POLITECNICO DI TORINO

Master’s Degree in Mechatronic Engineering

Master’s Degree Thesis

Exploring a Photorealistic Simulator to Help Data Collection for Real-World Applications

Supervisors
Prof. Alessandro Rizzo
Ing. Simone Panicucci
PhD. Enrico Civitelli
Ing. Luca di Ruscio

Candidate
Edoardo Necchi

Academic Year 2023/2024

Abstract

Introduction: The thesis project focused on exploring NVIDIA Isaac Sim, a software platform developed by NVIDIA to assist developers in designing, simulating, testing, and training AI-based robots and autonomous machines within a physic-based virtual environment.

The primary objective was to create a synthetic dataset using Isaac Sim, aiming to evaluate how this technology can support COMAU in their future projects.

Synthetic data refers to artificially generated data originating from the digital world rather than physical world.

It is particularly valuable when real data is expensive, unavailable, unusable, or subject to bias.

The case study for this project involved the dataset used to develop MI.RA/Depalletizer, a system designed to handle Stock Keeping Units (SKUs) of unknown shape, material, and texture,

This system leverages a combination of AI, 3D cameras and point cloud analysis algorithms to achieve its functionality.

The environment created in NVIDIA Isaac Sim replicated the setup used during the training phase of the real system, incorporating randomizations in box positions, orientations, textures, lighting conditions, and more.

The number of boxes varied from 0 to 20, with a set of textures randomly applied to them.

The position of the lights in the scene were also randomized, creating a variety of scenarios with shadows, glare, and other lighting effects.

The boxes were dropped from a predefined height onto a pallet, resulting in random arrangements that included overlapping boxes, boxes falling off the pallet, and boxes with surfaces that are not parallel to the ground.

The simulation outputs included RGB file, instance segmentation and normal image.

Once the dataset was generated, the following step involved training a convolutional neural network to explore the potential of synthetic data.

Testing: To evaluate the quality of the generated images, a pre-trained Mask R-CNN Model with COCO v1 weights was utilized.

Modifications were made to the source code to align it with the objectives of this thesis.

The segmentation masks and the normal images obtained from the NVIDIA Isaac Sim simulation were used to extract the ground truth targets required for training the neural network.

From the instance segmentation of the scene, it was possible to isolate the mask of each box.

Additionally, the normal images allowed for the separation of each box into its individual faces.

This operation was necessary because the real dataset labelled the faces of the boxes rather than the boxes as whole objects.
After each epoch, the model produced various metrics for both for bounding boxes and segmentation.
However, for this project, the focus was on the metrics related to bounding boxes, with the Intersection over Union (IoU) computed at 75%.

Results and conclusions: The goal of the thesis was to explore the possibilities offered by NVIDIA Isaac Sim and to evaluate possible applications in an industrial environment.
The tests were conducted in two main directions:

- Enhancing a synthetic dataset with real images
- Enhancing a real dataset with synthetic images

In the first test, the performance of a synthetic dataset comprising 250 synthetic images improved significantly when 65 real images were added, achieving a mAP value slightly smaller to that of the benchmark model trained exclusively with real images
This indicated that only a relatively small quantity of high-quality real images was required to achieve notable performance improvements in a model trained exclusively with synthetic images.
The second test involved augmenting a subset of the benchmark real-image training set with synthetic images to assess potential quality improvements.
The results demonstrated that the subset alone produced results comparable to the benchmark, highlighting the high quality of the real images.
However, the addition of synthetic images did not affect performance, suggesting that they did not introduce any bias into the model.
An additional study was carried out at the end of the project, using as benchmark MI.RA/OnePicker, a COMAU system capable of detecting generic objects in a bin and correctly grasping them.
The goal was to extract from NVIDIA Isaac Sim the parameters of the camera used during simulations, to compute the intrinsic matrix and being able to reconstruct the scene in 3D.
Future work should focus on developing domain adaptation techniques to bridge the evident sim-to-real gap observed in these results, as the test using only synthetic images yielded low performance.

These findings served as a foundation for future applications within the COMAU environment, showcasing the potential of using NVIDIA Isaac Sim to generate photorealistic data for training AI-based applications.

Table of Contents

1 Artificial Intelligence 9

1.1 Introduction to Artificial Intelligence 9

1.2 Artificial Neural Network 9

1.3 Loss Function 12

1.3.1 Mean Squared Error 12

1.3.2 Mean Absolute Error 13

1.3.3 Huber Loss 13

1.4 Minimizing the loss function 14

1.5 Overfitting 15

2 Computer Vision and Object Detection 17

2.1 Introduction 17

2.2 Convolutional Neural Networks (CNN) 17

2.2.1 Convolutional layer 18

2.2.2 Pooling layers 18

2.2.3 Fully connected layers 19

2.3 Object detection 19

3 Data 21

3.1 The importance of data in AI 21

3.2 Collection of data 21

3.3 Creation of a dataset 22

4 Synthetic data 23

4.1 General presentation 23

4.2 Unlimited data 23

4.3 Privacy issues 24

4.4 Employment of synthetic data 24

5 NVIDIA Isaac Sim 26

5.1 Introduction 26

5.2 Universal Scene Description (USD) 26

5.3 Digital Twin 27

5.4 Replicator 28

5.5 Python interface 28

6 Project presentation 30

6.1 Introduction 30

6.2 MI.RA / Depalletizer 30

6.3 Dataset Validation 30

6.4 MI.RA / One Picker 31

7 Synthetic dataset 32

7.1 Environment 32

7.1.1 Ground 32

7.1.2 Pallet 32

7.1.3 Disturbance element 32

7.1.4 Lights 33

7.1.5 Camera 33

7.1.6 Boxes 33

7.2 Simulation 35

8 Convolutional Neural Network Model 37

8.1 Mask R-CNN 37

8.1.1 COCO Dataset 37

8.2 Dataset definition 38

8.2.1 Synthetic Dataset Outputs 38

8.2.2 Custom Mask Creation 38

8.2.3 Dataset Preparation and Augmentation 39

8.3 Metrics 39

8.3.1 Precision 39

8.3.2 Recall 40

8.3.3 Precision-Recall Curve 40

8.3.4 F1 Score 40

8.3.5 Average Precision 40

8.3.6 mean Average Precision 41

8.3.7 Intersection over Union (IoU) 41

9 Synthetic dataset evaluation 43

9.1 Introduction 43

9.2 Random textures evaluation 43

9.3 New textures in test set 45

10 Real dataset evaluation 47

10.1 Introduction 47

10.2 Results 49

11 Mixed datasets 50

11.1 Introduction 50

11.2 Synthetic data with real data 51

11.2.1 Synthetic images without real images 51

11.2.2 Synthetic images with 25 real images 52

11.2.3 Synthetic images with 50 real images 54

11.2.4 Synthetic images with 65 real images 55

11.3 Real images with synthetic data 56

11.3.1 Real images 56

11.3.2 Real images and 100 synthetic images 57

11.3.3 Real images and 250 synthetic images 58

12 Discussion 60

12.1 Synthetic data 60

12.2 Real data 62

12.3 Mixed data 63

12.3.1 Synthetic data with real images 63

12.3.2 Real images with synthetic data 65

13 MI.RA/OnePicker Dataset 68

13.1 Introduction 68

13.2 Dataset definition 68

13.3 Environment description 68

13.4 Procedure 70

14 Conclusions 73

Appendix – Images from testing 74

References 79

1 Artificial Intelligence

1.1 Introduction to Artificial Intelligence

Artificial Intelligence (AI) is a technology that allows computers and machine to simulate the behaviour of human intelligence and the capability of problem solving.
AI includes Machine Learning (ML) and Deep Learning (DL), disciplines involving the development of algorithms capable of learning from available data and make predictions or classifications more and more accurate every time, behaving like the human brain in the decision-making processes[1].
Predictions are made with a specific algorithm, which takes data as input and gives an output. The way the parameters of such algorithm are tuned strictly depends on data, necessary to learn the relationship between Input and Output
Machine Learning and Deep Learning algorithm can be divided in:

