Bachelor Informatica

University of Amsterdam

Exploring the Effectiveness of Object Detection Training in Virtual Environments

Tobie Werner

June 23, 2023

Supervisor(s): Arnoud Visser

Signed: Signees

Abstract

Large and diverse datasets are required to train accurate object detection models. To create these datasets, specialized annotation software is used to specify where objects are present in an image. Object locations are defined by bounding boxes, which are then exported in a format that can be used in model training. A lot of human labour is required to construct these datasets. Alternatively, synthetic datasets can be generated in 3D virtual environments. No human annotators are needed in this case, as computers can calculate ground truth bounding boxes for all objects present in the environment.

In this thesis, we explore how training object detection models using synthetic datasets can contribute to the performance of an object detection model. In particular, we focus on object detection tasks relevant to the RoboCup Standard Platform League. Namely, detecting NAO robots and a small football. We do so by generating synthetic datasets from a 3D virtual RoboCup environment using the Unity Perception package.

From our experiments, we found that synthetic data can be used to increase a models performance when used in combination with real-world images. However, using only synthetic images as training data will result in poor model performance compared to comparable amounts of real-world images.

We conclude that synthetic datasets can be used to increase the accuracy of object detection and could therefore be a valuable addition in preparing NAO robot systems for the RoboCup Standard Platform League.

Contents

CHAPTER 1

Introduction

Because of the rapid acceleration of developments in the field of artificial intelligence, computers are now able to perform human-like tasks better than ever before [16]. Tasks and activities that were once too complex for robots to perform, such as delivering packages, working at construction sites and playing sports are becoming increasingly achievable for robots [25]. The problem of making robots interact with their surroundings in the way humans do is multifaceted. To do so, computers need to have the capability of observing the world around them, reason about the circumstances and take action [16]. Developing each of these requirements is very challenging.

1.1 Object Detection

In order for robots to interact with the world around them, it is important for them to observe their environment [1]. A robot has to know what type of object is in its field of vision before it can decide what to do with that object. Human beings can easily recognise objects, but it is quite difficult for computers to do it accurately [24]. Also, computers have to be able to recognise an object in an image under different conditions. For example, a computer should be able to determine that an object is a coffee cup if it fell on its side, is occluded by another object or not lit very well. It is also important for a computer to know where objects are located in an image.

Detecting and localizing objects in images or frames of a video are much researched tasks in the field of computer vision. Different machine learning algorithms have been developed over the years. One family of machine learning models, which we further describe in Chapter 2.1, is the "You Only Look Once" (YOLO) line of models. The YOLO model, when trained on a dataset that is sufficiently large and diverse, can determine bounding boxes around multiple objects in an image and determine what class each object belongs to using a single network pass [13]. The YOLO models are fast enough to be used in real-time object detection applications [20]. In this project we use the eighth version of this family of models. In particular, we use the smallest "nano" variant of the model.

Large datasets are required to train accurate object detection models [6]. In order to create these datasets, images need to be labeled. The labeling of these images requires a substantial amount of human labour [10]. Instead of using real-world images for object detection model training, researchers have looked at using synthetic data.

1.2 Synthetic Datasets

3D virtual environments can be used to generate images [2]. Synthetic datasets can then be constructed with these images. The objects in these pictures have corresponding ground truth bounding boxes. Since it is a 3D environment, the computer is able to compute the exact location and dimension of each bounding box, as well as the class that the object inside of it belongs to [2]. This means that the ground truth bounding boxes precisely contain the objects inside of them, without including too much of the background. The images and their ground

truth bounding boxes can be exported into an appropriate format for object detection model training. Frameworks exist to generate synthetic datasets.

The Perception package is the framework that we use in this project. It is a package that can be imported into the Unity game engine. The package adds functionalities to the existing Unity environment that can be used to generate large and diverse synthetic datasets [3]. An example of a synthetic training example is visible in Figure 1.1. Different kinds of parameters can be set and randomized through an interface to create synthetic images fit for object detection systems. This thesis goes over the details of the randomization parameters that we use in this project in chapter 2.3.1.

There are several benefits to generating synthetic datasets and using them as training data for object detection algorithms as opposed to using real-world datasets [3]. Firstly, real-world datasets require human labour to create [10]. Therefore, the production of large datasets can be expensive [11] [21]. This is especially the case for small companies [21]. Since human beings are prone to errors, especially when working for long hours, using computer generated datasets becomes an especially attractive alternative. This is because synthetic datasets do not need to be audited [3].

It is also important that a dataset captures a wide range of situations [3]. It is important for such diverse datasets to contain labeled images that accurately represent the real world. This means that rare cases need to be included in the dataset. For example, a self driving car should be able to recognize a cat that crosses the road or a tree that fell down. Other cases include blur and overexposure of the camera lens. Additional work is required to ensure that these cases are present in the dataset. However, with synthetic datasets, you can make sure that even rare situations are included in the dataset with a certain probability.

Besides these difficulties, privacy is also an important part of data collection [3]. Potentially harming data with regard to personal privacy must be filtered from the training set. This is not as much of a problem when generating synthetic datasets. If it is established that 3D environments and models within them do not violate anyone's privacy, resulting datasets do not either.

We can mitigate these problems by generating synthetic datasets [3]. What remains to be seen is if synthetic datasets can actually benefit the performance of object detection models, which is what we investigate in this thesis. In particular, we focus on object detection tasks relevant to the RoboCup Standard Platform League, described in the next section.

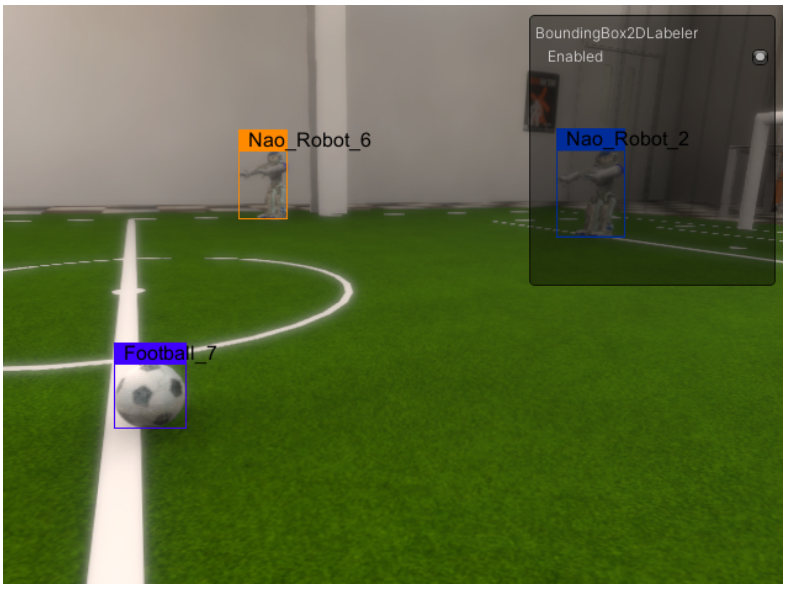


Figure 1.1: A screenshot of the Unity camera window. The bounding boxes and classes are determined and visualized using the Perception package.

1.3 RoboCup Standard Platform League

The focus of this project is on object detection tasks relevant to the RoboCup Standard Platform League. The RoboCup Standard Platform League is a championship in which autonomous robots play football matches against each other. RoboCup football events are organised worldwide on a yearly basis. A photo of such an event is shown in Figure 1.2. The robots that compete in the league are of the NAO variety. Systems and functionalities used by the NAO robots are researched and developed by teams all over the world. The purpose of the league is to promote research into artificial intelligence and autonomous robots[1].

Many different algorithms are needed in order to allow robots to play a game of football. The object detection tasks that we focus on in this research are detecting NAO robots and a small football.

Figure 1.2: A photo taken at the 2019 RoboCup Standard Platform League event in Sydney[2].

1.4 Research Question

The goal of this project is to contribute to a better understanding of model performance when trained, either fully or partially, using synthetic datasets. In particular, the aim of this project is to investigate the performance of models that are trained in order to detect objects relevant to the RoboCup Standard Platform League. That is, detecting NAO robots and a small football in images.

This study will answer the following research question:

- *How can using synthetic training data improve the performance of object detection models?*

Besides this, we want to gain insights in the factors that contribute to the performance of the trained models. This leads to the subquestion:

- *How do randomizations in a 3D environment that generates synthetic data influence the performance of an object detection model?*

[1] `https://spl.robocup.org/`
[2] `https://www.robocup.org/photos`

1.5 Related Work

Multiple papers have been released that document the use of synthetic datasets in the training of object detection models. Borkman et al. [3] describe their experiment, called SynthDet, in which they trained a Faster R-CNN model to detect grocery items. The experiment showed promising results as synthetic data contributed positively to the model's performance, but only when used in combination with real data.

Similar results were achieved by Vanherle et al. [21], who trained object detection models to detect industrial metal objects. They also conducted tests to investigate the importance of object positions and lighting in the environment. They emphasize that it is important to mimic the real world as closely as possible to get an accurate model. But this was solely the case when using transfer learning.

Another way of using 3D environments to generate effective synthetic datasets is described by Tremblay et al. [19], in which they train a model to detect cars. In their research they examine a so-called "domain randomization" method. With this method, they add different kinds of randomized objects to the 3D world that are not necessarily realistic (random meshes and textures). When trained this way, a model learns to focus only on the important features [19]. They found that such a model can even outperform models trained with photorealistic synthetic datasets.

The thesis by Deprez [4] provides insights into the use of synthetic datasets in the context of the RoboCup Standard Platform League. In particular, the thesis describes an approach with which images obtained from virtual environments can be augmented in order to resemble real-world images more closely. The augmented data was shown to more closely resemble real data by measuring the "Fréchet Inception Distance". The author notes that augmenting synthetic images can improve the performance of an instance segmentation model. In our project, we perform experiments with object detection models instead.

Hess et al. [5] describe their approach to designing a system that generates synthetic training images for the RoboCup SPL. To do so, they used the Unreal game engine. The framework calculates semantic segmentation masks for each of the generated synthetic images. Convolutional neural networks were then trained using only synthetic images. High segmentation accuracy was achieved.

