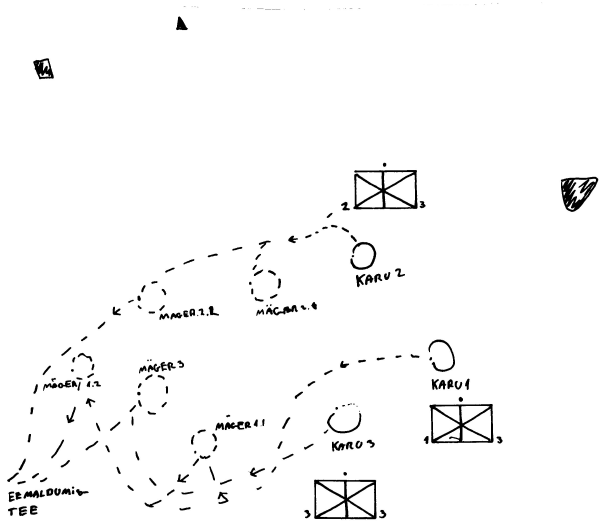


Figure 17. Background image without any mission task symbol. This military plan is drawn originally without any mission task symbols.

Additionally, we created a script to remove all mission task symbols from the military plans, i.e., we separate the signal from the background. Given the images and the labels, it removes the symbols by deleting everything within their bounding boxes (leaving white boxes) and only leaves the background.

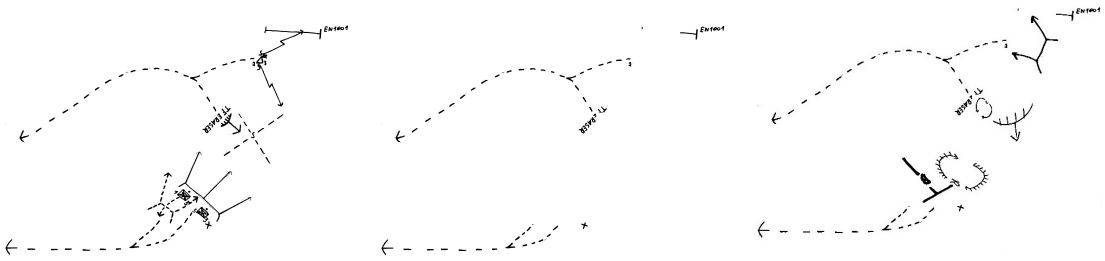
Using this script, we increased the number of background images. However, this still results only in a small number of backgrounds, and to make them have more impact on the training, we use the augmentations methods discussed in the Methods section. We flip the backgrounds horizontally, vertically, and both ways, resulting in four times as many data points.

In the implementation, the YOLOv5 code recognizes background images if they do not have corresponding label files. Therefore, if we add the backgrounds created to the training set without any associated label files, YOLO will recognize them as background images.

In this case, the training set consists of 39,688 images generated with the generator, plus 312 (real) background images with no labels.

3.4.3 Using Real Backgrounds in the Generator

Backgrounds can serve a purpose beyond their use in the training as no-object examples. We can generate real-looking images with a range of symbol placements by inserting randomly picked symbols into the original symbol locations on the background images (see Figure 18). Using these kinds of images would expose the model to a diverse range of imitated real data, which can improve its ability to distinguish the difference between symbols and background.



(a) Original Collected data.

(b) Background image created by removing the original symbols.

(c) Generated image by inserting different symbols into original symbol spaces.

Figure 18. Process of generating images with real backgrounds.

To accomplish this, we use the label files associated with these images, which contain bounding box information for the removed symbols. In place of the original symbols from the real images, a generated symbol with a random scale and a rotation is placed. However, the backgrounds may come in different dimensions, such as when they are cropped, resulting in varying resolutions compared to the other generated images. Although YOLO can resize the inputs, ensuring consistency in the data dimensions across the dataset is necessary. To preserve the integrity of the generated data, we

adjust the generated data with the backgrounds' resolution to match the generator's base resolution by either padding smaller images or scaling down larger ones. We implemented this method of using real backgrounds in the generator. The number of images the generator produces with real backgrounds can be modified with a parameter. For example, if we wish to generate 40,000 images and want 30% of it to be generated using real background images as the canvas, then the dataset generated would contain 12,000 images generated using real background images, and 28,000 images would be generated with normal canvases (white background and limited set of other elements like phase lines and location markers).

3.4.4 Using Real Symbols in the Generator

One limitation of the generator is that it lacks symbol variety. The symbols in real data have a huge variety that we cannot sufficiently imitate even when using dozens of sets of templates and augmentations. This variety comes from different drawing styles, markers used to draw the symbols, and noise that can appear in the real data (see Figure 19). Including symbols extracted from the real data in the generation process may help the models achieve higher robustness.