- Supervised Learning: Input and output data are both available during the training phase, and the goal is to tune the parameters of the algorithm in a way that when new data are available the output will be as accurate as possible. A typical application is for classification, in which a set of input data must be divided into two or more classes.
- Unsupervised Learning: only Input data is available, so the algorithm must learn autonomously the pattern characterizing the data. A common application is clustering, when the algorithm detects relationships among data, grouping them according to their similarities.
- Reinforced Learning: the system continuously interacts with the environment to produce the desired behaviour and receives feedback from the environments itself, to adapt the model.

Other types of learning algorithms can be deployed, but these three are the most common[2].

1.2 Artificial Neural Network

Machine Learning and Deep Learning use, among other algorithms, Artificial Neural Networks (ANN) to learn the relationship between input and output data.
An Artificial Neural Network is a structure making decision like the human brain, using processes that mimic the way biological neurons work together to identify phenomena, situations and get to a conclusion.

A neuron (Figure 1) is the fundamental building block of the central nervous system and consists of three primary components, along with other elements:

- Soma, the nucleus of the neuron.
- Dendrites, treelike structures responsible for receiving messages from other neurons.
- Axon, structures responsible for the link with the dendrites of other neurons, sending them messages via electric signal passing through synapses.

Every information coming from other neurons is gathered into neuron's cell body where is processed and sent to axon terminals via electrochemical processes. [2,3]

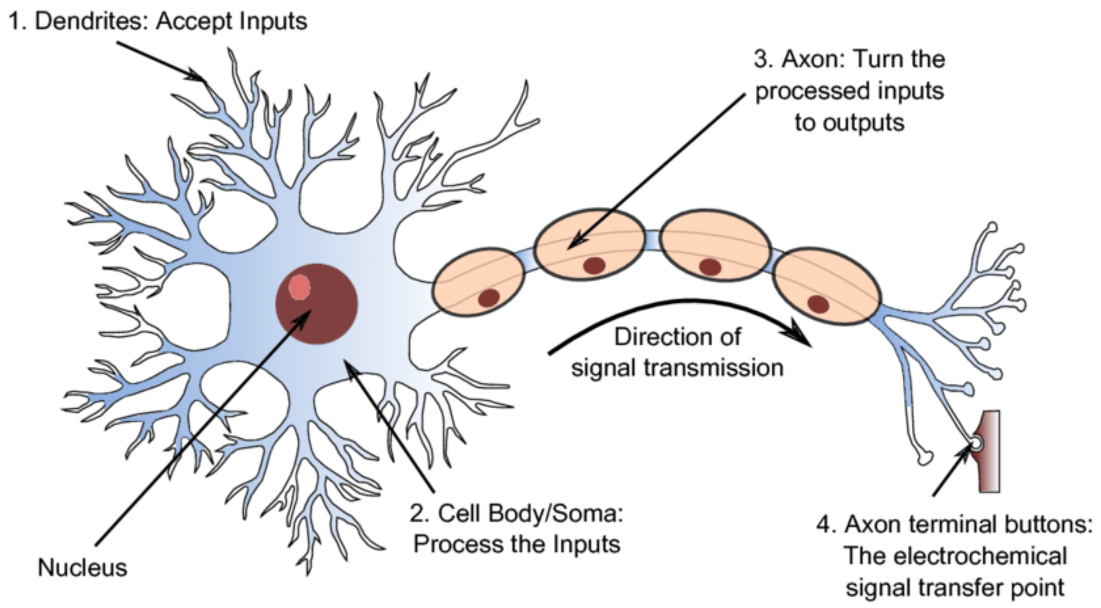

1. Dendrites: Accept Inputs
3. Axon: Turn the processed inputs to outputs
Direction of signal transmission
Nucleus
2. Cell Body/Soma: Process the Inputs
4. Axon terminal buttons: The electrochemical signal transfer point

Figure 1 - Representation of a human neuron[4]

The neuron can be artificially modelled with the perceptron[5] (Figure 2), a model composed of several inputs in which a weighted sum is performed to produce an output.

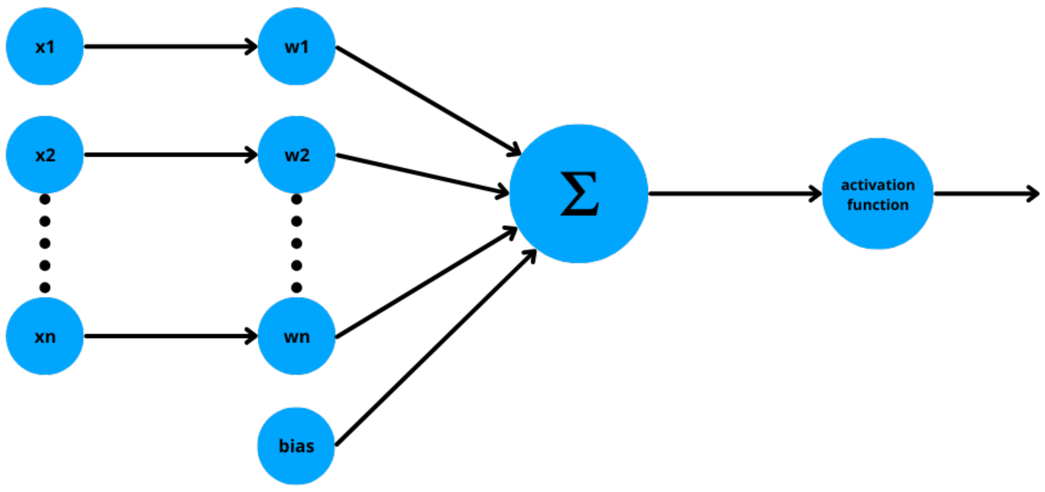


Figure 2 - Representation of a perceptron

The weights define the importance of a single variable, with the higher ones contributing to the output more than the smaller ones, while the bias term adds a bias to the model.

The activation function makes the model nonlinear, deciding whether the perceptron should produce an output or not.