In a previous project undertaken by Rogier van der Weerd at the University of Amsterdam [22] a variant of the YOLO object detection model was trained to detect Nao robots and a ball. The YOLO model was chosen because of its inference speed and accuracy. With this model, object detection tasks could be performed more accurately compared to an older "Haar" method. This particular model was then used to allow a Nao robot to do pathfinding tasks on a football field.

1.6 Thesis Outline

The thesis is structured in the following way. Chapter 2 provides relevant background information regarding this project. This includes information about the YOLO model, commonly used object detection metrics, the Unity Perception package and the virtual 3D environment we use in this project. Subsequently, Chapter 3 goes into the project method. In particular, we specify the datasets that we generate and use for our experiments, as well as the way we train our YOLO models. Next, in Chapter 4 we explain our results and findings[3]. Lastly, in Chapter 5 we conclude this thesis by formulating our insights and answering our research question.

[3] `https://github.com/Tobie1999/SyntheticObjectDetection`

CHAPTER 2

Theoretical background

2.1 Object Detection Models

Object detection is an important and well researched task within the field of computer vision. Object detection deals with the task of recognising contents of an image or frames of a video. In the past, multiple different approaches have been taken to address this problem. In this thesis, we use train and test a YOLO model. In this section we describe this model in further detail.

2.1.1 You Only Look Once (YOLO)

In this project, we train and test a particular model from the "You Only Look Once" (YOLO) line of object detection models. We now provide some details of this family of models.

The YOLO algorithms are an especially fast and accurate family of object detection systems. YOLO models are used in a wide variety of computer vision applications because of the speed at which they can make predictions. These applications include autonomous robots, self-driving cars and security systems [17]. With each new version of the YOLO algorithm, several unique and innovative techniques have been devised to improve the performance of the model. In this project, we use YOLO version 8 to do experiments. In order to understand the workings of this version of YOLO, we go over some of the main developments that have been made in the preceding versions.

2.1.2 Basics of the YOLO model

The paper describing the first version of the YOLO algorithm was released in 2015 [13]. The way the YOLO model made predictions was quite unique compared to other models at the time. Other object detection algorithms required multiple steps of execution in order to make bounding box predictions. For example, some algorithms used image classifiers at multiple regions in an image to perform object localization. Because of these repeated steps, such types of systems were difficult to optimize [17]. Instead of doing that, YOLO makes multiple bounding box predictions in an image with only a single network use [13]. This is the reason the algorithm got its name.

Besides the speed of YOLO, [13] mention other benefits to using YOLO as well. One of which is that YOLO makes very few false positive predictions because it looks at an image as a whole. Besides this, YOLO is better at recognizing objects in different situations compared to other models.

The algorithm

The YOLO model works by first separating an image into an S x S grid [13]. Each square in the grid is in charge of detecting an object whose center is in that square. B bounding box predictions are made for each square in the grid. The model also predicts so called "confidence

scores" associated with each bounding box. The confidence score indicates how sure the model is about the location of that particular bounding box prediction and the associated classification. The value of the confidence specifies how close a bounding box prediction resembles the ground truth bounding box. This is determined by the Intersection over Union, which we explain in Section 2.2.1.

A bounding box prediction is made up of five values. These include the coordinates (x and y), the dimensions (width and height) and the confidence score of the bounding box. The coordinates indicate the location of the center of the bounding box. They are relative to the square in the grid that the object is in. The width and height of the bounding box relative to the width and height of the image.

2.2 Metrics for object detection

In order to compare the performance of trained object detection models, it is important to understand the metrics that are commonly used to analyse them. We now provide an overview of these metrics.

2.2.1 Intersection over Union (IoU)

The metric "Intersection over Union" (IoU) is used to describe how well an object detection model predicts the bounding box placements and dimensions in an image [9]. The IoU is determined by comparing the bounding box prediction with the ground truth bounding box. To do so, we calculate the "Area of Overlap" and the "Area of Union" between the two bounding boxes [14]. To obtain the IoU we de divide the Area of Overlap by the Area of Union, as shown in figure 2.1.

An IoU value of one indicates a perfect prediction, where the predicted bounding is exactly the same as the ground truth bounding box. And an IoU of zero means that the predicted bounding box is not accurate at all, as the prediction and ground truth bounding box do not overlap. The "average precision" metric, described in Section 2.2.2, makes use of the IoU. It is important to note that the way the IoU is calculated inflicts penalties on predicted bounding boxes in two situations. Namely, when there is little overlap with the ground truth bounding box and too much overlap with the background [7].

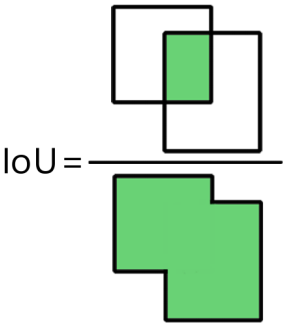


Figure 2.1: The IoU is computed by dividing the area of overlap by the area of union of a bounding box prediction and ground truth. This figure is inspired by the article by [15].

2.2.2 Average Precision (AP)

One of the most popular metrics used to measure object detection performance is the average precision (AP) [12]. If a model is trained to detect multiple classes, the AP is calculated for each class. Then we take the average of all the APs to get the Mean Average Precision (mAP).

The way in which researchers compute the average precision (and mAP) has changed over the years. The reason for this is because it was noticed that models which scored equally on the mAP test, did not perform equally in reality [8]. Even currently, different kinds of AP metrics are in use throughout the object detection scene.

TP, FP and FN

To determine the average precision of an object detection model, an understanding of the terms "true positive" (TP), "false positive" (FP) and "false negative" (FN) is essential. For object detection, the definitions of these terms are slightly different than for the task of image classification [8]:

- **True positive**: The bounding box prediction closely matches the ground truth bounding box. A bounding box prediction is measured as true positive if the IoU measure of that bounding box is greater or equal than a certain threshold value t.
- **False positive**: An incorrect bounding box prediction. This is the case when the IoU is lower than threshold t. In this case, the predicted bounding box either differs too much from the ground truth bounding box, or detects an object that is not present in the image [26].
- **False negative**: An object was not detected. This means that no bouning box prediction was made for an object that has a ground truth bounding box around it.

In the context of object detection, true negative does not exist, as the background does not need a bounding box prediction (there would be a near infinite ways to do this) [8].

Knowing these terms enables us to understand the "Precision" and "Recall" metrics.

Precision and Recall

- **Precision** indicates the fraction of correct positive predictions out of all the (positive) predictions:

$$\text{Precision} = \frac{\text{TP}}{\text{Total bounding box predictions}} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{2.1}$$

- **Recall** is the fraction of correct positive predictions out of all ground truth bounding boxes:

$$\text{Recall} = \frac{\text{TP}}{\text{Number of ground truth boxes}} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{2.2}$$

Ideally, the Precision and Recall of a model are both close to one. In reality however, there is often a trade-off between the two [12]. A model has a perfect precision if it makes no FP predictions. This can occur if the model is careful in making predictions and therefore only makes a bounding box prediction when it is absolutely certain that is has a high IoU. This, however, causes the model to have a high number of FN predictions, as a significant amount of ground truths are missed. This means that although the model has a very high Precision, the Recall could be very low. This also works the other way around. If the model makes a large number of predictions, the number of ground truths (TP + FN) it misses would be very low. This results in a high Recall value. However, an increase in the number of guesses can cause the number of false positive to grow. This leads to a lower Precision.

This trade-off property is important to keep in mind to understand the Precision-Recall curve.

Precision-Recall curve

We can calculate the Average Precision by determining the area under the Precision-Recall (P-R) curve. The P-R curve shows how the Precision and Recall of a model changes when only looking at all the bounding box predictions with a certain confidence value or higher. The best way to understand this is with an example, shown in Appendix A.

2.2.3 Mean Average Precision (mAP)

To describe the accuracy of a model in detecting all classes, the Mean Average Precision (mAP) metric is used. The Average Precision (AP) only describes a model's accuracy at predicting one particular object class. However, most of the time object detection models are used to detect different types of objects in an image. We calculate the mAP by first calculating the AP for each detectable class. Then we simply take the average of all the AP values to get the mAP. The mAP is a value between zero and one just like the AP. If the mAP is one the model has perfect accuracy and a value of zero means that it is not accurate at all.

It is important to note that the IoU threshold that is chosen influences the final mAP score. In object detection literature and challenges, such as PASCAL VOC 2010 and ImageNET, threshold values of 0.5 (mAP@0.5) and 0.75 (mAP@0.75) are standard [8]. It is also common to increment the IoU from 0.5 to 0.95 with steps of 0.05 (mAP@[0.5:0.05:0.95]). With this method, the Mean Average Precision mAP is calculated multiple times and then averaged.

2.3 Dataset generation with the Unity Perception package

To generate the synthetic datasets, we use the Unity game engine in combination with the "Unity Perception" package. The Unity Perception package was released in 2021 and adds functionalities to the Unity environment to create synthetic datasets [3]. The synthetic datasets that Unity Perception generates contain perfectly annotated training examples [3]. The Unity Perception team also mentions an additional benefit. Namely that the virtual environments and objects within them can be used many different times, each time with slight modifications. The Unity Technologies team mentions that synthetically generated datasets generated with the Perception package can successfully be used in combination with real labeled images. As the use of this combined dataset can even lead to a better model performance.

2.3.1 Unity Perception workflow

Setting up

After a Unity scene with an environment and objects is constructed, Unity Perception components can be set up[1]. The Perception package provides users with so called "**labeler**" components. With these labelers, different types of ground truths can be generated. The type of labelers to use depends on the computer vision task to be performed. The supported computer vision tasks include object detection (2D and 3D bounding boxes), instance segmentation and semantic segmentation (per-pixel classification). The "perception camera" component needs to be attached to the camera that we use to generate images with.

To specify the class that each object belongs to, we can attach the "**label**" component to those objects. The "labeler", attached to the perception camera component, then uses these labels to determine the ground truth class information. Users can specify the exact names that objects should be labeled as. However, the classes for each ground truth in the final dataset are represented as numerical IDs.

Randomizations

An object with a "scenario" component has to be created before randomizations can be configured. Different scenarios can be created with different randomization behaviours. The scenario also controls some important settings, such as the randomization seed and the number of images that are be generated. Multiple "randomizers" can be added to the scenario in any order. A large number of randomizers are available, so we only describe the ones used in this project.