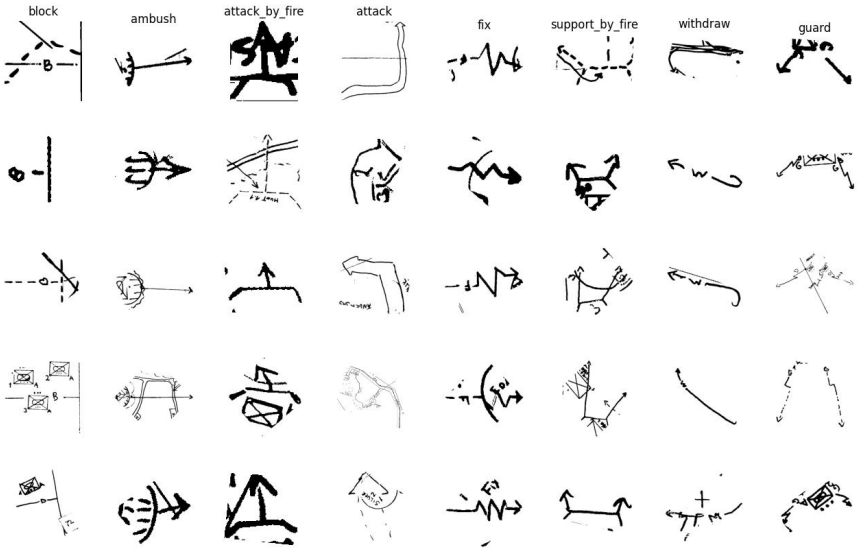
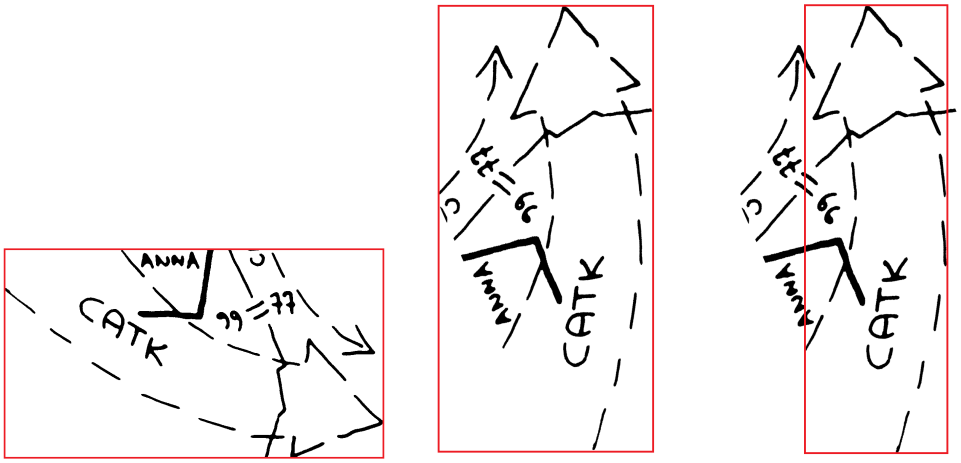


Figure 19. Symbols extracted from the real data. The symbol are noisy and their variety is high.

We can extract the symbols from the real data and use them in the generator to increase the symbol variety. However, these extracted symbols are not clean, meaning they have many background clutter included in the bounding box. We include the background clutter because they are a natural occurrence that is present in the real data

and we wish not to remove it. Additionally, we would want to rotate these symbols as well in order to get more variations of them. However, after rotating them, the bounding box of the symbol will change, and the background clutter may prevent the generator from calculating the correct bounding box (see Figure 20). The generator needs two sets of extracted symbols to fix this issue. The original one and one with all background clutter erased. Versions with erased clutter are used for calculating the correct bounding box after rotation, the found box is then taken from the original cluttered versions and placed on the images.



(a) Original bounding box of the extracted real symbol.

(b) The rotated version of the symbol bounding box calculated wrong because of the background clutter.

(c) Rotated version of the symbol with correct bounding box.

Figure 20. Extracted "ambush" symbol. The symbol is extracted from collected real data and bounding boxes of it are drawn with red rectangles.

To summarize, for including real symbols, we created a script to extract these symbols and implemented a way for the generator to include them in the data generation process. We also created cleaned versions of the extracted symbols as the implementation requires it for rotating these real symbols. Because of the scarcity of real symbols and their noisy nature, we may not want to use them always. The ratio of real symbols and author-created templates used by the generator can be modified via a parameter.

3.5 Evaluation

The metrics reported are very important for evaluating the performance of any solution. Metrics like mAP@0.5 and mAP@0.5-0.95 are used in object detection. The mAP is the

mean of AP for all classes, which is

$$mAP = 1/N \times \sum_{i=N}^{N} AP_i, \tag{4}$$

with AP_i being *i*th class and N is the total number of classes evaluated. mAP@0.5 means the Equation 4 was applied at 0.5 IoU threshold, while mAP@0.5-95 also averages mAP over the range of 0.5-0.95 IoU thresholds.

Selecting the right IoU threshold depends on the task. We believe detection of mission task symbols requires high precision in localization accuracy, therefore we selected 0.8 IoU threshold for the benchmark and report all metrics at 0.8 IoU threshold. This means that we report mAP@0.8 and mAP@0.8-0.95, which should give stricter results in localization accuracy.

However, due to the scarcity of data and class imbalance in the benchmark dataset, traditional mAP calculation methods may not accurately reflect the overall performance of the model. These traditional methods use macro averaging, which calculates the average precision (AP) individually for classes and then takes the mean to report mAP. This puts equal weight to APs of classes with 1 instance and 19 instances, resulting in the final number being not very representative of the performance we observe and would want to rank models by. Instead, with imbalanced data micro averaging should be preferred. Micro-averaged metrics are calculated by using statistics per instance instead of per class. For example, instead of calculating precisions for each class and averaging this to get overall precision for all classes, micro-averaging calculates precision by combining FP and FN counts for all classes. This approach provides a more comprehensive evaluation of the model's performance, considering the challenges of data scarcity and imbalance. Therefore, this study uses micro-averaging for all of the metrics reported. Micro-averaged statistics computation was implemented and added to our branch of YOLOv5 by the author.

Additionally, we introduce and report micro-averaged precision at 0.8 recall(P@R(80)) at 0.8 IoU threshold and micro-averaged recall at 0.8 precision(R@P(80)) at 0.8 IoU threshold. In principle, these metrics find the confidence threshold that yields the highest possible precision(recall) while also satisfying the condition 0.8<recall (precision). The code to find these values was implemented by the author. We include these metrics because they reflect the ideas "if we want high precision, what recall can we get?" and "if we want high recall, what precision can we get?", which we believe are more intuitive and informative than mAP for non-experts.

The F1 score is another performance metric for classification tasks. It is calculated by taking the harmonic mean of precision and recall. As it is an important performance metric for classification tasks, we also report the max F1 score that can be obtained at IoU 0.8 threshold.

To summarize, we are reporting the following micro-averaged (instance-level) metrics:

mAP@0.8, mAP@0.8-95, P@R(80), R@P(80), and max F1 score. We believe that these metrics enable us to describe the performance of different models thoroughly and in a manner useful for the end-users.