Most common activation functions are:

- Sigmoid: $\mathrm{f(z)} = \frac{1}{1+\mathrm{e}^{-\mathrm{z}}} = \frac{\mathrm{e}^{\mathrm{z}}}{1+\mathrm{e}^{\mathrm{z}}}$
- Hyperbolic tangent (tanh): $\mathrm{f(z)} = \tanh(\mathrm{z}) = \frac{\mathrm{e}^{\mathrm{z}}-\mathrm{e}^{-\mathrm{z}}}{\mathrm{e}^{\mathrm{z}}+\mathrm{e}^{-\mathrm{z}}}$
- Rectified linear unit (ReLU): $\mathrm{f(z)} = \max\,(0, \mathrm{z})$

Activation functions are non-linear because linear activation functions do not enhance the capabilities of the model.

A linear activation function would preserve the overall linearity of the model, regardless of the number of nodes.

An ANN is constituted by many layers (Figure 3), divided in:

- input layer.
- one or more hidden layers.
- output layer, with the output of each layer used as input by the next one.

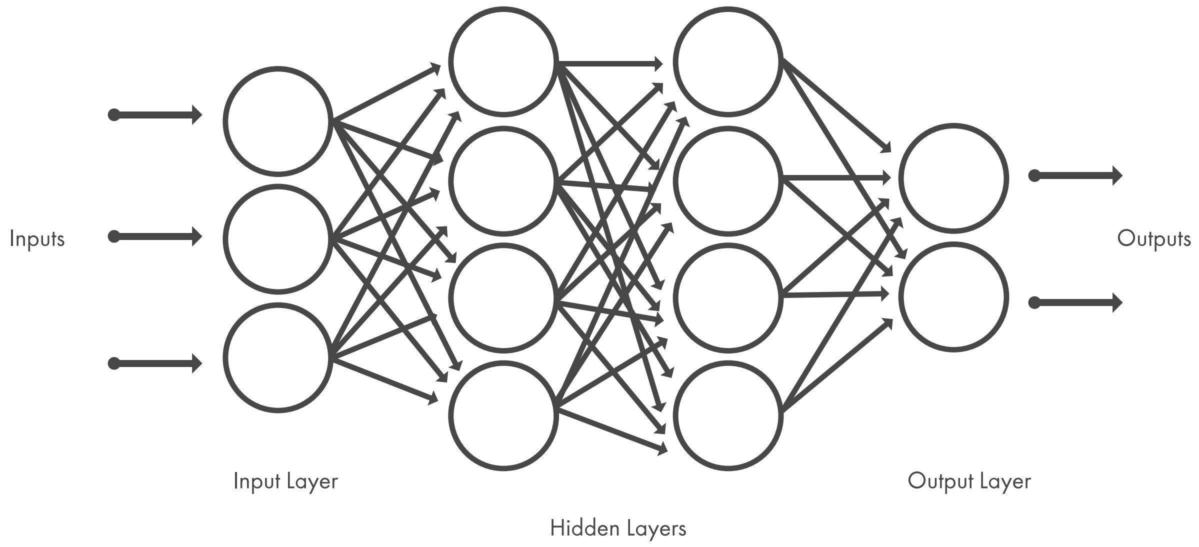


Figure 3 - Representation of a Neural Network model[6]

Each layer can contain an arbitrary number of nodes or perceptron, even higher than the number of actual inputs, helping to improve the distribution and, consequently, the overall performance.

The number of hidden layers typically depends on how complex the relation between input and output is.

There is no indication on how many layers are needed in an ANN since a higher number of layers guarantees high performances, but, at the same time, high complexity. [2,3]
The training process consists in learning from the labelled data several parameters to optimize the network.

1.3 Loss Function

During the training process, when the Artificial Neural Network returns an output a loss function is deployed.
Loss function is a method that computes the error between the predicted output and the real output, also known as ground truth, to quantify how well the algorithm models the dataset, and, consequently, how to modify the weights to have a better result.
The error is propagated in backward direction through every layer to update the weights to minimize the error.
Training a neural network means finding the correct weights and biases terms that minimize the loss function under a certain threshold.
The choice of loss function typically depends on the architecture of the model and the specific task at hand.
There are many loss functions, like Mean Squared Error, Mean Absolute Error, Huber Loss, and others[7].

1.3.1 Mean Squared Error

Mean Squared Error (MSE) measures the average of the square differences between the predicted valued and the true values. Its mathematical definition is:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Where n is the number of samples, y_i is the true value of the i^{th} sample and $\hat{y}_i$ is the predicted value of the i^{th} sample.
MSE is a non-negative function, with a value of 0 indicating a perfect fit. Higher values correspond to larger discrepancies between predictions and actual values.
Being a quadratic function, MSE is sensitive to outliers, which means it places more emphasis on reducing larger errors than smaller ones.

1.3.2 Mean Absolute Error

Mean Absolute Error (MAE) measures the average of the absolute differences between the predicted value and true value.

The definition of Mean Absolute Error is:

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |y_i - \widehat{y_i}|$$

Where n is the number of samples, y_i is the true value of the i^{th} sample and $\hat{y}_i$ is the predicted value of the i^{th} sample.

Like MSE, MAE is non-negative, with its minimum in 0, while higher values are indices of discrepancies between ground truth and predictions.

MAE is a linear function, meaning that treats all errors equally, regardless of their magnitude, making it robust to outliers, as it does not disproportionately emphasize large errors.

This property makes MAE suitable for applications where the presence of outliers is expected, or the distribution of errors is not symmetric.

Being non-differentiable, the optimization processes relying on gradient-based techniques can have difficulties, but subgradient methods can overcome the problem.

1.3.3 Huber Loss

The Huber Loss combines the properties of both Mean Squared Error (MSE) and Mean Absolute Error (MAE).

Huber Loss is designed to be more robust to outliers than MSE, while maintaining smoothness and differentiability.

The Huber Loss function is defined as:

$$Huber = \begin{cases} \frac{1}{n}\sum_{i=1}^{n} \frac{1}{2}(y_i - \widehat{y_i})^2, & |y_i - \widehat{y_i}| \le \delta \\ \frac{1}{n}\sum_{i=1}^{n} \delta(|y_i - \widehat{y_i}| - \frac{1}{2}\delta), & |y_i - \widehat{y_i}| > \delta \end{cases},$$

Where n is the number of samples, y_i is the true value of the i^{th} sample, $\hat{y}_i$ is the predicted value of the i^{th} sample and δ is a user-specified threshold value.

When the error is small, the Huber Loss function behaves like the MSE loss function, while when the error is large, the Huber Loss function behaves like the MAE loss function.

This property makes the function more robust to outliers, as it is less sensitive to large errors.

The value of the threshold δ can be chosen empirically by a trial-and-error procedure, trying different values, and evaluating the corresponding performance.

In practice:

- A small value of δ is typically chosen when the dataset contains a significant amount of noise.
- A larger δ is preferred when the dataset includes outliers[7].

A comparison of the behaviours of MSE, MAE and Huber Loss functions is represented in Figure 4.