The first one is the transform randomizer. With it, the position and rotation of all objects with a "transform randomizer tag" component are randomized. We can specify the bounds within which an object is randomly placed by setting minimum and maximum values for the three axes of motion (x, y and z). The same can be done for the three rotational axes.

The sun angle randomizer is used to vary the rotation of the directional light. The time of day, the day of the year, and the latitude can all be randomized.

The skybox randomizer is used to randomly pick a skybox from a list of skyboxes. The rotation of the skybox can also be randomized.

Properties of the global volume can be randomized using the volume randomizer. The properties include bloom effects, lens distortion, depth of field, motion blur and camera type. We only randomize the first two properties for this project.

[1] `https://docs.unity3d.com/Packages/com.unity.perception@1.0/manual/Tutorial/Phase1.html`

Finally, the developers of the Perception package provide additional code to perform dataset analyses and create visualizations. Examples are the number of objects per generated image and the occurrences of objects in the entire dataset.

2.4 Virtual RoboLab environment

The virtual environment that we use in this project is a replication of the Intelligent Robotics Laboratory at the University of Amsterdam (UvA) inside of Unity. It was kindly provided to us by Joey van der Kaaij, who works at the the Visualization Laboratory at the UvA.

Inside of the Unity environment is a complete RocoCup football field, along with NAO robot and football models. The Unity project also has Unity Perception installed and configured. Several objects inside of the scene have randomizers attached to them in order to generate diverse synthetic datasets. Figure 2.2 shows a screenshot of the environment.

Figure 2.2: A screenshot of the virtual Intelligence Robotics Lab at the University of Amsterdam inside of Unity. The environment was created by Joey van der Kaaij.

CHAPTER 3

Method

3.1 Real-World Datasets

3.1.1 SPL Dataset

In this project we use a subset of the images in the SPL dataset, made public by [23]. The dataset contains just under 5000 images. The images are all taken from the view of a NAO robot at positions on soccer fields used in Robocup tournaments. Over all the image, about 5500 NAO robots and 4500 footballs are visible. Some images contain multiple robots and some contain none. Diverse situations are captured in the set of images. Some occlusions of robots and the ball occur in the dataset. The lighting conditions also vary over the dataset, as some images are taken under natural lighting and some under artificial lighting. Some images are heavily blurred because of a fast motion of the camera. Images are also present with different people in the background.

We made some alterations to this dataset. In this project, we are only considering the detection of NAO robots and a football. The SPL dataset however, also contains two additional classes: goal post and pentalty spot. We removed all occurences of these labels because they are not needed in this project.

For our project, we randomly sample a smaller subset of 2500 images from the SPL dataset. This is our largest dataset made up of purely real-world data used in the project.

3.1.2 Experimental Real-World Datasets

For our experiments, we use multiple real-world datasets of different sizes. We do this to be able to compare models trained on real data with models trained on synthetic data at different levels of training. Also, we use these real-world datasets in combination with synthetic datasets in order to gain additional insights in model performance. All datasets are split into a training, validation and testing set. In this thesis we perform tests on four datasets containing only real-world images.

Our largest real-world dataset contains 2500 labeled images. It is split into a training set of 1500 (60%) training examples, a validation set of 500 (20%) images, and a testing set of 500 (20%) images. We obtained this dataset by randomly sampling images from the SPL dataset described in the previous section. The test set of 500 images are used for all tests performed in this project. This includes tests on models trained on real-world, synthetic and combined data.

The second largest dataset consists of 900 labeled images. It consists of a training set of size 300, a validation set of size 100, and a testing set of 500 images. The training set is a subset of the training set of our largest dataset and contains 20% of its images. In the same way, the validation set is a subset of the validations set of the largest dataset. As described above, the same 500 images are used for testing all models.

Next, we use a dataset that has 580 images total. The training set contains 60 images. This is also a subset of the training set of the largest dataset (4% of 1500). Similarly, the validation

set holds 4% of the images of largest dataset's validation set.

Lastly, only 516 images make up the smallest dataset. Of those 516 images, 12 are the training set and 4 the validation set. Like with the previous datasets, the validation and training sets are subsets of the training and validation sets of the largest dataset (0.8%).

Below is a concrete list of the four real-world datasets:

- Real-world dataset 1: Size 2500, 1500 Train, 500 Validation, 500 Test
- Real-world dataset 2: Size 900, 300 Train, 100 Validation, 500 Test
- Real-world dataset 3: Size 580, 60 Train, 20 Validation, 500 Test
- Real-world dataset 4: Size 516, 12 Train, 4 Validation, 500 Test

3.2 Synthetic Datasets

We construct multiple synthetic datasets to perform experiments with. The datasets differ both in size and in the randomizations applied during the generation process. We run experiments with models trained on these synthetic datasets, as well as combinations with real-world datasets described in Section 3.1.2. The synthetic datasets are all generated using the Unity Perception package.

3.2.1 Converting SOLO to YOLO format

YOLO is unique in the way its dataset folders are structered and how it specifies bounding boxes. The Unity Perception package outputs its images and ground truth bounding boxes in a different format, called SOLO. In order to train a YOLO model with the data from Unity Perception, we have to convert the datasets from the SOLO format to the YOLO format.

The Perception package outputs a single folder which contains folders that correspond to each generated image. Each folder contains the generated image and a JSON file containing the ground truth bounding boxes and some metadata. The bounding boxes are defined differently than the way YOLO defines them. The position of the bounding boxes are defined by their pixel coordinate, from the top left of the image to the top left of the bounding box. The width and height are also expressed in pixels. However, the YOLO format defines the coordinates of the bounding box centres, relative to the size of the image. The dimensions are also relative to the image size. Besides these differences, the folder structures also differ, as YOLO separates the data into train, test and validation sets. Unity Perception outputs all images and bounding boxes into a single folder.

We wrote code to do this conversion, as well as to split the data into training, validation and testing sets [1].

3.2.2 Randomized datasets

While generating synthetic datasets using the Perception package, we apply multiple randomizations to our Unity scene. The reason for this is that, hypothetically speaking, a more diverse training dataset leads to a model that is able to detect objects in more situations. In order to create these randomizations, we use the randomizers provided by the Unity Perception package. We describe randomizers in Section 2.3.1.

In total, we randomize five components in our scene. The randomization parameters were specifically chosen to create a dataset that resembles a real-life dataset. The five Perception randomizers we use are: TransformRandomizer, LightRandomizer, SunAngleRandomizer, VolumeRandomizer, SkyboxRandomizer.

Below follows an explanation of the settings we used for each of these randomizers.

[1] `https://github.com/Tobie1999/SyntheticObjectDetection`

Transform randomization

The transform ramdomizer varies the location and rotation of objects in each generated image. We attach the "Transform Randomizer Tag" component to the objects whose transform we want to randomize. In our case, we attach the tag to six NAO robots, one football and the camera. The tag settings for the position are the same for all objects, namely:

- X: between -40 and 40 units (uniformly sampled)
- Y: always 0 units
- Z: between -30 and 30 units (uniformly sampled)

The positions are relative to the starting position of the object (the center of the football field). The position ranges are set to these values so the robots are placed over the whole football field.

The rotation randomizer settings differ slightly between the three types of objects. NAO robots are only rotated around the Y axis (between 0°and 360°, uniformly picked). The ball is rotated around all three axes (also between 0°and 360°). The camera is mainly rotated around the Y axis (between 0°and 360°), but is also given a slight tilt around the X and Z axes (between -2°and 2°).

Light intensity/temperature randomization

The two main sources of light in the scene are the directional light and the ceiling lights. All of these objects have the "Light Randomizer Tag" components attached to them in our scene. With the tag we can randomize the intensity and the temperature of the light. The directional light's randomizations are set as follows. The intensity is randomly picked between 15 and 500 lux (uniformly sampled). The temperature is set between 4000 and 7200 kelvin (uniformly sampled). As for the ceiling lamps, the intensity is a random value between 900 and 9500 lux (uniformly sampled) and the temperature a value between 2400 and 9000 kelvin (uniformly sampled).

Sun angle randomization

We vary the time of day by setting up the "SunAngleRandomizer". The scene contains a single directional light, which has a "Sun Angle Randomizer Tag" connected to it. We use the following randomization parameters:

- Hour: between 0 and 24
- Day of the year: between 0 and 364
- Latitude: between -90 and 90

Global volume randomization

Within a Unity Scene, an object with the global volume component can control various post-processing effects. For our experiments, we have a global volume object whose settings we vary using the "VolumeRandomizer". Specifically, we randomize the "bloom" and "lens distortion" volume effects. We use the following settings on the volume randomizer tag:

- Bloom: threshold between 0 and 0.75, intensity between 0 and 1 and scatter between 0 and 0.75.
- Lens distortion: intensity between -0.5 and 0.5, X and Y multiplier between 0 and 1, center at X = 0.5 and Y = 0.5, and scale between 0.01 and 5.

Skybox randomization

Lastly, we randomize the skybox of our scene. One out of eight skyboxes is randomly picked for each generated image. The skybox is visible when looking out of the window of the virtual RoboLab environment.

We generated 1600 synthetic images with all of these randomizations enabled. We set the number of robots in the 3D environment to six, as this results in a robot count per image that roughly matches the real-world SPL dataset. The frequencies of the object counts per image is visible in figure 3.1. In total, 2670 NAO robots and 400 footballs are present in the dataset. From these synthetic images we created three datasets to experiment with.

Our largest synthetic dataset contains 2100 images. 500 of these images are real-world images only used for testing. These are the same 500 real images that we use to test every model in this project (Section 3.1.2). The 1600 remaining images are synthetic images, split into a training set of 1200 images and a validation set of 400 images.

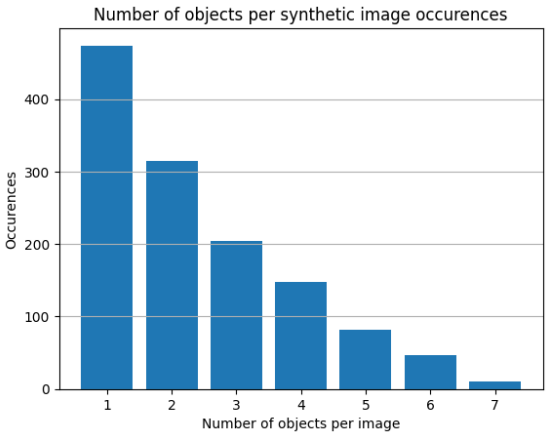


Figure 3.1: Occurrences of the object counts per image. In the 3D environment are six robots and one football.