3.6 Experiments

During training, YOLOv5 applies several augmentations to the input images, including random flipping, brightness adjustments through changes in hue, saturation, and value channels, scaling, translation, and creating mosaics. Mosaic augmentation involves stitching together four cropped images to create a new training image. These augmentations are enabled by default, as the creators of YOLOv5 recommend them to improve model learning [ea21]. Although some of these augmentations can be helpful for the model to learn better, some are not necessary because of the nature of the data we are using. Specifically, since the data we are using will always be binarized and will not have varying brightness levels, hue, saturation, and value channel augmentations have been disabled. All other augmentations remain enabled during training. The main focus of this thesis is to improve the performance of symbol detection via the use of real data. To fairly compare data usage schemes, the experiment setup and hyperparameters of the training are fixed. To fix them at useful values, some preliminary experimentation is done. Different network sizes, image sizes, weights (pre-trained and randomly initialized weights), and mosaic settings (enabled and disabled) are tried to get the best experiment setup and hyperparameters.

The final experiments were conducted using the large YOLOv5 model architecture with pre-trained weights of yolov5l6.pt[4] and an image size of 1280 by 1280 pixels. 40,000 images with approximately 6,000 symbol instances per class were used as the primary data source. For validation and early stopping, 10,000 generated images with real backgrounds were used. 50% of the symbols in the validation set were symbols extracted from the collected real data, and the rest was template symbols. Notice that for all data usage schemes, the same validation set was used in order for the performance differences not to emerge from the early stopping. Early stopping with the patience of 25 epochs and a 0.0001 improvement threshold was applied. YOLOv5 default augmentations were used for the training set, including translate with 10% probability, scale with 50% probability, horizontal flipping with 50% probability, and mosaic with 100% probability. For the validation set, no augmentations were applied by YOLOv5.

As one of the main contributions of the thesis, the following different ways of including real data in training were compared:

1. **Baseline** - No real data was included in generating the data and only generated data was used. 40,000 images without any inclusion of real data were used.

[4]Pre-trained weights downloaded from YOLOv5 GitHub repository. https://github.com/ultralytics/yolov5 Accessed on 25.04.2023

2. **Generated Data + Real Data** - Real images augmented with horizontal, vertical and both vertical and horizontal flipping were used alongside the generated data. As a result, 39,688 images without any inclusion of real data and 312 real images were used.

3. **Generated Data + Empty Backgrounds** - Real background images extracted from the real data, augmented with horizontal, vertical and both vertical and horizontal flipping, were used additionally to the generated data. As a result, 39,688 images without any inclusion of real data and 312 real background images were used.

4. **Generated Symbols on Real backgrounds** - Generated data had template symbols placed in the original symbols positions of real backgrounds created from the collected data, augmented with horizontal, vertical and both vertical and horizontal flipping. For the model reported in the Results section, 30% of the data (12,000 images) were generated using real backgrounds, and the rest (28,000 images) used the default generation process with generated canvases with a limited set of background items.

5. **Real Symbols in Generated Data** - Real symbols extracted from the real data were used in the generator. No real backgrounds were used. The ratio indicates the percentage of real symbols that appear in generated data. Different ratios were initially tested, but only the most promising ratio was finally re-trained and is reported in the Results. For the reported model, 50% of the symbols generated were real symbols from real data.

6. **Real Symbols + Real Backgrounds in Generated Data** - Both real symbols extracted from the collected dataset and real backgrounds created from the collected data, augmented with flipping and with symbols placed in the original symbol positions, were used in the generator, but separately. This means no real symbols were generated in the real backgrounds. 30% of the generated data (12,000 images) is generated using real backgrounds and template symbols, for the other 28,000 images, the default generation process with limited background elements was used. For those images, 50% of the symbols generated were symbols extracted from the real data. In total, 40,000 images were used.

7. **Real Symbols in Real Backgrounds in Generated Data** - Both real symbols extracted from the collected dataset and real background images extracted from the collected data, augmented with flipping and with symbols placed in the original symbol positions, were used in the generator and might appear together. This means real symbols can appear in the real background images. For this data type, two different experiments are done;

- Real Symbols in Real Backgrounds in Generated Data with 30% real backgrounds usage with 50% real symbols usage. In total 12,000 images were generated using real backgrounds, and 28,000 using white canvases. 50% of the symbols generated were from real data in both cases.
- Real Symbols in Real Backgrounds in Generated Data with 20% real backgrounds usage with 25% real symbol usage. In total 8,000 images were generated using real backgrounds, and 32,000 used generated canvases. 25% of the symbols generated were from real data.

Only the best result is reported for every data usage scheme among the above list of 7 approaches.

3.7 Tools

All the scripts and tools the author of this thesis created are in Python 3.10 programming language. The images of the plastic films were acquired using the Xiaomi Mi 11 Lite phone camera and an A3 size lightboard[5]. The acquired images were partially labeled using Labelimg [Tzu15]. Additionally, the language model ChatGPT and Grammarly were used to enhance the readability and style of the thesis. It should be noted that ChatGPT was solely used for style and grammar correction and not for content creation.

[5]Used lightboard can be found at this site: https://shop.kl24.ee/bulb-pirn-lamppu-LED/0/readmore/27489 Accessed on 04.05.2023

4 Results

The thesis comprises three parts: collecting a dataset, presenting a benchmark, and using the collected dataset to enhance object detection models' performance in detecting NATO mission task symbols. The collected dataset includes 113 digitized and labeled images, encompassing 23 symbol classes and 789 symbols. The sorting and digitization of images took 5 hours, and 40 hours were spent labeling the data at an average of 20 minutes per image and 2 minutes per symbol. The images were divided into training and benchmark datasets, with 20 images and 297 symbols excluded as they were deemed low quality or unrepresentative by the military academy. In this section, we present the benchmark dataset, the selection of performance metrics for the benchmark, and the results of experiments using different approaches to incorporate real data in improving the model performance.

4.1 Collected Dataset

The collected data set was divided into training and benchmark sets. The former contains originally 78 images, but is further processed into 78 natural background images (all symbols removed) and 395 symbols from 17 different symbol classes. The benchmark set includes 15 images and 97 symbols within 18 different symbol classes. Individual symbol counts are shown in Figure 21.