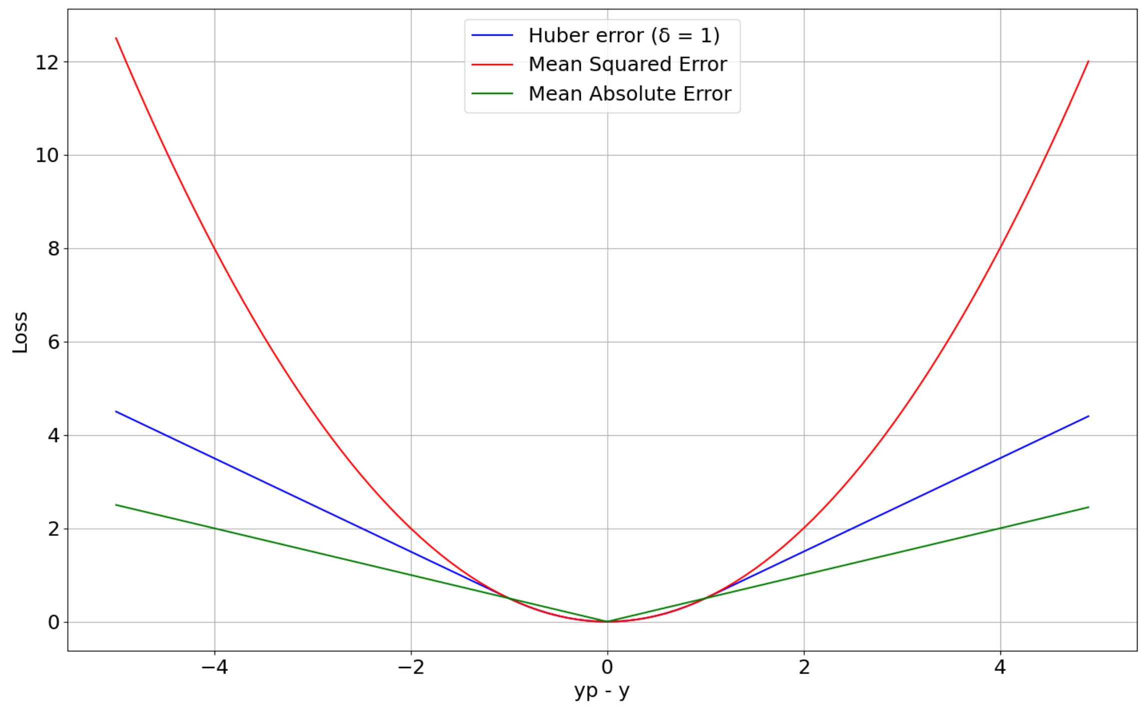


Figure 4 - Comparison of Mean Squared Error, Mean Absolute Error, and Huber Loss

1.4 Minimizing the loss function

One of the most common algorithms for minimizing a loss function is gradient descent, where at every iteration the weights and the bias terms are updated as follows:

- $W_{new} = W_{previous} - \alpha \frac{\partial J(W,b)}{\partial W_{prev}}$
- $b_{new} = b_{previous} - \alpha \frac{\partial J(W,b)}{\partial b_{prev}}$

The parameter α is called “learning rate”, a tuneable value that defines the size of the steps that are taken to reach the minimum.

The loss function is minimized by updating the parameters in the opposite direction of the gradient of the objective function $J(W, b)$ with respect to the parameters.

A high learning rate results in large steps, with the risk of overshooting the minimum, while a small learning rate increases the number of iterations, comprising the overall efficiency[89], as shown in Figure 5.

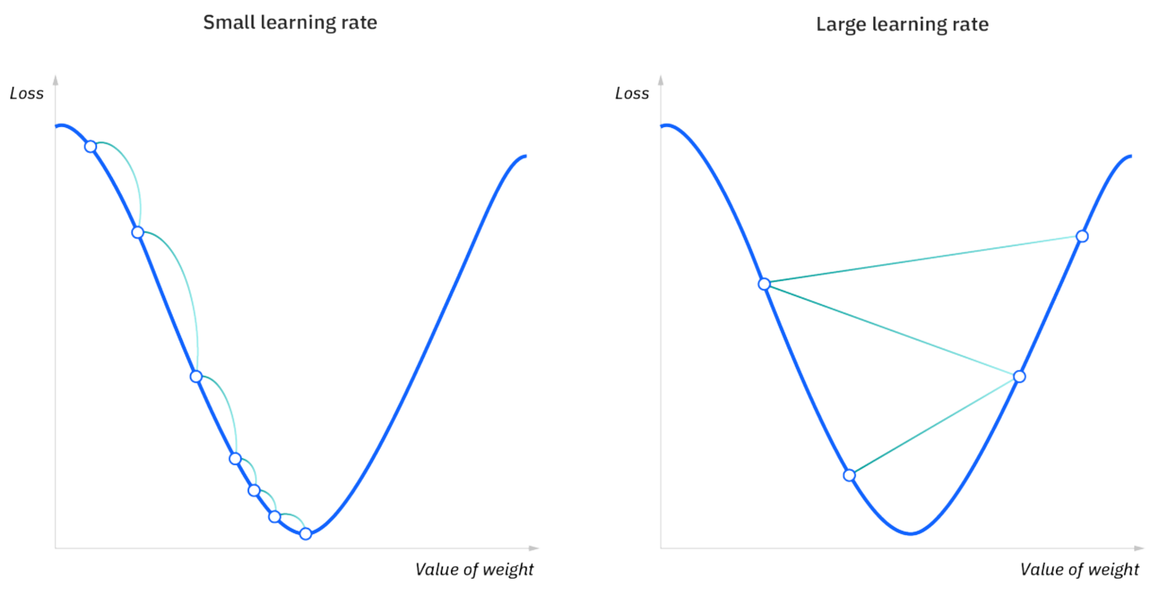


Figure 5 - Representation of how the size of learning rate can affect the performance of gradient descent method, leading to the minimum (left image) or to an overshoot (right image)[10]

1.5 Overfitting

An ANN is trained to make predictions on new data, with the goal of maximising the predictive accuracy on unseen data rather than on training data.

However, focusing excessively on fitting the training data can lead the model to memorize specific details of the dataset rather than identifying general predictive patterns.

This issue is known as overfitting, and it results in poor accuracy when predicting outcomes for unseen data.

Several factors can contribute to overfitting, including:

- Insufficient training data: A small training dataset may not adequately represent the diversity of input cases.
- Data noise: The model may incorrectly learn noise or anomalies as significant patterns.
- Excessive training epochs: Prolonged training on the same dataset can lead to over-specialization on the training data.
- Model complexity: An overly complex model may attempt to fit the data too precisely, capturing irrelevant details.

To mitigate overfitting, a variety of techniques can be applied:

- Early stopping: Halting training before the model begins to overfit the data. This method requires careful calibration, as stopping too early may lead to underfitting, where the model fails to perform optimally, due to insufficient training.
- Dataset expansion: Increasing the dataset size helps ensure a broader range of inputs is covered during training, reducing the risk of the model learning dataset-specific quirks.
- Feature selection: Identifying and retaining only the most relevant features while discarding irrelevant or redundant ones makes the data clearer and more concise.
- Regularization: Adding penalties to model parameters discourages reliance on less critical features, thereby promoting simplicity and generalizability[11].