We also created a dataset containing 1300 images. 800 of the images are synthetic, split into a training set of 600 images and a validation set of 200 images. The training set is a subset of the training set of the largest synthetic dataset. In the same way the validation set is a subset of the largest datasets validation set. The remaining 500 images are real and used for testing.

Lastly, the smallest synthetic synthetic dataset contains 900 images, split into 300 synthetic training images, 100 synthetic validation images and 500 real-world testing images. The training set is a subset of the training set of the synthetic dataset of 1300 images, and the validation set a subset of its validation set.

Below is a list of the synthetic datasets:

- Synthetic dataset 1: Size 2100, 1200 Train, 400 Validation, 500 Test (real-world)
- Synthetic dataset 2: Size 900, 600 Train, 200 Validation, 500 Test (real-world)
- Synthetic dataset 3: Size 580, 300 Train, 100 Validation, 500 Test (real-world)

In addition, we also have 3 additional synthetic datasets with 1500, 300, 60 and 12 training examples. We did this in order to compare the accuracy with the real-world datasets for the same amounts of training data.

3.2.3 Datasets for randomization experiments

The datasets described here are made in order to find out how much each randomizable setting contributes to the accuracy of the resulting model. For example, the Perception package can randomize the intensity of the lights in each generated image when provided with a minimum and maximum value (0 and 500 lux in our case). For each image, Unity uses these values to sample an intensity value from a uniform distribution. However, it is also possible to use a fixed intensity for all generated images.

With this experiment, we generate datasets keeping all but one parameter randomized. The parameter that we do not randomize is kept constant throughout the image generation process. In some cases we generated multiple datasets for a parameter, each with a different constant value for that parameter (e.g. one dataset with a light intensity of 50 lux, one with an intensity of 250 lux, and one with an intensity of 500 lux).

Our hypothesis is that a better synthetic dataset (for yolov8 model training) can be generated by randomizing the different Unity Perception parameters (light intensity/temperature, sun angle, blurriness and skybox) instead of keeping the parameters constant. This hypothesis seems intuitive because randomizing the parameters gives a more diverse dataset, and a model trained on a diverse dataset should be able to accurately detect objects in more situations. Also, the real-world dataset that we are testing our models with contains images taken in different

situations. This should mean that a model trained with diverse synthetic data performs better on the real-world test set.

For each experiment we use 300 synthetic training examples in combination with 300 real training examples to train a model. The synthetic datasets are generated using the same seed each time. We do this to only measure the effect of disabling one randomization. When we describe our results in Chapter 4.1.3, we only consider the dataset with the best result for each randomizer configuration.

3.3 Combined Datasets

Other types of datasets we use in our project are combined datasets. These datasets contain both real-world and synthetic images. To create combined datasets, we use the real-world and synthetic datasets described in the previous sections.

To create the combined dataset, we combine the real-world and synthetic datasets by adding their entire training sets together. We do the same with the validation sets. The test set consists of the same 500 real-world images used as test set throughout this project. We created combinations with the datasets described in the previous sections in order to get a insights into what the effect is on performance at different blends of real and synthetic data.

3.4 YOLOv8 nano Training

All models in this research are trained for 50 epochs with a batch size of 16. The hyperparameters are set to the default values of YOLOv8. However, we also analyzed and compared the accuracy of models with different hyperparameter settings. An increase of accuracy can be achieved when modifying one of the mosaic augmentation parameters. Setting the *close_mosaic* parameter to 10 will increase a models mAP[0.5] score (when trained with 300 real-world images) by nearly 2 percentage points.

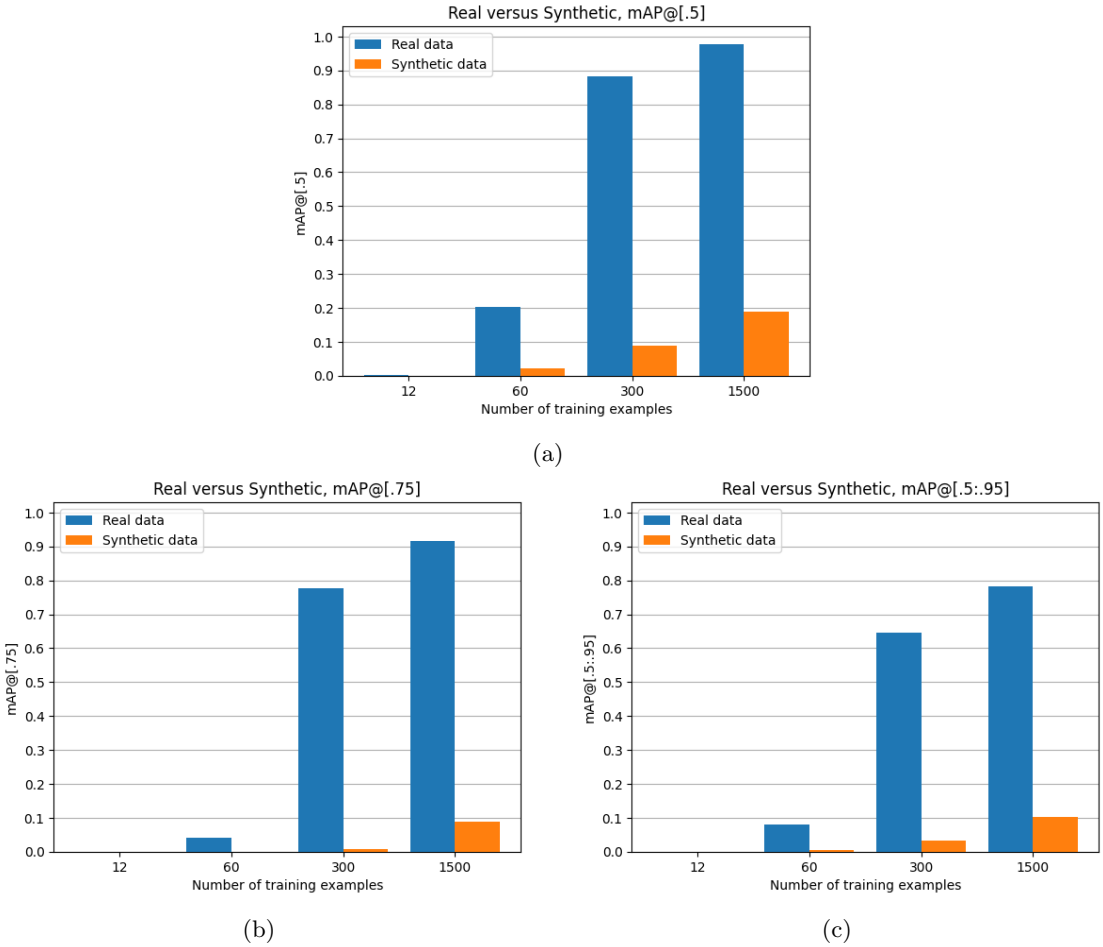
We also experiment with training a YOLOv3 model using different combinations of real and synthetic data. These models are also trained for 50 epochs. We do this in order to obtain more robust findings. Where possible, the hyperparameters are set to the same values as for the YOLOv8 model.

CHAPTER 4

Experiments

4.1 Results

4.1.1 Real-world versus synthetic datasets



(a)

(b)

(c)

Figure 4.1: Diagrams showing the difference between the accuracy of models trained using only real-world images versus those trained using only synthetic images. Plots (a), (b) and (c) show the metrics mAP@[.5], mAP@[.75] and mAP@[.5:.95] respectively.

Figure 4.1 shows comparisons between models trained on real-world data and those trained on synthetic data. As can be seen, models trained using only real-world data greatly outperform models trained using only synthetic data at multiple training set sizes. The law of diminishing returns is visible. Significant mAP score improvements are visible when using 300 training examples instead of 60. At 300 training examples, models trained on real-world data already reach sufficient accuracy (mAP@[0.5] = 0.882). However, there is still room for improvement. The three metrics show a similar pattern. We will therefore only focus on the first metric (mAP@[0.5]) in the following experiments.

4.1.2 Combined datasets

We now look at the performance of models that are trained using datasets that consist of both real-world and synthetic images. Figure 4.2 shows that adding synthetic data to real data in the training set improves the performance of the model. The blue bars in the figure show that using only 300 real-world training images to train a YOLOv8 model results in an mAP@[.5] of around 0.88. The orange bars show the accuracy boosts that are obtained by adding 300, 600 and 1200 synthetic training examples to the 300 real-world examples. In all three cases, the model's accuracy improves. To check whether the results are robust, some experiments were repeated a number of times. This can be seen in Appendix C

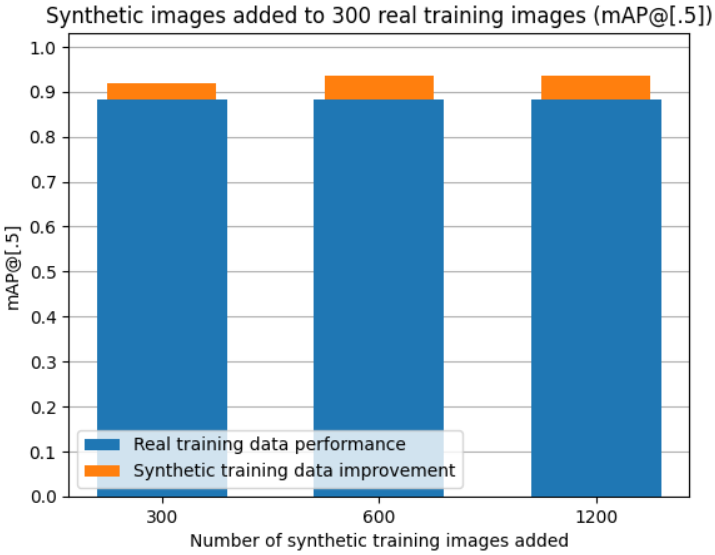


Figure 4.2: The accuracy of YOLOv8 models trained on a combined dataset of real-world and synthetic images. The figure shows the mAP of models trained on 300 real-world images (blue bars), as well as the mAP improvements observed when adding 300, 600 and 1200 synthetic images to the 300 real images (orange bars).