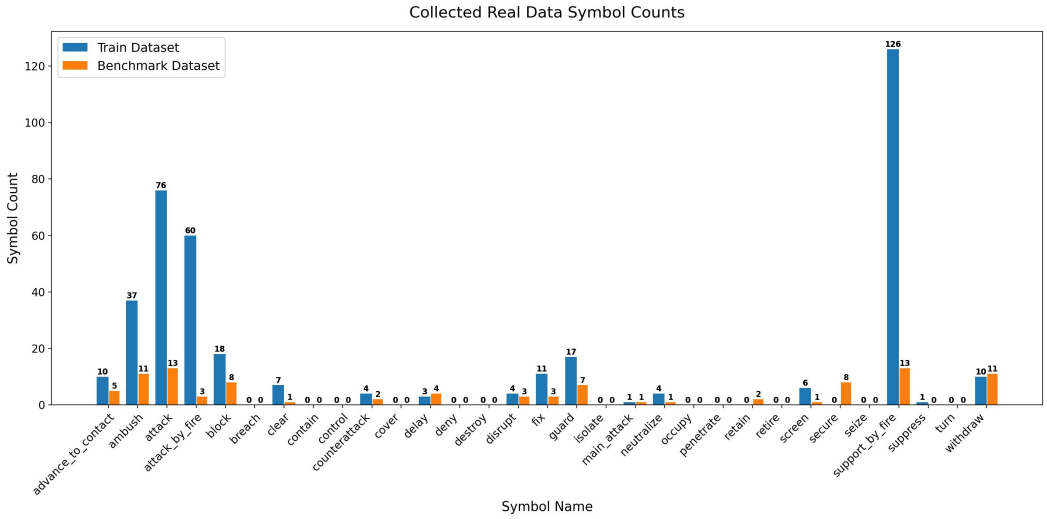


Figure 21. Symbol distribution of the collected real data.

Figure 21 shows that the collected datasets are very imbalanced. Because of the imbalance, we chose to report micro-averaged metrics for the benchmark evaluation.

4.2 Defining a Benchmark

We define a benchmark based on the 15 labeled images and 97 symbols. We are aware this data set is clearly too small to capture the ability of solutions to cope with all the diversity contained in the real data. However, the number of labeled images can easily be expanded, whenever more military plans become available. The size of the benchmark dataset is small due to external factors (unavailability of the raw data), not the willingness of the author to label the data.

Any model evaluated on this benchmark must make predictions and return identity, location and confidence in the format YOLOv5 does, based on which our code can compute the following metrics:

- P@R(80)
- R@P(80)
- mAP@0.8
- mAP@0.8-0.95
- maxF1

All of the metrics are calculated using micro-averaged statistics and at IoU 0.8 threshold, which means the benchmark assumes failing in rarely encountered symbol types is acceptable if over all instances of all classes the performance is good. A model can be better than another model according to one, many, or all metrics. Different metrics capture slightly different aspects of usefulness and aim to form a complete and intuitive picture of the performance of the models compared.

Experiment	P@R(80)	R@P(80)	mAP@0.8	mAP@0.8-0.95	maxF1
Experiment 1	0	0.66	0.699	0.644	0.746
Experiment 2	0	0.64	0.727	0.683	0.73

Table 1. Example table. The experiments and results are placeholder values.

From looking at Table 1 we can say that Experiment 1 has slightly better results in P@R(80) and maxF1 but it has lower scores in mAP@0.8 and mAP@0.8-95 than Experiment 2. Our benchmark does not say which model is better and in this case, the user should decide if more recall or localization precision is preferred or required.

4.3 Results of Experiments

Seven different experiments were conducted to analyze the effects of different ways of using real data in training. One experiment did not include any real data during its training and was used as a baseline model. The rest of the experiments incorporated the methods in Section 4.3 of different ways to include real data. The experiment "Real Symbols in Real Backgrounds in Generated Data" presented in this section had 30% of its data consisting of real backgrounds and 50% real symbols. Finally, the experiments are evaluated on the proposed benchmark dataset with proposed metrics of P@R(80), R@P(80), mAP@0.8, mAP@0.8-0.9, and max F1 score. All of the metrics reported used the IoU threshold of 0.8 except mAP@0.8-0.95. Due to the imbalance of the benchmark dataset, all the reported metrics are micro-averaged, meaning they are calculated using instance-based statistics and not class-based.

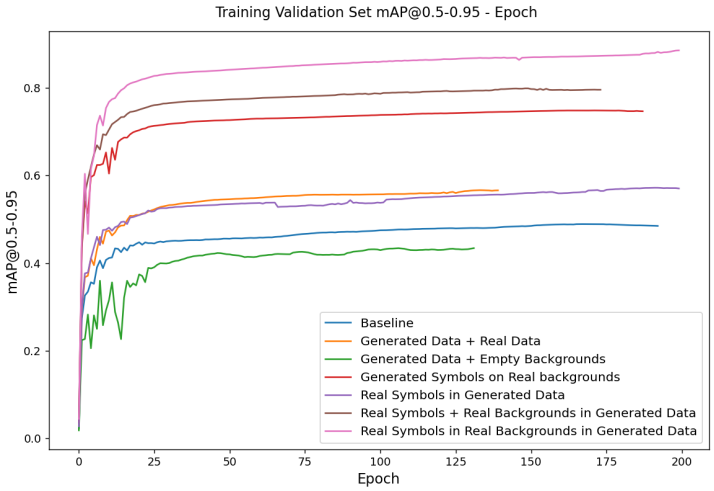


Figure 22. Plot of validation mAP@0.5-0.95 on training for experiments.

During the training by default, YOLOv5 calculates macro averaged mAP@0.5-0.95 on the validation set after every epoch (notice it is 0.5 and not 0.8). The graph of validation mAP@0.5-0.95 - epoch in Figure 22 gives a good feel for the performance of different experiments. Analyzing macro averaged statistics was an appropriate approach in this instance, given the balanced distribution of symbols across all classes in the validation set. The experiment "Real Symbols In Real Backgrounds" seemed to give more promising results on the validation set compared to other experiments. However, notice that the validation set consists of generated data, not real data. Secondly, notice that our benchmark assumes higher spatial precision is needed and does not report mAP@0.5-0.95. The test results (benchmark results) are presented in Table 2 and are discussed below.