There are just few of the many techniques available to reduce the risk of overfitting.
As illustrated in Figure 6, the goal is to strike an optimal balance between underfitting and overfitting, achieving the best possible performance on unseen data.

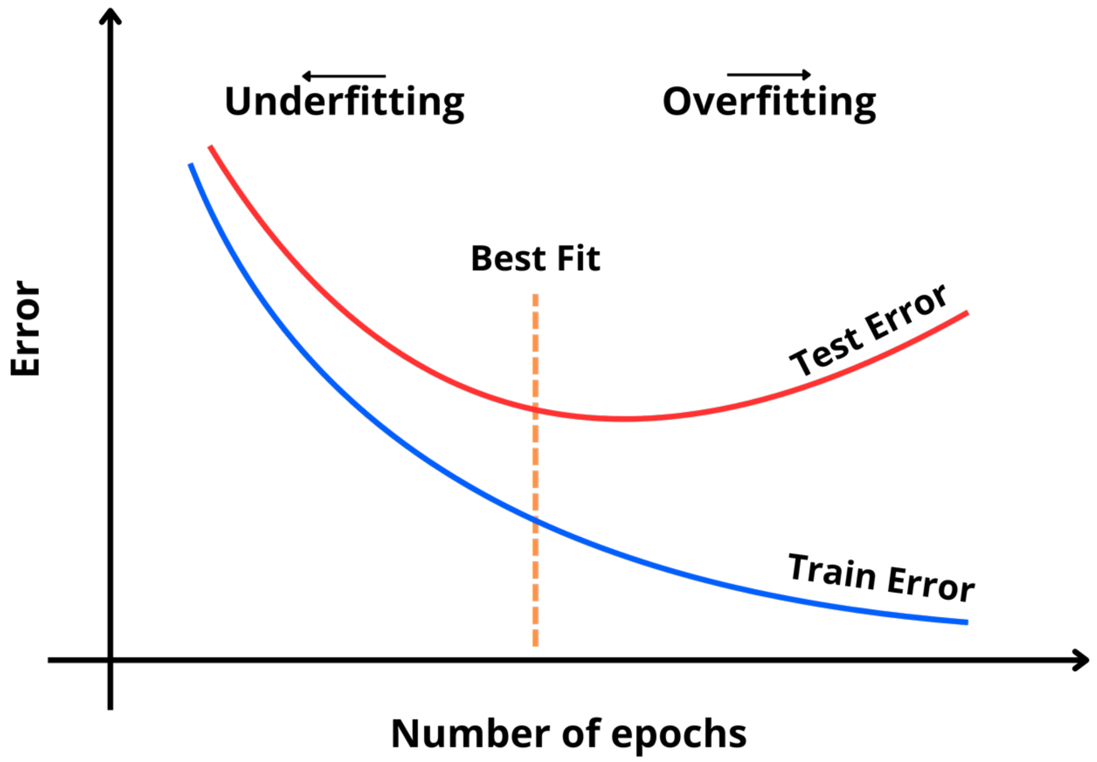


Figure 6 - Underfitting and overfitting represented with respect to training and test set

2 Computer Vision and Object Detection

2.1 Introduction

The focus of the thesis is the creation of a synthetic dataset of images to be used in industrial visual applications.

Computer Vision is a field in AI whose aim is to interpret image content, striving to create systems with the same perceptual capabilities of human visual system.

Human brain can easily interpret an image in milliseconds, discriminating between thousands of categories and rapidly find objects in a scene.

Computer Vision address various tasks, including:

- Image classification: identifying and categorizing images, e.g., pictures uploaded online.
- Objects detection: detecting distinct objects in an image, e.g., faces, numbers, etc.
- Semantic/Instance/Panoptic segmentation: processing the image at pixel level, annotating pixels as belonging to a specific class or instance.

These are just few examples of the possibilities of Computer Vision, a field which is continuously improving and advancing.

2.2 Convolutional Neural Networks (CNN)

The ANN architectures used in the field of Computer Vision are mainly Convolutional Neural Networks (CNN).

A CNN (Figure 7) is composed of neurons that receive an input and performing an operation to give an output, self-optimizing through learning in an analogous way to ANN.

The key difference with ANN is that the neurons are organized in three dimensions:

- Height
- Width
- Depth

Depth does not refer to the total number of layers of the ANN, but the third dimension of an activation volume (e.g., in case of an RGB image depth will be 3, corresponding to one channel for each colour).

The most common building blocks of a CNN are:

- Convolutional layer, like a feature extractor.
- Pooling layer, acting as a dimensionality reductor.
- Fully connected layer, like a classification layer.

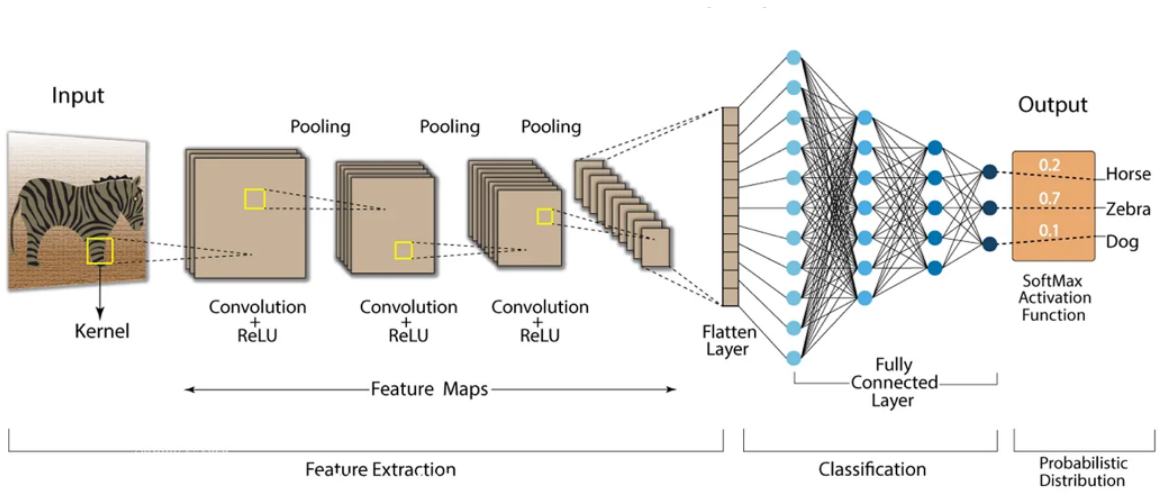


Figure 7 - CNN Scheme[12]

2.2.1 Convolutional layer

A convolutional layer uses a convolutional kernel as a filter for the input, that could be many and various.
During each forward pass, a filter slides across the input volume and computes the activation map of the filter at that point by computing the pointwise product of each value and adding these to obtain the activation at the point.
The linear operator implementing this type of function is the convolution, which can be written as a dot product.
During training, the network learns filters that can capture different kind of visual information such as an edge, orientation, and eventually, entire patterns.

2.2.2 Pooling layers

Pooling layers are responsible for the reduction of the spatial dimension of feature map, without losing essential information.