The performance boost is especially apparent when the number of real-world images is not very large, such as 300. If the model is already quite accurate, because it has been trained on more real-world data, adding synthetic data only results in a small performance boost. For example, at 1500 real-world training examples, the accuracy of the model is already high (almost 0.98 mAP[0.5]). This leaves little room for improvement. Figure 4.3 shows that the addition of synthetic data does not lead to significant accuracy improvements, as the orange bars are barely visible.

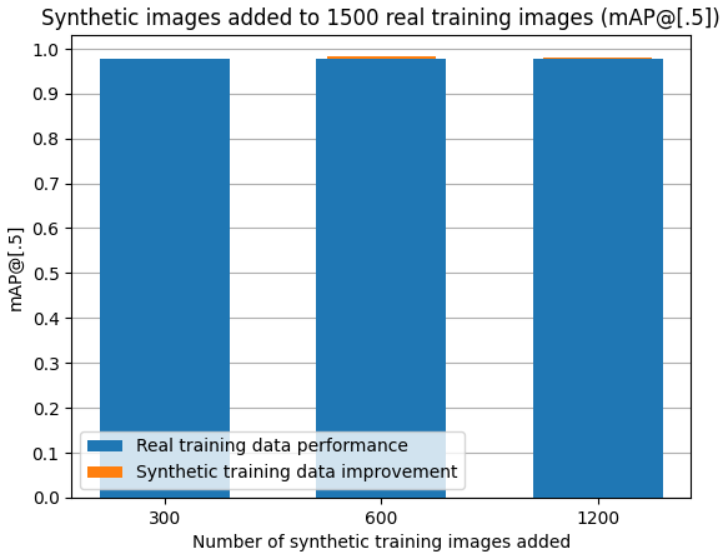


Figure 4.3: The mAP of models trained on 1500 real-world images (blue bars), as well as the mAP improvements observed when adding 300, 600 and 1200 synthetic images to the 300 real images (orange bars).

Adding synthetic data to the training set only improves the accuracy of a model up to a certain amount. For example, the accuracy of a model increases when adding up to approximately 600 synthetic images. After that no significant performance improvements are achieved anymore. This can be seen in Figure 4.4. The figure shows by how many mAP[.5] points a model's accuracy increases when adding different amounts of synthetic training examples to 300 real-world training examples. The mAP increases by roughly 0.037 points when adding 300 synthetic training examples. The accuracy further improves when 600 synthetic examples are added to the 300 real-world examples, increasing the mAP by approximately 0.053. No further improvements are made to the model when adding 1200 synthetic training examples, as the mAP also improves by 0.053 points. A similar pattern is visible for the model that uses 1500 real-world training images, as can be seen in Figure 4.5.

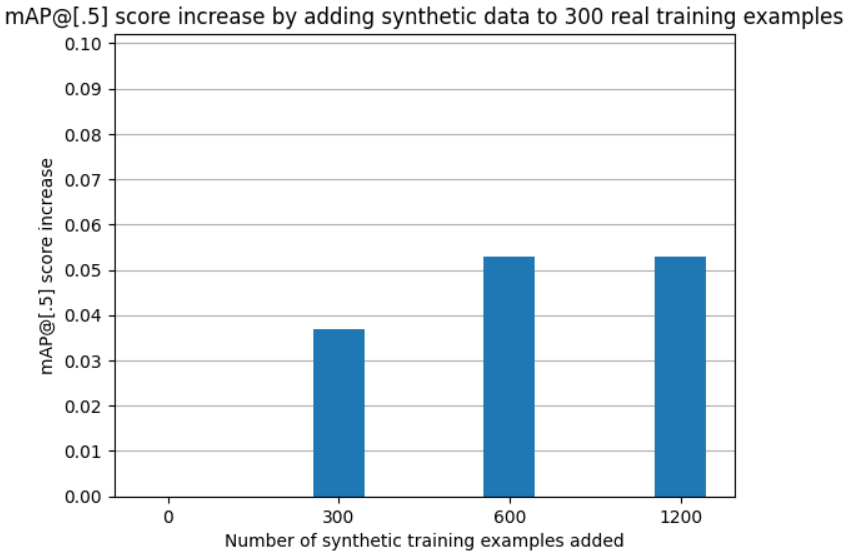


Figure 4.4: The mAP@[0.5] increase when using 300 real-world images in combination with different amounts of synthetic images

Furthermore, Figure 4.5 below shows that adding synthetic images to an already large training set of real-world images only leads to a small performance boost. The main reason for this is that the model already reaches a near perfect mean average precision when trained on that amount of real-world data.

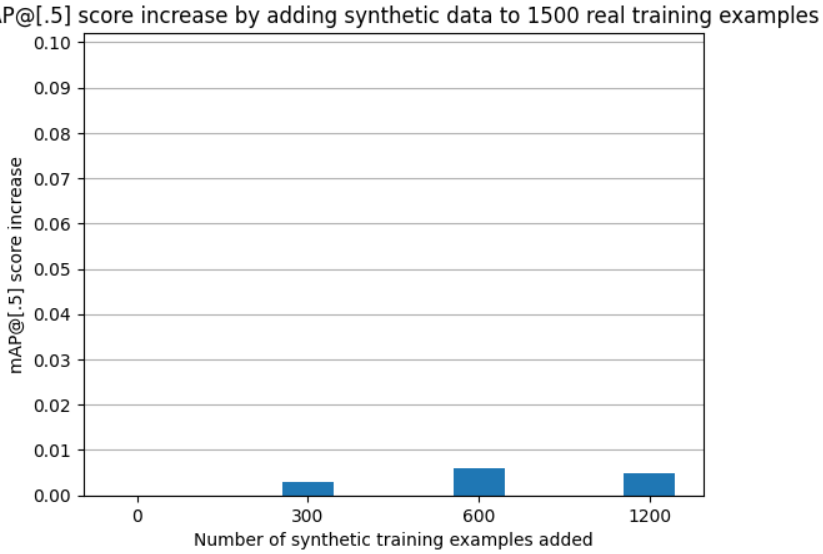


Figure 4.5: The mAP@[0.5] increase when using 1500 real-world images in combination with different amounts of synthetic images

The mAP measurements we observed from models trained using synthetic, real and combined datasets are visible in table 4.1. Besides these tests, we also performed similar experiments on YOLOv3 models. The results are described in Appendix B.

Table 4.1: Mean average precision of YOLOv8 nano models trained on real-world, synthetic or combined datasets (both football and NAO robot).

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
12	0	0.0009	0.0002	0.0003
60	0	0.204	0.040	0.079
300	0	0.882	0.777	0.645
1500	0	**0.976**	**0.917**	**0.783**
0	300	0.089	0.008	0.032
0	600	0.161	0.060	0.074
0	1200	**0.183**	**0.082**	**0.092**
300	300	0.919	0.807	0.682
300	600	0.935	0.798	0.680
300	1200	**0.935**	**0.824**	**0.698**
1500	300	0.979	0.917	0.787
1500	600	**0.982**	**0.928**	**0.798**
1500	1200	0.981	0.927	0.796

Another finding is that using real-world data, if available, is still preferred over synthetic data. Figure 4.6 shows that a model performs better if a large share of its training set is made up of real-world images. The figure shows that exchanging real data for synthetic data slowly deteriorates the accuracy. It can therefore be concluded that real data is more valuable than synthetic data.

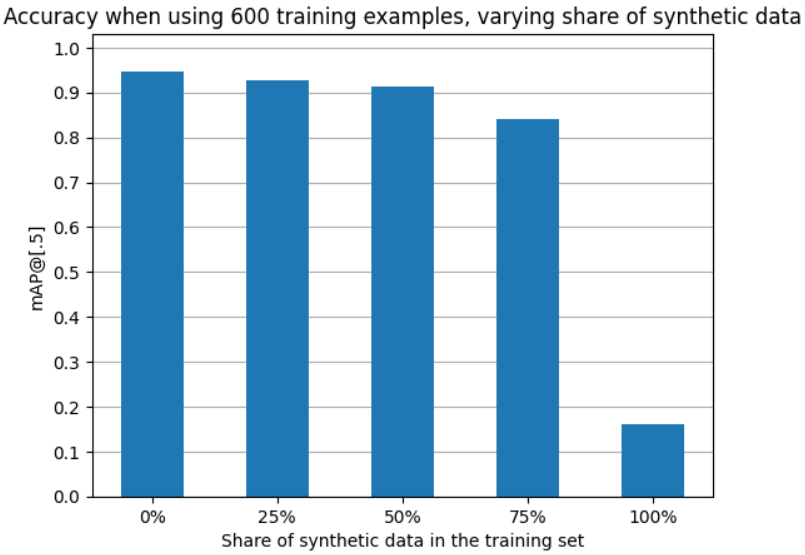


Figure 4.6: The mAP[0.5] of models trained using 600 training examples. Each model is trained using a different percentage of synthetic data in its dataset.

Based on the results of the previous experiments we can conclude that even though synthetic data does not perform well on its own, it can result as a performance boost when used in addition to available real-world data.

Accuracy in predicting the football and robot class

We also analyze the accuracy of models at predicting a football and NAO robot separately. Again, models trained purely with real-world data perform significantly better than models trained with only synthetic data. This applies to both the detection of a football and a robot. This is illustrated in Figure 4.7. Nevertheless, adding synthetic data to a real-world training set improves the accuracy in predicting both the football and the robot, as shown in Figure (4.8).