Table 2 demonstrate that all of the methods described in Sections 4.3 and 4.5 per-

Experiment	P@R(80)	R@P(80)	mAP@0.8	mAP@0.8-0.95	maxF1
Baseline	0	0.125	0.301	0.265	0.403
Generated Data + Empty Backgrounds	0	0.186	0.417	0.344	0.398
Generated Symbols on Real backgrounds	0	0.309	0.47	0.42	0.564
Real Symbols in Generated Data	0	0.495	0.565	0.504	0.635
Generated Data + Real Data	0	**0.66**	0.699	0.644	0.746
Real Symbols + Real Backgrounds in Generated Data	0	0.649	0.702	0.661	0.718
Real Symbols in Real Backgrounds in Generated Data	0	**0.66**	**0.727**	**0.683**	**0.75**

Table 2. Experiment results on benchmark set. The best scores on each metric are highlighted in bold.

formed better than the "Baseline" experiment, which lacked real data in training. Notably, all experiments had a P@R(80) metric of 0, indicating that achieving 0.8 recall was impossible with all models, no matter how lenient the confidence threshold would be. This means there are a significant amount of symbols in the benchmark that are very hard to correctly detect or localize. Upon comparing the rest of the metrics, the "Real Symbols in Real Backgrounds in Generated Data" experiment stands out with the best results across all metrics. It is closely followed by the "Generated Data + Real Data" experiment, which has equal performance on R@P(80) but lower numbers in other metrics.

Table 2 also shows us that using real backgrounds as no-object examples led to improved scores on R@P(80), mAP@0.8, and mAP@0.8-0.95, in comparison to the "Baseline" approach. This is logical if we assume that using background images lowers the false positive rate, boosting precision. An increase in R@P(80) suggests that at lower confidence thresholds, we are still able to maintain high (>0.8) precision, meaning the number of false positives remains low. As we lower the confidence threshold, the model becomes more inclusive, increasing recall as more relevant predictions(true positives) are captured. Additionally, an increase in mAP@0.8 indicates lower false positives and higher relevant predictions overall, while an increase in mAP@0.8-0.95 suggests an increase in the spatial precision of the detections.

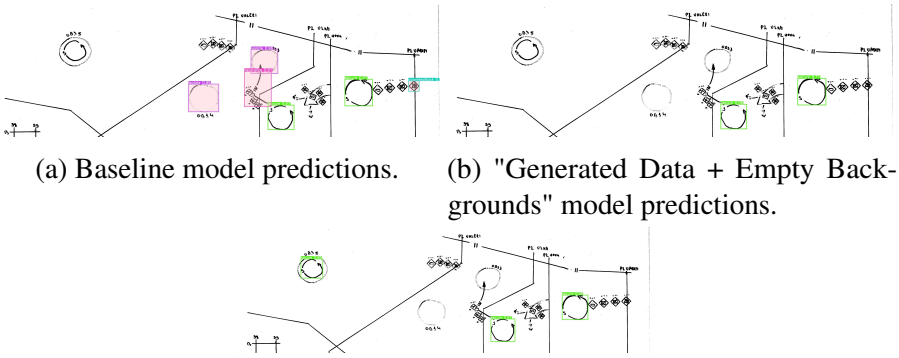
Furthermore, generating template symbols on real backgrounds improved all metrics

scores compared to the simpler approaches, as seen in the "Generated Symbols on Real backgrounds" experiment results. The best performance was achieved by introducing real symbols extracted from the collected dataset to the mix, as demonstrated by the results of the "Real Symbols in Real Backgrounds in Generated Data" experiment. This trend of improvement is logical because the models are iteratively introduced to different parts of the real data, culminating in the best results when combined. However, this result was not trivial, introducing too much complexity via mixing everything could have hindered the training by making the task impossible or too difficult to solve. We could not be sure the most complex data usage scheme is the most optimal before experimenting.

4.4 Visually Comparing Model Errors

While metric scores provide insights into the performance of different experiments, they do not reveal the types of errors made by each model. In order to gain a better understanding of these errors, we must examine the results in more detail. In this section, we will provide examples of the mistakes that were removed through the use of more advanced data usage techniques. We will also identify the types of errors that still remain even with the most capable models. By identifying these failure cases, we can emphasize addressing them in the future.

When we look at detections visually, we can see that introducing real backgrounds as no object examples makes the model less confident in the predictions and conditions it to predict less overall. However, because of this, it also makes fewer mistakes meaning fewer false positives. This trend continues with the experiment “Generated Symbols on Real Backgrounds”. In Figure 23 we can see an example where the use of real backgrounds decreases the background false positive count.



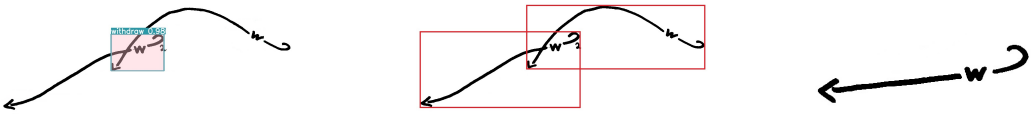
(a) Baseline model predictions.

(b) "Generated Data + Empty Backgrounds" model predictions.

(c) "Generated Symbols on Real Backgrounds" model predictions.

Figure 23. Different model predictions on a benchmark image with false positives bounding boxes filled with transparent pink color.

Some of the false and missed detections in the experiment "Real Symbols in Real Backgrounds in Generated Data" can be explained by looking at them visually. In Figure 24a we can see that the model confuses the crossing point of two withdraw symbols for a withdraw symbol that does not exist. Figure 24b shows these withdraw symbols' ground truth bounding boxes. It is important to note that these withdraw symbols are drawn in irregular shapes, and the model only trained on more regularly shaped withdraw symbols which can be seen in Figure 24c.



(a) "Real Symbols in Real Backgrounds in Generated Data" experiment false positive detection bounding box of withdraw symbol marked with pink overlay by form the benchmark dataset.