The operation carried out by pooling layer consists in combining the output of neuron clusters at one layer into a single neuron in the next layer.
There are two types of pooling:

- Max pooling, reporting the maximal values in each rectangular neighbourhood of each point (i, j).
- Average pooling, reporting the average value.

2.2.3 Fully connected layers

Fully connected layers are used as final layers in classification problems, where a few of them are attached on top of a CNN.
The structure of a fully connected layer resembles the one of traditional ANN, in which every neuron in one layer is connected to every neuron in the next one.

2.3 Object detection

This thesis, as it will be explained in Chapter 6, uses as benchmark a system developed by COMAU for the recognition of boxes on a pallet in an industrial environment.
The focus will be on Object Detection, which is the primary objective for this purpose.
Object Detection is a Computer Vision task whose goal is to locate objects within an image and classifying them according to a specific semantic.
This involves determining the specific location while simultaneously identifying the class to which they belong.
During the testing phase of the dataset (Chapter 11) a CNN model for object detection was be employed. In addition to bounding boxes, the chosen model also performs instance segmentation, which involves classifying the image on a pixel-by-pixel basis, providing a more precise output compared to a bounding box.
Instance segmentation distinguishes between individual instances of the same class labels, labelling each object separately.
This contrasts with semantic segmentation, which labels objects as part of a general class, without differentiating between individual instances (Figure 8).

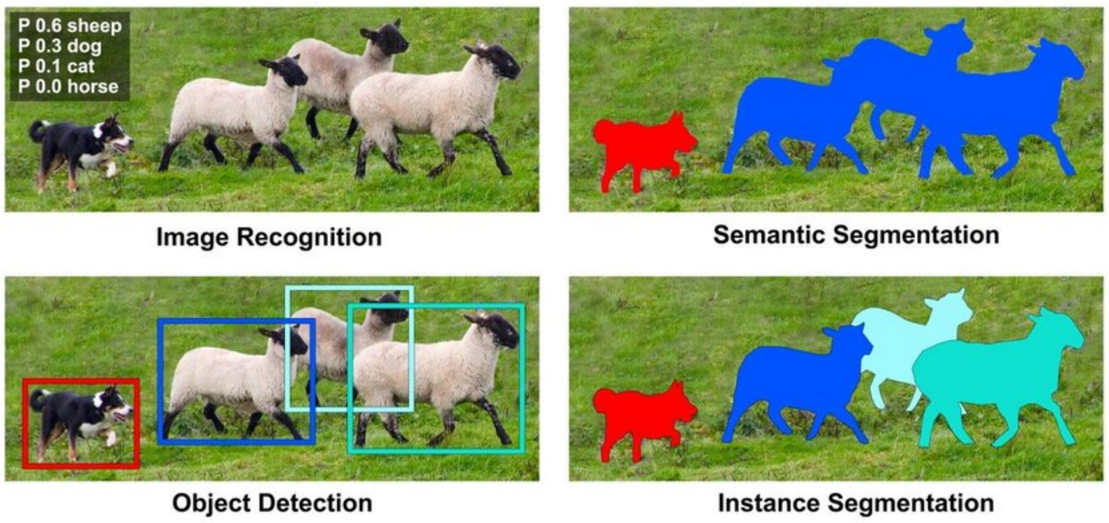


Figure 8 - Example of object detection, semantic segmentation, and instance segmentation[13]

During the dataset definition, specific areas of the image are labelled as a region with distinct pixel-level features, enabling the model to recognize similar features in new input images.

The idea is that the model does not directly recognize an object itself, but rather aggregates similar features (e.g., size, shape, colour, etc.) to classify the object within a specific region of the image[14].

3 Data

3.1 The importance of data in AI

The predictions made by Deep Learning and Machine Learning algorithms rely on vast amount of data from large and diverse datasets.
The quantity and, especially, the quality of the data influence the performances of the models. The quality of the data can significantly bias the accuracy of a model, as unclear and inaccurate data can distort the results produced by the AI model.
Data used in AI training must reflect real-world scenarios, as a model may yield poor results if the environment in which is working differs significantly from the domain of training data or if the data is incomplete or missing.
Various types of bias, especially noise, can corrupt data, with the consequence of having a model which cannot display meaningful results.

3.2 Collection of data

The collection of data for training AI models is a crucial process, as it directly impacts the performance of the algorithm.
The data must be organized in a way that allows the creation of a powerful model that meets the required specifications.
There are several data collection techniques, such as:

- Public datasets: widely used for various AI applications, such as computer vision and speech recognition. These datasets offer unlimited access, use, and modification.
- In-house data collection: data is directly collected by developers, allowing for a complete control over quality and quantity. However, this method can be time-consuming, costly and requires expertise in data gathering techniques.
- Synthetic datasets: synthetic data refers to artificially generated data that mimic the diversity, patterns, and complexity of real datasets. Such data can be created with software or algorithms.

Data collection is a critical and initial step in the development of AI models.

The quality, quantity, and relevance of the data used during the training and validation phases significantly influence their performances of these models, generalization capabilities, and real-world applicability.

3.3 Creation of a dataset

A set of data used in AI training algorithms is typically divided into three subsets:

- Training set
- Validation set
- Test set

The training set is used to build the model by adjusting multiple parameters based on the method chosen by the designer.
It constitutes the largest portion of the original set, typically around 70%.
The validation set, which contains from 15% to 20% of the original data, is distinct from the training set.
It provides an unbiased evaluation of intermediate models fitted on training data, enabling fine-tuning to achieve the desired performance for the specific task.
The test set comprises the remaining data that has never been used during training or validation.
Its purpose is to offer an unbiased evaluation of the final model, verifying the results obtained during the training-validation phase without further parameter adjustments.
This separation of data into training, validation and test sets is essential to avoid training and testing on the same data, and to mitigate the risk of overfitting, ensuring that the model generalizes well to new, unseen data.

4 Synthetic data

4.1 General presentation

Synthetic data is a class of data artificially generated, coming from the digital world rather than real world.
Starting from the assumption that real data is for sure the best source of information, they can often be expensive, unavailable, unusable, or biased.
The role of synthetic data is to help developers in these situations, acting as a supplement or, in some cases, alternative to real data.
A report from Gartner[15] shows that by 2030 more than 70% of data used in training AI models will be synthetic.
Synthetic data can be divided into two types:

1. Synthesized from real datasets: this type of data is generated by building a model that captures the structure and distribution of real datasets. Synthetic data is then sampled or generated from this model.
2. Not synthesized from real datasets: this data is generated without relying on real datasets but starting from existing model or by background knowledges of the analysts. These models can include processes or simulations, which may be created using tools such as gaming engines or simulations software[16].

4.2 Unlimited data

The main advantage of synthetic datasets is the ability of generating unlimited amount of labelled data, which can deliberately include rare but crucial corner cases.
Analysts, especially at the beginning of a project, often use open-source datasets as a starting point.
However, these datasets may not align well with the specific problem or model objectives.
In some cases, datasets may be unlabelled, and the process of gathering and labelling datasets - ranging from few thousand to millions of data points - can be consuming and, particularly, very costly.
In the context, for example, of manufacturing and distribution environments, the goal is to create automated systems capable of performing increasingly complex tasks.