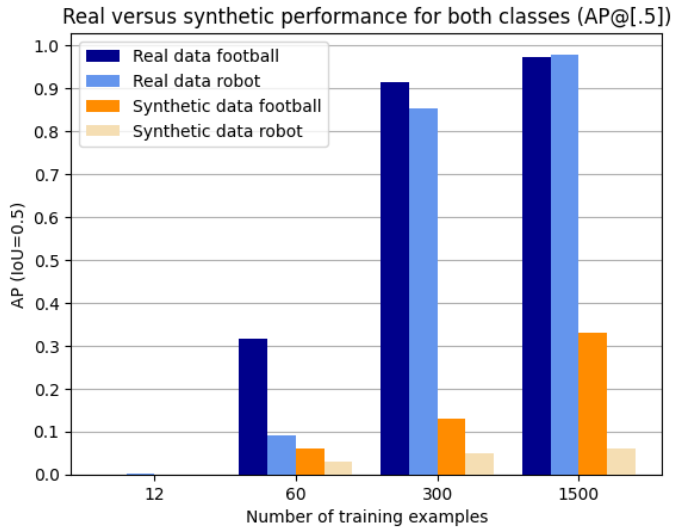


Figure 4.7: The accuracy of models in predicting the football and robot class. The models are trained with real-world images or synthetic images. The blue bars indicate the model's accuracy when trained using only real-world data. The orange bars show the accuracy when trained on only synthetic data. The dark blue bar shows the model's accuracy in detecting the football, and the light blue bar the accuracy in detecting the robot. The same applies to the dark and light orange bars.

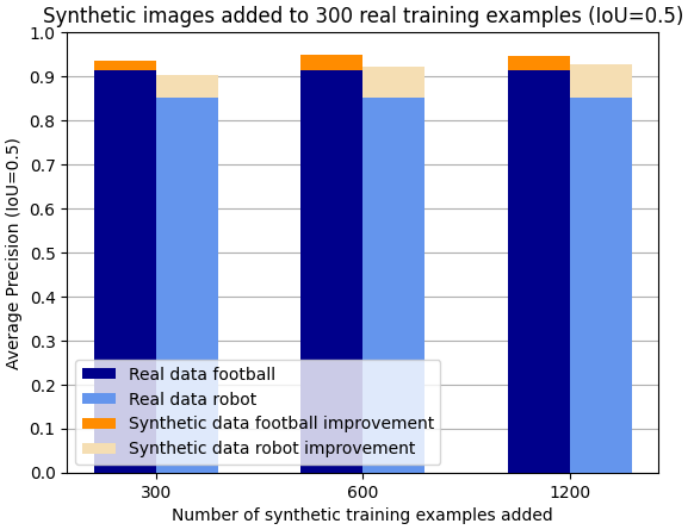


Figure 4.8: The accuracy of models trained using 300 real-world images in addition to 300, 600 and 1200 synthetic images. The dark and light blue bars show the performance of models trained on 300 real images in detecting a football and a NAO robot. The orange bars show the accuracy improvements when adding different amounts of synthetic training examples.

An observation can be made when using only real-world data for model training. At small amounts of real data, such as 300 images, there is a discrepancy between the accuracies in detecting the football and the robot. The model detects footballs with higher accuracy (0.91 AP) than robots (0.85 AP). A possible cause for this is that the NAO robot is a more complex object. When synthetic data is added to a real-world training set of this small quantity, more improvements are made in detecting the robot than the football. This is visible in figure 4.9. This means that synthetic data helps in bridging the gap between accuracy in predicting different classes.

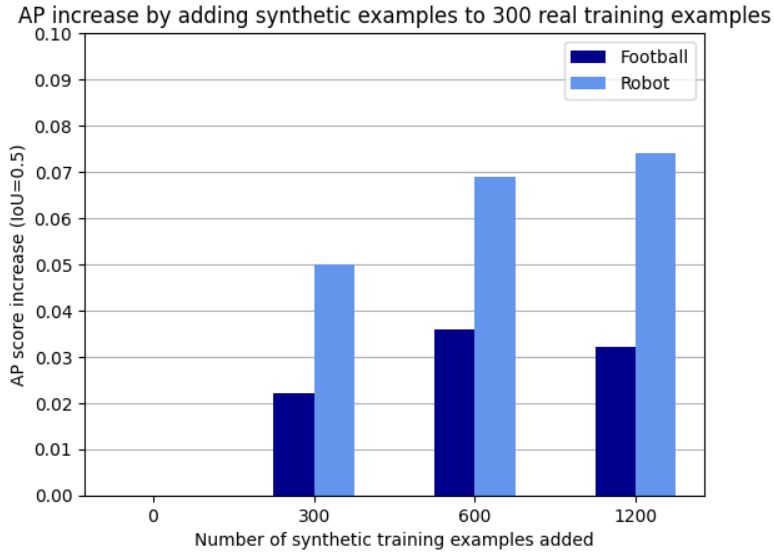


Figure 4.9: The AP increase when adding different amounts of synthetic images to 300 real-world training examples. The dark and light blue bars show the accuracy increase in detecting the football and robot respectively.

4.1.3 Effect of randomizations

We performed another experiment where we analyze the performance of a model when trained using different randomization settings in the 3D environment. In the previous section, we randomized multiple settings for each generated synthetic image. The following attributes inside the Unity scene were randomized: the positions of objects, lighting properties, the angle of the sun, bloom effects, distortion effects, and the skybox.

In order to understand the effect of the randomizations on the model's accuracy, we fixed each one of the aforementioned settings separately, except for the position randomizations, and tested the performance of the resulting models. The results are visible in figure 4.10. We also tested the performance of the model when keeping every setting fixed except the positions randomizations ("mostly fixed" in figure).

From the figure we can conclude that randomizing all settings results in a synthetic dataset that can be used to boost the performance of a model. Fixing settings did not lead to significant accuracy improvements compared to randomizing everything. The most accuracy was achieved when we kept the sun angle fixed inside of the 3D environment.

The exact measurements are visible in Appendix D.3.

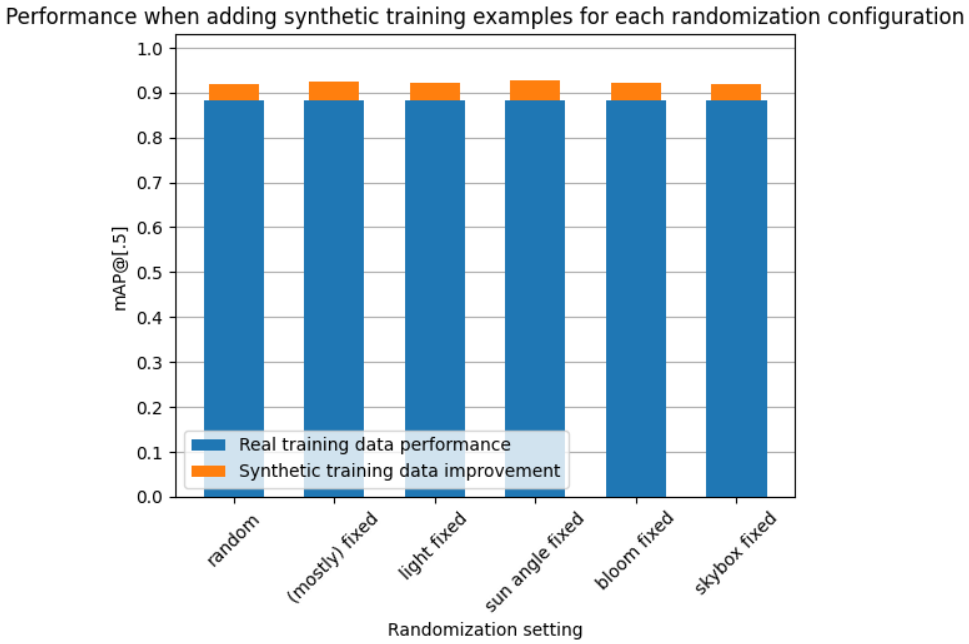


Figure 4.10: mAP@[0.5] of models trained with 300 real and 300 synthetic images. The synthetic training examples are generated using a different randomization configuration each time. All synthetic datasets had positions of objects randomized, and some had only one fixed parameter.

Figure 4.11 shows the accuracy improvements. Adding synthetic data leads to better results for all randomization configurations. Even randomizing only the positions of objects leads to a sufficient accuracy boost. Fixing the angle of the sun leads to a slightly better result compared to the other randomization configurations. However, this is not a significant improvement.

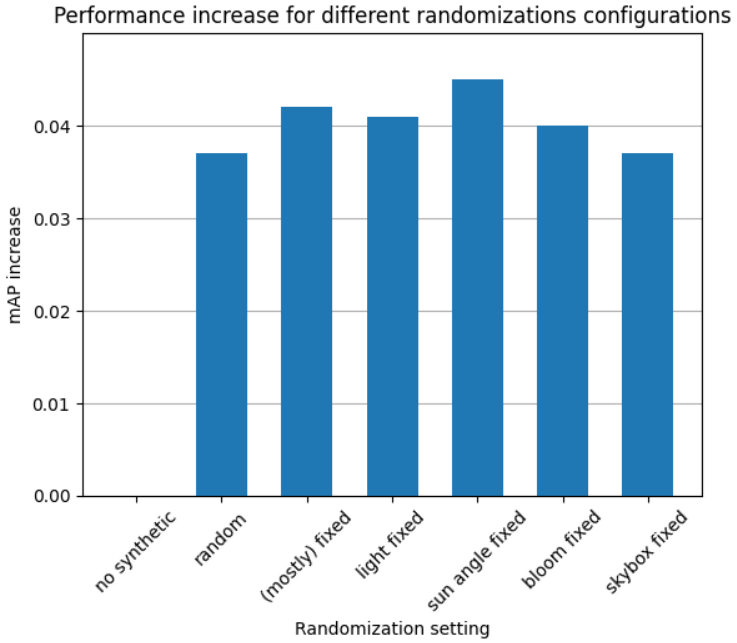


Figure 4.11: The mAP@[0.5] increase when adding synthetic data generated with different randomization configurations inside of the 3D environment. 300 synthetic training examples are added to 300 real-world examples each time.

CHAPTER 5

Discussion

5.1 Comparison with Relevant Works

The results obtained are consistent with the findings of previous studies. Previous studies have shown that synthetic data on its own does not lead to reasonable model performance compared to real data [3] [21]. This is most likely due to the so-called "reality gap" between synthetic data and real data [18]. The studies conclude that adding synthetic data to real data leads to satisfying results. We have also noticed this with our research. The research by Vanherle et al. [21] also shows that models perform better when a large proportion of the training data consists of real data. We have also seen this.

However, our observations differ from one of the relevant sources when it comes to the effect of randomizations. Vanherle et al. [21] conclude in their study that randomizing lighting settings leads to better performance. However, our research shows that this is not always the case. Fixing certain parameters in the 3D environment, such as the angle of the sun, did not degrade the model. A possible explanation for this is that a large part of the SPL dataset consists of photos with similar lighting conditions. As a result, keeping certain settings fixed can yield better results than the fully randomized settings.