(b) Ground truth bounding boxes of withdraw symbols drawn with red.

(c) A regularly shaped withdraw symbol.

Figure 24. Irregular "withdraw" symbols. False and missed detections of the experiment "Real Symbols in Real Backgrounds in Generated Data" compared with the ground truth bounding boxes and a regular symbol of the same type.

Additionally, the best-performing experiment, "Real Symbols in Real Backgrounds in Generated Data", still fails to detect symbols when the background elements are very dense near a symbol. Figure 25 shows an example of a densely filled area. The model misses all of the symbols shown in this figure (false negatives).

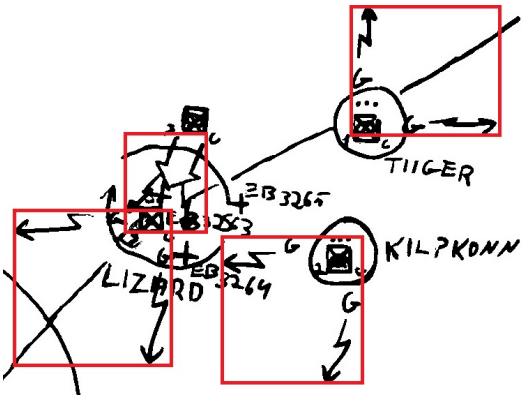
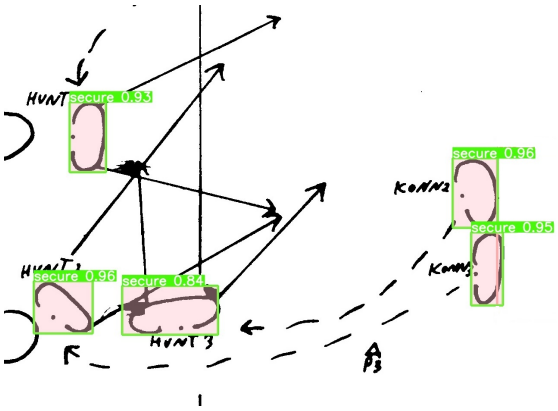
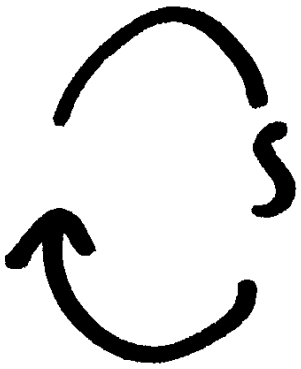


Figure 25. A dense area of mission task symbols with many background elements from the benchmark dataset. Mission task symbols are marked with red boxes.

One common false detection that most trained models (i.e., all models except “Generated Data + Real Data” and “Generated Data + Empty Backgrounds”) make is confusing the circular background symbols seen in Figure 26a with mission task symbol “secure”. The shape of these background symbols resembles the shape of the mission task symbol “secure” closely (see Figure 26b).



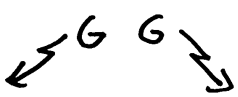
(a) False detection of background symbols overlaid with transparent pink color.



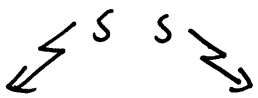
(b) "Hand-drawn mission task symbol "secure" template.

Figure 26. "Secure" symbol false positive detection. False positive detection of "secure" by the model "Real Backgrounds in Generated Data" on the benchmark dataset and a hand-drawn "secure" symbol is shown.

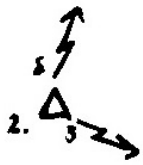
Another common mistake all models make is confusing the mission task symbols “guard” and “screen” on the benchmark. These symbols resemble each other significantly, with the only difference being the letter that the symbol contains. The “Guard” symbol has a lowercase or uppercase “G” in it, and the “screen” symbol has a “S” instead. However, because of the small size of the symbol on the image, in some instances, “S” can resemble a lowercase “G”. Figure 27c shows an example of a “screen” symbol detected as a “guard” by all models.



(a) Guard Symbol.



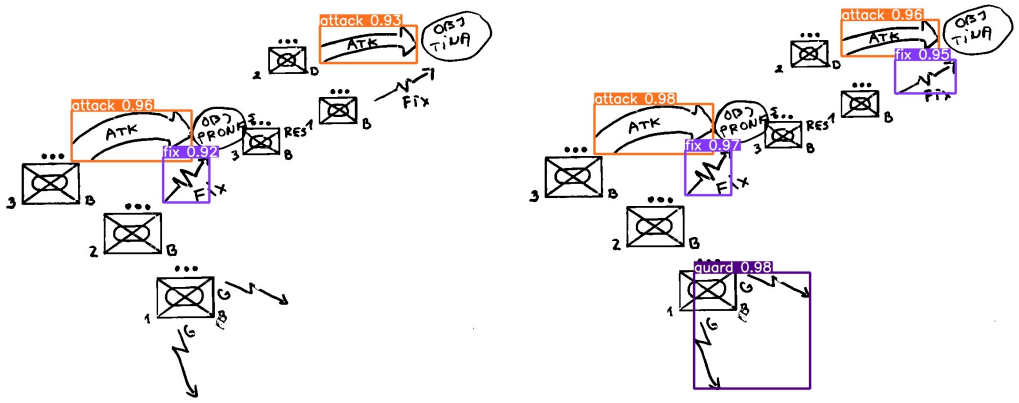
(b) Screen Symbol.



(c) A screen symbol from the benchmark.

Figure 27. Misclassified screen symbol. Examples of guard and screen symbols and an example of a screen symbol from the benchmark which is confused as a guard symbol.

Finally, when comparing the two best-scoring models, "Generated Data + Real Data" and "Real Symbols in Real Backgrounds in Generated Data", we observe that the former achieves high scores on all metrics due to not falsely classifying the "secure" symbols(see Figure 26a), whereas the latter does falsely classify it. However, the "Real Symbols in Real Backgrounds in Generated Data" experiment detected more symbols correctly in most cases. For example, in Figure 28, "Generated Data + Real Data" missed some symbols, while "Real Symbols in Real Backgrounds in Generated Data" detected all symbols successfully.