The difficulty in this type of scenarios is the need of a various training dataset covering multiple anticipated scenarios, as well as uncommon ones.
An example could be a vision system designed to recognize objects on an industrial conveyor. The required dataset must include images captured under varying light conditions, featuring different objects with diverse textures, positions, orientations, etc.
Manually creating such a dataset could be trivial, time consuming and, in particular, very expensive, requiring significant human resources.[17]
A Deloitte analysis stated that the issues related to data access are ranked in the top three challenges faced by companies when implementing an AI project[18], while a survey from MIT Technology Review reported that almost half of the respondents identified data availability as a constraint to the use and development of AI within their company[19].

4.3 Privacy issues

Many problems in availability of data are related to privacy issues, as individuals are increasingly aware of how their data is used and shared, while privacy laws are becoming stricter.
Contemporary privacy regulations, like General Data Protection Regulation (GDPR) in Europe, impose restrictions on the usage of personal data for a secondary purpose – i.e., purposes other than those for which the data was originally collected. These regulations often require obtaining additional consent or authorization, leading to possible biases, as those who provide consent may differ from those who do not in important characteristics.[20]
For example, financial services often rely on sensitive customer data for their internal operations, but privacy regulations can complicate data collection and usage.
Synthetic data can solve this issue, as they do not contain sensitive information.

4.4 Employment of synthetic data

The employment of synthetic data can serve various purposes, such as validating analysts' assumptions and demonstrating the potential results that can be obtained with the model they developed.
By presenting or analysing results obtained with synthetic data, it becomes possible to decide whether to undergo the complex process of acquiring real data or, on the contrary, avoid extra effort for obtaining real data if the results are not as expected.

At the beginning of a project, real data may be unavailable.
In such cases, synthetic data can act as the “first brick” in the training process, enabling the development of an initial model that can later be refined using real data.
In extreme cases, where real data do not exist – such as with new models or when creating a real dataset is time consuming and costly – synthetic data offer a solution.
It can be used to simulate edge or rare case that are difficult, impractical, or unethical to collect in real world.
For example, missing data might involve specific corner cases, that can be covered with synthetic data, leading to a more robust model.[17]
Using both synthetic and real data in the dataset creation process can help reduce data gathering costs while maintaining high-quality in results.
A study of 2020, for instance, demonstrated that a computer vision system trained with only 10% of real images within a synthetic dataset achieved performances comparable to those of a system trained entirely on real data in recognizing planes in satellite images.[21]

5 NVIDIA Isaac Sim

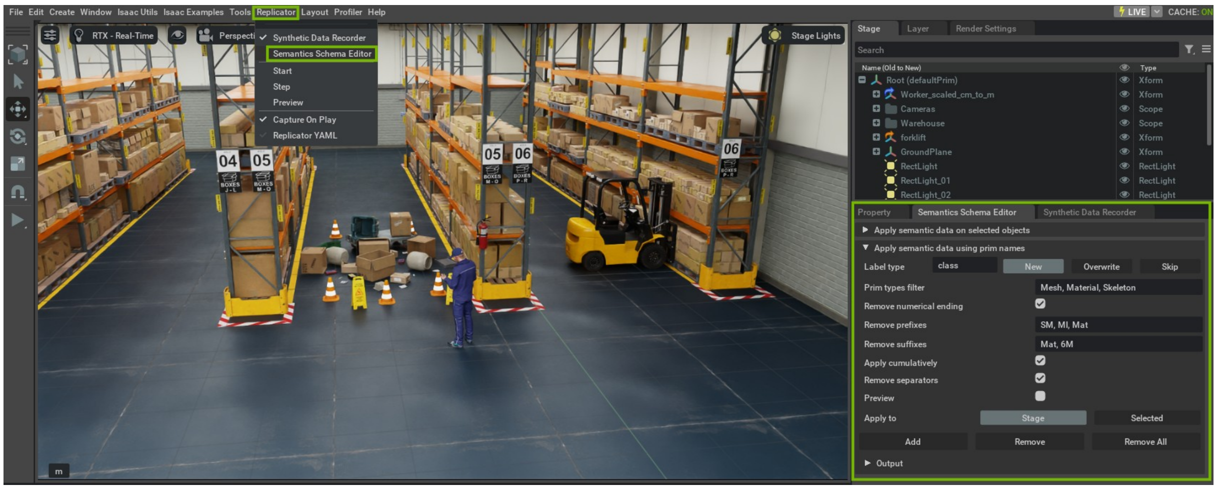
5.1 Introduction

The aim of the thesis is to explore the applicability of a photorealistic image simulator for generating synthetic images.

NVIDIA Isaac Sim^{TM} is a software platform from NVIDIA which is designed to assist developers in designing, simulating, testing, and training AI-based robots and autonomous machines within a physically accurate virtual environment[22].

The simulator is built on NVIDIA Omniverse^{TM}, a modular development platform that provides APIs for creating 3D applications.

Powered by Universal Scene Description (OpenUSD) and Nvidia RTX^{TM}, Omniverse enables the creation of physically accurate world-scale simulations, a digital twin of the real environment taken into consideration[23]. (Figure 9)



Label type: class
Prim types filter: Mesh, Material, Skeleton
Remove numerical ending: ✓
Remove prefixes: SM, MI, Mat
Remove suffixes: Mat, 6M
Apply cumulatively: ✓
Remove separators: ✓
Preview:
Apply to: [✓ Stage] Selected

Figure 9 - NVIDIA Isaac Sim Graphical User Interface, with a highlight (in green) on the interface for the semantic annotation of objects in the scene[22]

Thanks to Nvidia PhysX^{TM}, the physics engine of Omniverse, it is possible to reproduce the behaviour and the physics reaction of rigid bodies inside the simulated environment.[24]

5.2 Universal Scene Description (USD)

Isaac Sim includes a collection of workflows that allows designers to import and tune mechanical systems created in various formats (like Onshape[25], URDF[26], MJCF[27], etc…), thanks to the use of Universal Scene Description (USD), a technology developed by Pixar Animation Studio.[28]

USD is the first publicly available software that enables the robust and scalable interchange and enhancement of arbitrary 3D scenes, which may be composed of assets coming from different software or application.

The power of USD lies in its ability to transmit virtual assets among applications without loss of information, allowing the designers to edit them with a single, consistent API within a unified scenegraph.[29]

5.3 Digital Twin

Digital twin are physically accurate virtual replicas of assets, processes, and environments, incorporating true-to-reality physics, materials, lights, and behaviour.

Companies use digital twin to simulate real-world situations, gather data and information, and test new technologies before implementing them in real world.

For example, Amazon Robotics trained and tested their robots in a controlled environment prior to real-world deployment.

Engineers were able to generate large, photorealistic synthetic datasets, which enhanced the capabilities of Proteus robot's (Figure 10) autonomous driving system.

This system, based on fiducial markers for location detection, benefited significantly from these synthetic datasets [30].