5.2 Conclusion

As mentioned in Section 1.4 this project's goal is to explore object detection model performance when trained, either entirely or partially, using synthetic datasets. We particularly focused on detecting objects relevant to the RoboCup Standard Platform League, a football and NAO robots.

The main research question we wanted to answer was:

- How can using synthetic training data improve the performance of object detection models?

This research question now has an answer. Using synthetic data as training data on its own will result in poor model performance, especially when compared to the same amount of real-world training. However, the strength of synthetic data lies in using it in addition to real-world data.

From our experiments we found that a significant performance improvement can be obtained when only adding a small amount of synthetic data (300) to a real-world dataset. The performance boost is especially significant if the number of real images available is not very high. If a lot of real data is already present in a dataset, adding synthetic data will not result in a significant accuracy improvement.

Another important finding is if we add synthetic images beyond a certain amount, the performance will not increase anymore. In fact, it can even cause the model's accuracy to slightly decrease.

We also wanted an answer to the following subquestion:

- How do randomizations in a 3D environment that generates synthetic data influence the performance of an object detection model?

During our project we experimented with different randomization settings. From our experiments we conclude that a dataset obtained from a virtual environment with all parameters randomized can improve model performance. However, keeping certain parameters fixed, such as the lighting and sun direction, can cause the model to perform slightly better. However, the effects of this are not significant.

5.3 Ethical Aspects

There are concerns regarding data privacy when it comes to machine learning. According to Borkman et al. [3], generating synthetic images from 3D environments can help mitigate this problem. Choosing to use 3D simulations to create images gives us more control over the images that end up in the resulting dataset. However, in our case we concluded that synthetic training examples are only effective when used in combination with real-world data. Studies on synthetic datasets for other applications [3] [19] [21] draw the same conclusion. Since synthetic datasets can not be completely relied on, we emphasize that privacy is still a major concern when it comes to creating object detection datasets.

Furthermore, the usage of synthetic training sets may cause people to lose their jobs. Companies hire employees to annotate and audit datasets [21]. Less annotators and auditors are needed to create a dataset if synthetic images are added to them [3]. More research is needed to gain insights into potential loss of employment in this field of work.

5.4 Future Work

Future research could focus on doing experiments with different real-world datasets. Throughout our project we only used the SPL dataset described in Section 3.1.1. These studies are needed to arrive at more robust findings on the impact of synthetic training data on model accuracy. Future studies could also look at the effect of synthetic data using newer versions of the YOLO model to be released in the future, or experiment with entirely different models.

Bibliography

[1] Qiang Bai et al. "Object detection recognition and robot grasping based on machine learning: A survey". In: *IEEE access* 8 (2020), pp. 181855–181879. DOI: 10.1109/ACCESS.2020.3028740.

[2] Ari Yair Barrera-Animas and Juan Manuel Davila Delgado. "Generating real-world-like labelled synthetic datasets for construction site applications". In: *Automation in Construction* 151 (2023), p. 104850. ISSN: 0926-5805. DOI: https://doi.org/10.1016/j.autcon.2023.104850. URL: https://www.sciencedirect.com/science/article/pii/S0926580523001103.

[3] Steve Borkman et al. "Unity Perception: Generate Synthetic Data for Computer Vision". In: *ArXiv* abs/2107.04259 (2021). DOI: 10.48550/arxiv.2107.04259.

[4] Hidde G.J. Lekanne Deprez. "Enhancing simulation images with GANs". 2020.

[5] Timm Hess et al. "Large-Scale Stochastic Scene Generation and Semantic Annotation for Deep Convolutional Neural Network Training in the RoboCup SPL". eng. In: *RoboCup 2017: Robot World Cup XXI*. Lecture Notes in Computer Science. Cham: Springer International Publishing, 2018, pp. 33–44. ISBN: 3030003078.

[6] Matthew Johnson-Roberson et al. "Driving in the matrix: Can virtual worlds replace human-generated annotations for real world tasks?" In: *arXiv preprint arXiv:1610.01983* (2016). DOI: 10.48550/arXiv.1610.01983.

[7] Kukil. *Intersection over Union (IoU) in Object Detection Segmentation*. 2022. URL: https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/#Intersection-over-Union-in-Object-Detection/.

[8] Kukil. *Mean average precision (MAP) in object detection*. 2022. URL: https://learnopencv.com/mean-average-precision-map-object-detection-model-evaluation-metric/.

[9] Rohit Kundu. *YOLO: Algorithm for Object Detection Explained [+Examples]*. 2023. URL: https://www.v7labs.com/blog/yolo-object-detection (visited on 05/08/2023).

[10] Charles Thane MacKay and Teng-Sheng Moh. "Learning for Free: Object Detectors Trained on Synthetic Data". In: *2021 15th International Conference on Ubiquitous Information Management and Communication (IMCOM)* (2021), pp. 1–8. DOI: 10.1109/IMCOM51814.2021.9377353.

[11] Farzan Erlik Nowruzi et al. "How much real data do we actually need: Analyzing object detection performance using synthetic and real data". In: *arXiv preprint arXiv:1907.07061* (2019). DOI: 10.48550/arXiv.1907.07061.

[12] Rafael Padilla, Sergio L. Netto, and Eduardo A. B. da Silva. "A Survey on Performance Metrics for Object-Detection Algorithms". In: *2020 International Conference on Systems, Signals and Image Processing (IWSSIP)* (2020), pp. 237–242. DOI: 10.1109/IWSSIP48289.2020.9145130.

[13] Joseph Redmon et al. "You Only Look Once: Unified, Real-Time Object Detection". In: *CoRR* abs/1506.02640 (2015). DOI: 10.1109/CVPR.2016.91. arXiv: 1506.02640. URL: http://arxiv.org/abs/1506.02640.

[14] Seyed Hamid Rezatofighi et al. “Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (2019), pp. 658–666. DOI: 10.1109/CVPR.2019.00075.

[15] Adrian Rosebrock. *Intersection over union (IOU) for object detection*. 2022. URL: https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/.

[16] Mohsen Soori, Behrooz Arezoo, and Roza Dastres. “Artificial intelligence, machine learning and deep learning in advanced robotics, A review”. In: *Cognitive Robotics* (2023).

[17] Juan Terven and Diana Cordova-Esparza. *A Comprehensive Review of YOLO: From YOLOv1 and Beyond*. 2023. DOI: 10.48550/arXiv.2304.00501. arXiv: 2304.00501 [cs.CV].

[18] Josh Tobin et al. *Domain Randomization for Transferring Deep Neural Networks from Simulation to the Real World*. 2017. arXiv: 1703.06907 [cs.RO].

[19] Jonathan Tremblay et al. “Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomization”. eng. In: *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. IEEE, 2018, pp. 1082–10828. ISBN: 9781538661000. DOI: 10.1109/CVPRW.2018.00143.

[20] Viswanatha V, Chandana R K, and Ramachandra A C. “Real Time Object Detection System with YOLO and CNN Models: A Review”. eng. In: (2022). DOI: 10.48550/arxiv.2208.00773.

[21] Bram Vanherle et al. “Analysis of Training Object Detection Models with Synthetic Data”. eng. In: (2022). DOI: 10.48550/arxiv.2211.16066.

[22] Rogier van der Weerd. *Project AI: Real-time object detection and avoidance for autonomous Nao Robots performing in the Standard Platform League*. Tech. rep. July 13, 2021. DOI: 10.13140/RG.2.2.26378.18885.

[23] ZhengBai Yao et al. “Faster YOLO-LITE: Faster Object Detection on Robot and Edge Devices”. In: *RoboCup 2021: Robot World Cup XXIV*. Ed. by Rachid Alami et al. Cham: Springer International Publishing, 2022, pp. 226–237. ISBN: 978-3-030-98682-7. DOI: 10.1007/978-3-030-98682-7_19.

[24] Syed Sahil Abbas Zaidi et al. “A survey of modern deep learning based object detection models”. eng. In: *Digital signal processing* 126 (2022), pp. 103514–. ISSN: 1051-2004. DOI: 10.1016/j.dsp.2022.103514.

[25] Cheng Zhou and Chenhui Pei. “The Future Training of Sports Intelligent Robot Technology”. In: *system* 3.13 (2021), pp. 7–13. DOI: 10.25236/IJFS.2021.031302.

[26] Haidi Zhu et al. “A Review of Video Object Detection: Datasets, Metrics and Methods”. In: *Applied Sciences* (2020). DOI: 10.3390/app10217834.

APPENDIX A

Precision-Recall Curve Calculation Example

Below follows an example of how the area under the precision-recall curve (AP) is calculated.

Table A.1 shows ten bounding box predictions. The predictions are not real as they are just for illustrative purposes. The bounding box predictions in this table could have been the results from analyzing multiple images, or just one. In this example, we assume that there were five total ground truth boxes that had to be correctly guessed by the model. The predictions in this table are sorted by their confidence values in descending order. Also, the table displays for each prediction whether it was true positive of false positive (we assume that an appropriate IoU threshold was chosen in order to determine this).

In order to construct the P-R curve, we first have to compute the cumulative number of TPs and FPs for different minimum bounding box confidence values [26]. We do this by simply adding one to the cumulative TP or FP value for each row. The results of this are shown in the Cumulative TP and FP columns. Then we calculate the Precision and Recall for each confidence level, as shown in the last two columns. We use these Precision and Recall values as coordinates in order to plot the P-R curve. The P-R curve plot is shown in blue in figure A.

Table A.1: An example showing the calculations needed to construct a Precision-Recall curve. The P-R curve is needed to calculate the Average Precision metric. The bounding box predictions are sorted by their confidence level in descending order from top to bottom. The Precision and Recall are computed for each confidence level by using the cumulative TP an FP values. In this example, it is assumed that there are a total five ground truth bounding boxes to be predicted. The plot of the P-R curve is visible in figure A.