(a) "Generated Data + Real Data" model predictions.

(b) "Real Symbols in Real Backgrounds in Generated Data" model predictions.

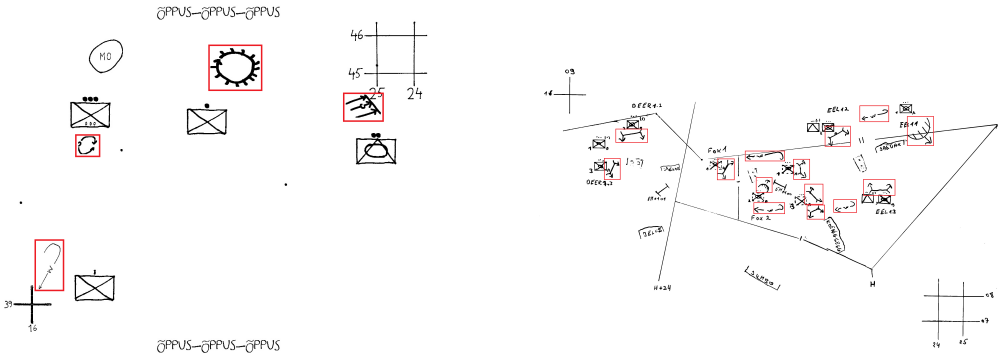
Figure 28. Comparison of detections between the two best models. The true positive predictions of models the "Generated Data + Real Data" and "Real Symbols in Real Backgrounds in Generated Data" are shown.

4.5 Discussion

The methods described in Section 4.3 affect the performance of symbol detection in various ways. Many factors may cause these effects. First, the vanilla generator-generated images differed from the real data in their overall layout and background. Secondly, even though we used a variety of templates, it turned out that real symbols were from a much wider distribution with messy backgrounds and handwriting, so we concluded that imitating real data requires using real data. With the methods in sections 4.3.3 and 4.3.4, we introduced ways to include real data in the generator to address the limitations of the initial generator implementation.

In section 4.3.3, we introduced a way to use backgrounds from real data as a canvas for the generator to generate symbols. This method addresses the limitation of the image canvas being too empty and unrealistic compared to the real data (a comparison

between generated data on an empty canvas and real data can be seen in Figure 29). The results showed that generating data using real backgrounds enabled the model to learn to distinguish symbols from backgrounds and produce fewer false positives.



(a) Example of generated data on an empty canvas.

(b) Example of real data from the training set.

Figure 29. Comparison between generated data and real data. (Symbols are marked in red boxes.) Generated data is generated on an empty canvas, and real data is created to have a military sense.

In section 4.3.4, we introduced a way to use symbols extracted from the real data in the generator. This method addresses the limited symbol variability within used templates compared to the real data. Real symbols are highly variable, with irregular shapes and high levels of noise. By introducing real symbols to the models, we observe that R@P(80) metric increased in all cases compared to approaches where the model never saw any real symbols. This indicates that using real symbols in the generator helps to overcome the symbol variability limitation and leads to better generalization.

These methods are designed to introduce real data distribution to the training, and using them separately in the generator already achieves this, but by combining them, we got more diversity in the training, which in turn improved the performance of the trained models.

Real data includes real symbols and real backgrounds, even though with a limited number of images and variability, which means it is one of the most difficult data types. However, it was still surprising that “Real Data + Generated Data” performed almost as well as "Real Symbols in Real Backgrounds in Generated Data". There could be several explanations for this performance. For example, "Real Symbols in Real Backgrounds in Generated Data" may be overfitting to the real backgrounds, as a unique background is used multiple times. In contrast, "Real Data + Generated Data" is exposed to each unique real background only once. Another possible explanation is that "Real Data + Generated Data" avoiding false positive detections of the "secure" symbol, as discussed in the previous section, either by luck or by not overfitting to the real symbols that

"Real Symbols in Real Backgrounds in Generated Data" is using. Finally, for the "Real Symbols in Real Backgrounds in Generated Data" model, the training and validation sets are generated using the same real backgrounds, which may cause the model to overfit not only to the training set, but also the validation set. As the model overfits to these backgrounds, the validation metrics keep improving and early stopping does not trigger. However, on the unseen benchmark set this leads to weaker generalization.

Another curious result is the P@R(80) scores of the experiments. None of the experiments achieved a recall greater than 0.8 at however low confidence threshold value, resulting in a score of 0 for this metric. While this surely reveals the weakness of our current models, it does not make tracking this metric useless in the long run when models gradually improve. The set of metrics in the benchmark was chosen to align with the requirements of our collaboration partner and remains relevant.

In summary, we see using real data in a trivial way helps to improve performance. If sufficient data exists, it will be possible to get the desired performance. However, in our case, using real backgrounds and real symbols in the generator still gives a benefit because of the scarcity of the data.

5 Conclusion and Future Work

The creators of YOLOv5 recommend that the training data for the model should represent the deployment environment. By applying the techniques described in this thesis, we aimed to train the modes on images more similar to what the military actually draws and hence improve the symbol detection on real data. The results of the experiments show that the proposed methods achieve this goal to some extent. Especially extracting the backgrounds and symbols from the collected dataset and using these to generate data with more variations in rotations and placements is a very promising method.

Although the methods used gave promising results, there are still limitations in using real data in the generator. The usage of real backgrounds in the generator is limited to only inserting symbols to the original places of the removed symbols. The implementation can be expanded by defining rules to find other spaces to place additional symbols or placing them randomly in any place on real backgrounds. This may create a challenging dataset and improve the performance of symbol detection.

395 symbols and 78 images from the collected dataset were used in training. This dataset can be increased when more data is made available. The model will be exposed more to real-life distribution, with more varieties of real data, which should improve its performance. Additionally, the proposed benchmark currently has 15 images with 97 symbols. To evaluate the symbol detection models more accurately and publish the benchmark, the dataset size should be improved in the future.

More in-depth analysis can be made to understand the generalization degree of models when using real data in the generator. With this analysis, real data usage can be optimized to give better results.