Bounding box prediction	Confidence	TP/FP	Cumulative TP	Cumulative FP	Precision $(\frac{TP}{TP+FP})$	Recall $(\frac{TP}{5})$
1	0.94	TP	1	0	1	1/5
2	0.93	FP	1	1	1/2	1/5
3	0.85	FP	1	2	1/3	1/5
4	0.74	TP	2	2	1/2	2/5
5	0.69	TP	3	2	3/5	3/5
6	0.60	FP	3	3	1/2	3/5
7	0.38	TP	4	3	4/7	4/5
8	0.25	FP	4	4	1/2	4/5
9	0.24	FP	4	5	4/9	4/5
10	0.23	FP	4	6	2/5	4/5

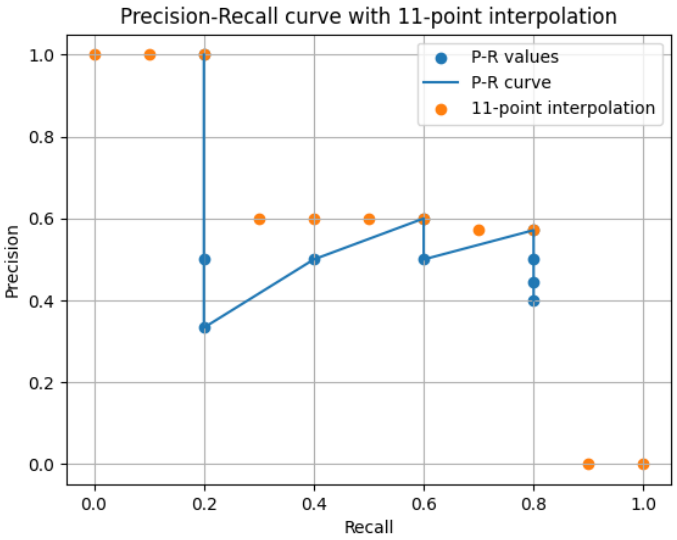


Figure A.1: The Precision-Recall curve of the example data in table A.1. The 11-point interpolation Precision values are shown in orange. We take the average of these 11 values to approximate the area underneath the P-R curve (AP).

Calculating the Average Precision (area underneath the P-R curve)

Preferably, a model has a high Precision for each Recall value, as that would mean that it is very accurate. A model with a large area underneath the P-R curve would be such a model. But as previously described, there is often a trade-off between the two.

It may seem easy to calculate the area underneath the P-R curve. However, this example only contains ten bounding box predictions. Depending on the size of the (testing) dataset, this table could be a lot larger and the resulting P-R curve a lot more complex. Different ways have been devised and used over the years to calculate the area underneath the P-R curve.

A common way to calculate the area under the curve is using 11-point interpolation [12]. With this method, we determine the Precision at eleven different Recall values. The Recall values we consider go from 0 to 1 with increments of 0.1. To determine the Precision at a Recall value, we take the maximum Precision of the P-R curve at that Recall value or to the right of it. The resulting Precision values are shown as orange dots in figure A. Then we take the average of these precision values to get an approximation of the area under the P-R curve, which is the Average Precision.

We emphasise that this is not the only way to calculate the AP. Other ways include using 101 interpolation points to approximate the area under the curve (MS COCO 2014) [8] .

APPENDIX B

YOLOv3 Experiments

The YOLOv8 nano model was used for the object detection model training in this project. However, we also evaluated the performance of a YOLOv3 model when trained using different combinations of real and synthetic training data. The same datasets used to train the YOLOv8 models are used to train the YOLOv3 models. The same real and synthetic training examples are always considered and the resulting models are also always evaluated on the same testing set. The YOLOv3 model's hyperparameters are also set, as much as possible, the same as the parameters we used for the training of the YOLOv8 model. The results are visible in Table B.1.

Remarkably, YOLOv3 scores better or equal when looking at the mAP@[.5] metric. With only 50 real training examples, YOLOv3 already gives a score of around 0.5. YOLOv8 is only around 0.2 in this case. When we use more real examples, the results converge to about the same maximum (0.98). Also, with YOLOv3 we see the benefits of adding synthetic examples to 300 real examples. Similarly, the benefit stops at about 600 synthetic examples. For both YOLOv3 and YOLOv8, the maximum mAP@[.5] of models trained on combined training data seems to be 0.98. The added value of YOLOv8 compared to YOLOv3 is reflected in the mAP[.5:.05:.95] metric. YOLOv3 scores a maximum of around 0.75 on this stricter metric. With YOLOv8, this maximum is a lot higher at 0.80.

Table B.1: Mean average precision of a YOLOv3 model when trained on different combinations of real and synthetic training data.

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=[.5:.05:.95])
12	0	0.0020	0.0006
60	0	0.507	0.232
300	0	0.921	0.601
1500	0	**0.980**	**0.744**
0	300	0.136	0.062
0	600	0.204	0.087
0	1200	**0.259**	**0.118**
300	300	0.942	0.647
300	600	0.951	0.651
300	1200	**0.953**	**0.659**
1500	300	0.979	0.748
1500	600	**0.981**	0.748
1500	1200	0.979	**0.754**

APPENDIX C

Statistical Analysis

To investigate the robustness of the obtained results, some tests were repeated a number of times. In total, the basic tests were performed 5 times. The results of this can be seen in Table C.1.

In all tests we see the same recurring pattern: with 300 real examples, stable results are obtained based on the mAP@[0.5] metric. The average score is 0.863 with a standard deviation of 0.014. Assuming a normal distribution, the 95% confidence interval for the true mean is 0.835 and 0.891. The value we measured Section 4.1.2 is 0.882, which is within this confidence interval. The value observed in this study is therefore not an outlier. After adding 300 Synthetic data, the resulting mAP@[0.5] increases in all cases, on average by 0.046. The associated standard deviation is 0.011. With 95% confidence, it can therefore be stated that adding 300 synthetic data to 300 real data leads to an mAP improvement between 0.024 and 0.068 (mean plus or minus 2 times the standard deviation). Based on the analyses, it can be excluded that adding synthetic data will not lead to an improvement of the mAP@[0.5]. Given the observed outcomes this probability is only 0.002%. We can therefore say that the conclusion, "synthetic data can improve model performance", is robust.

Table C.1: Results of the five tests. For each test two models were trained. The first model was trained using 300 real-world training images, the second model had 300 additional synthetic training images. For each model, the real dataset was randomly sampled from the SPL dataset dataset, described in Section 3.1.1. The synthetic datasets were generated using a different seed for each test.

Test	300 Real mAP@[0.5]	300 Real + 300 Synthetic mAP@[0.5]	Improvement
1	0.882	0.919	0.037
2	0.851	0.915	0.064
3	0.844	0.891	0.047
4	0.870	0.920	0.050
5	0.868	0.900	0.032

APPENDIX D

Results of the Experiments

D.1 Results of Different Percentages of Synthetic Data

The results of tests with different percentages of synthetic data in the training set are shown below.

Table D.1: Mean average precision of a YOLOv8 model when trained using 600 examples. The share of synthetic (and real) data is set to 100%, 75%, 50%, 25%, and 0% for the experiments.

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
600	0	0.946	0.853	0.730
450	150	0.926	0.825	0.696
300	300	0.912	0.776	0.662
150	450	0.841	0.666	0.574
0	600	0.161	0.060	0.074

D.2 Results of Detecting the Football and Robot

Table D.2: Average precision in detecting footballs

Number of real training examples	Number of synthetic training examples	AP (IoU=.5)	AP (IoU=[.5:.05:.95])
12	0	0	0
60	0	0.317	0.123
300	0	0.913	0.709
1500	0	**0.974**	**0.803**
0	300	0.129	0.044
0	600	0.247	0.123
0	1200	**0.312**	**0.163**
300	300	0.935	0.732
300	600	**0.949**	0.722
300	1200	0.945	**0.736**
1500	300	0.979	0.807
1500	600	**0.984**	**0.818**
1500	1200	0.983	0.816

Table D.3: Average precision in detecting NAO robots)

Number of real training examples	Number of synthetic training examples	AP (IoU=.5)	AP (IoU=[.5:.05:.95])
12	0	0.002	0.001
60	0	0.091	0.034
300	0	0.852	0.580
1500	0	**0.979**	**0.762**
0	300	0.050	0.019
0	600	**0.076**	**0.025**
0	1200	0.055	0.021
300	300	0.902	0.632
300	600	0.921	0.638
300	1200	**0.926**	**0.659**
1500	300	0.979	0.768
1500	600	0.979	**0.778**
1500	1200	0.979	0.776

D.3 Randomization Experiments Results

We performed experiments where we kept certain parameters fixed in the 3D environment instead of randomizing them. The randomizers that we disabled were: light, sun angle, skybox, and volume randomizer. For each experiment we use 300 synthetic training examples in combination with 300 real training examples to train a model. The results are shown below in the tables below. We compare only the best result of each fixed setting in Section 4.1.3.

No light randomization

Table D.4: mAP when lighting settings are fixed (temperature: 6500, intensity: 250 lux)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.042	0.006	0.014
300	300	0.923	0.813	0.679

Table D.5: mAP when lighting settings are fixed (temperature: 6500, intensity: 400 lux)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.022	0.003	0.006
300	300	0.922	0.803	0.684

Table D.6: mAP when lighting settings are fixed (temperature: 6500, intensity: 50 lux)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.061	0.007	0.021
300	300	0.921	0.821	0.687

No sun angle randomization

Table D.7: mAP with fixed sun angle (hour: 12, day: 182, latitude: 0)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.034	0.007	0.013
300	300	0.921	0.805	0.677

Table D.8: mAP with fixed sun angle (hour: 24, day: 364, latitude: 90)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.073	0.020	0.030
300	300	0.927	0.817	0.690

Table D.9: mAP with fixed sun angle (hour: 18, day: 273, latitude: 45)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.054	0.009	0.019
300	300	0.920	0.802	0.683

No skybox randomization

Table D.10: mAP with fixed skybox (always use the "3d_camera_skybox" skybox for each image)

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.098	0.024	0.040
300	300	0.919	0.789	0.676

No bloom and lens distortion randomization (volume randomizer)

Table D.11: mAP with fixed bloom and lens distortion.

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.051	0.012	0.020
300	300	0.922	0.805	0.680

No randomizations except for the transforms of objects.

Table D.12: No randomizations except the transforms.

Number of real training examples	Number of synthetic training examples	mAP (IoU=.5)	mAP (IoU=.75)	mAP (IoU=[.5:.05:.95])
0	300	0.008	0.001	0.003
300	300	0.924	0.803	0.683