In summary, the thesis goal of collecting and using a labeled dataset of NATO mission task symbols to improve symbol detection models and proposing a benchmark for it is achieved. When real data availability is limited, symbol detection performance can be improved by using a relatively small real dataset in different ways. While our current results may not have fully met the desired requirements to be successfully deployed, we view this as an opportunity for future work to further improve the performance of our solutions as data size and quality improve.

6 Acknowledgments

I would like to acknowledge and thank everyone who helped me during my time as a student at the University of Tartu.

I would like to specifically thank Mihkel Lepson and Karl-Kristjan Kõverik, who were members of the ICS team I was a part of. They were very understanding and helpful during my one year on the team.

I would like to sincerely thank my supervisor and team lead, Dr. Ardi Tampuu, for his exceptional support and dedication throughout my academic journey and the completion of this thesis. His invaluable guidance and feedback have been instrumental in my studies.

References

[AFG20] Yali Amit, Pedro Felzenszwalb, and Ross Girshick. Object detection. *Computer Vision: A Reference Guide*, pages 1–9, 2020.

[app11] APP-6(c). NATO joint military symbology. 2011.

[COR+16] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3213–3223, 2016.

[DDS+09] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[ea21] Glenn Jocher et. al. ultralytics/yolov5. `https://github.com/ultralytics/yolov5`, October 2021.

[EJAG20] Eyad Elyan, Laura Jamieson, and Adamu Ali-Gombe. Deep learning for symbols detection and classification in engineering drawings. *Neural networks*, 129:91–102, 2020.

[EVGW+09] Mark Everingham, Luc Van Gool, Christopher KI Williams, John Winn, and Andrew Zisserman. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88:303–308, 2009.

[GBC16] Arna Ghosh, Biswarup Bhattacharya, and Somnath Basu Roy Chowdhury. Sad-gan: Synthetic autonomous driving using generative adversarial networks. *arXiv preprint arXiv:1611.08788*, 2016.

[GDDM14] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[GPAM+14] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in Neural Information Processing Systems*, volume 27, pages 2672–2680. Curran Associates, Inc., 2014.

[JSA+20] Mamta Juneja, Shaswat Singh, Naman Agarwal, Shivank Bali, Shubham Gupta, Niharika Thakur, and Prashant Jindal. Automated detection of glaucoma using deep learning convolution network (g-net). *Multimedia Tools and Applications*, 79:15531–15553, 2020.

[KRZ+19] Jacob Kahn, Morgane Rivière, Weiyi Zheng, Evgeny Kharitonov, Qiantong Xu, Pierre-Emmanuel Mazaré, Julien Karadayi, Vitaliy Liptchinsky, Ronan Collobert, Christian Fuegen, Tatiana Likhomanenko, Gabriel Synnaeve, Armand Joulin, Abdelrahman Mohamed, and Emmanuel Dupoux. Libri-light: A benchmark for ASR with limited or no supervision. *CoRR*, abs/1912.07875, 2019.

[LHY21] Shangbang Long, Xin He, and Cong Yao. Scene text detection and recognition: The deep learning era. 129(1):161–184, 2021.

[LMB+14] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common objects in context. In David Fleet, Tomas Pajdla, Bernt Schiele, and Tinne Tuytelaars, editors, *Computer Vision – ECCV 2014*, Lecture Notes in Computer Science, pages 740–755. Springer International Publishing, 2014.

[Ots79] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[PHSS22] Pablo N Pizarro, Nancy Hitschfeld, Ivan Sipiran, and Jose M Saavedra. Automatic floor plan analysis and recognition. *Automation in Construction*, 140:104348, 2022.

[PNDS20] Rafael Padilla, Sergio L Netto, and Eduardo AB Da Silva. A survey on performance metrics for object-detection algorithms. In *2020 international conference on systems, signals and image processing (IWSSIP)*, pages 237–242. IEEE, 2020.

[PSP+19] Raymond Ptucha, Felipe Petroski Such, Suhas Pillai, Frank Brockler, Vatsala Singh, and Paul Hutkowski. Intelligent character recognition using fully convolutional neural networks. *Pattern recognition*, 88:604–613, 2019.

[RDGF15] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640, 2015.

[RF16] Joseph Redmon and Ali Farhadi. YOLO9000: better, faster, stronger. *CoRR*, abs/1612.08242, 2016.

[SK19] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019.

[TSS+22] Vajira Thambawita, Pegah Salehi, Sajad Amouei Sheshkal, Steven A Hicks, Hugo L Hammer, Sravanthi Parasa, Thomas de Lange, Pål Halvorsen, and Michael A Riegler. Singan-seg: Synthetic training data generation for medical image segmentation. *PloS one*, 17(5):e0267976, 2022.

[Tzu15] Tzutalin. Labelimg. git code. `https://github.com/heartexlabs/labelImg`, 2015.

[ZCS+23] Zhengxia Zou, Keyan Chen, Zhenwei Shi, Yuhong Guo, and Jieping Ye. Object detection in 20 years: A survey. *Proceedings of the IEEE*, 2023.

[ZM18] Zahra Ziran and Simone Marinai. Object detection in floor plan images. In *Artificial Neural Networks in Pattern Recognition: 8th IAPR TC3 Workshop, ANNPR 2018, Siena, Italy, September 19–21, 2018, Proceedings 8*, pages 383–394. Springer, 2018.

Appendix

I. Code Repository

The implementation of the methods described and trained object detection models are available at https://github.com/aralacikalin/NatoSymbols. This GitHub repository contains the combined work of the ICS team. The modified scripts for evaluating YOLOv5 models are in the folder “yolov5”. The tool created for labeling rotations of the symbols is in the folder “RotationLabeler”. The generator is in the folder “generator”. The collected real data is not publicly available.

II. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Aral Açıkalın**,
*(*author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Collecting and Using a Labeled Dataset of NATO Mission Task Symbols to Improve and Benchmark Detection Models,
   *(*title of thesis)

   supervised by Ardi Tampuu.
   *(*supervisor's name)

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Aral Açıkalın
09/05/2023