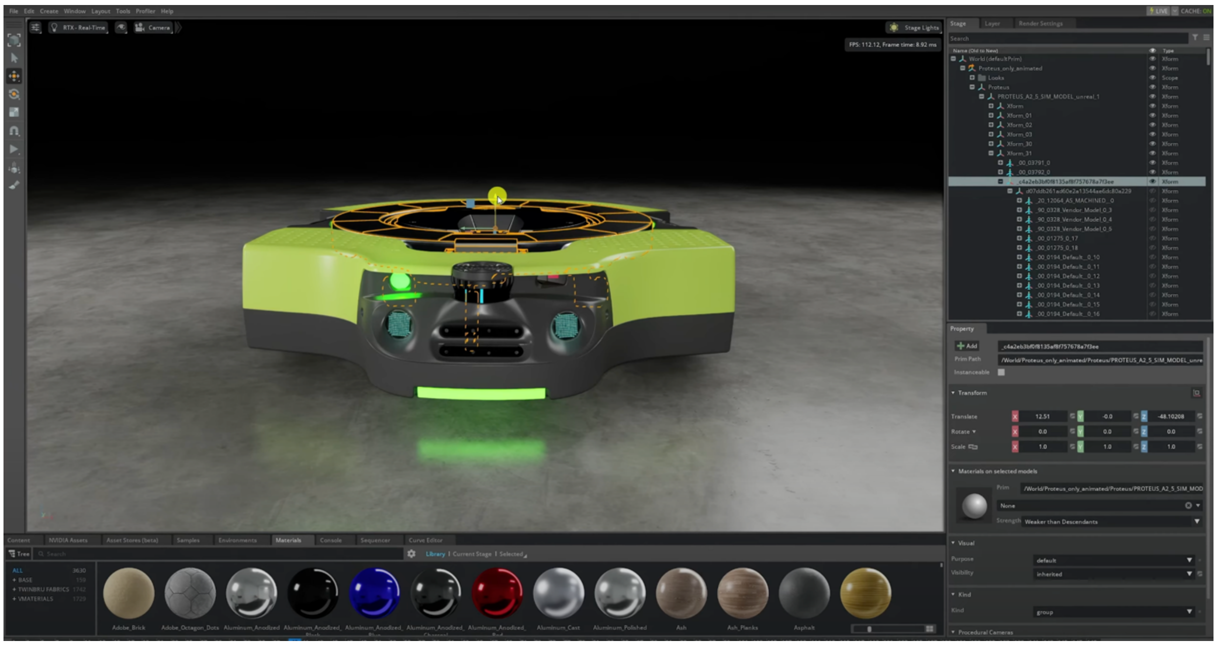


Figure 10 - Digital Twin of the Proteus Robot during a simulation in NVIDIA Isaac Sim[31]

5.4 Replicator

Replicator is an Omniverse extension that provides various methods for generating physically accurate synthetic data.
It features an extensible registry of annotators and writers to address custom requirements for annotations and outputs format needed for training AI models[32].
The randomization of infinitely aspects of the scene, like object position, orientation, texture, material, size, etc…, enables the creation of large datasets.
This enhances the training of neural network models by exposing it to a wide range of parameters that improve generalization for real-world applications.
The main advantage of Replicator is the possibility to set parameters through either a Graphical User Interface (GUI) or the Python script editor.
Nearly all scene parameters can be modified using different distributions, such as Gaussian, linear, conditional, etc.
For example, a CNN whose aim is object detection needs a dataset with the target object placed in different positions and environments, sometimes partially visible and occluded by other assets in the scene.
The position of the camera itself can also be randomized according to specifications, capturing different backgrounds, creating a wide dataset with different characteristics.
The outputs of the simulations are wide, including:

- RGB
- Grayscale
- Depth
- Semantic Segmentation
- Instance segmentation
- Pointcloud

This flexibility allows the generation of a variety of output data types and annotations[33].

5.5 Python interface

NVIDIA Isaac Sim provides a Python script interface, where every action in the Graphical User Interface (GUI) has a corresponding Python script.
However, the reverse is not true, as the script editor offers a wider range of commands than those available through the GUI.

The project was developed using Python scripting, as many of the Replicator commands used were not accessible via the GUI and were instead implemented using the *omni.replicator.core* library.

In Algorithm 1 (shown below), a cube is created with the following characteristic:

- Position: uniform distribution between (0,0,0) m and (10,10,10) m
- Scale: each side is 2 m
- Rotation: 45° around x-axis and 45° around y-axis
- Semantics: useful for labelling, in this case simply annotated as “cube”

```python
import omni.replicator.core as rep
cube = rep.create.cube(
    position=rep.distribution.uniform((0,0,0), (10, 10, 10)),
    scale=2,
    rotation=(45, 45, 0),
    semantics=[("class", "cube")],)
```

Algorithm 1 – Python script using omni.replicator.core APIs to insert in a scene a cube[34]

6 Project presentation

6.1 Introduction

The project primarily focuses on evaluating the performances of synthetic data, exploring potential applications and limitations, and determining how this technology can benefit COMAU in their future projects.
The project is divided into three main phases:

- Dataset Creation: Utilizing NVIDIA Isaac Sim to generate a dataset based on specific case study (MI.RA / Depalletizer system).
- Dataset Validation: Using a pre-trained Convolutional Neural Network (CNN) and fine-tuning it to adapt to its function.
- New Dataset Development: Creating a dataset for a different application, with the MI.RA / OnePicker as the case study.

6.2 MI.RA / Depalletizer

The case study for the first part of the project was the dataset used for developing the MI.RA/Depalletizer, a system in the MI.RA (Machine Inspection and Recognition Archetypes) flexible vision systems portfolio of COMAU.
Depalletizer system handles Stock Keeping Units (SKUs) of unknown shape, material, and texture, by leveraging on the combination of AI, 3D cameras and pointcloud analysis algorithms.
The dataset used during the training phase of this algorithm was not available, meaning that all the pictures used were taken and labelled manually.
The system is basically an object detector, recognising boxes on a pallet, choosing via a pointcloud analysis the best point for picking the box and move it around[35].

6.3 Dataset Validation

The evaluation of the dataset for the case under study was divided into three parts:

- Training and testing of the synthetic dataset only, to gain experience with Synthetic Data Generation (SDG) process and with the model chosen.

- Training and testing of the real dataset, to have a benchmark for the metrics.
- Training and testing of a mixture of the two, to discuss the improvements that synthetic data could bring to a real dataset

6.4 MI.RA / One Picker

The last part of the thesis focused on the creation of a dataset resembling the characteristic of the one used for the MI.RA / One Picker training process.
This solution is a picking system capable of emptying bins full of heterogeneous objects detected and localized by a vision system which interprets information coming from RGB and depth camera.
The position of the camera can change, being both stationary (e.g., with fixing picking areas) or on-robot (e.g., multiple bin applications).
The system can accurately determine the best picking pose for various objects, without requiring extra information, and determine the most effective way to empty the bin[36].

7 Synthetic dataset

7.1 Environment

The first step involved the creation of a realistic environment where images could be captured. The objective was to design an environment that mirrored the characteristics of the setting in which the real images were taken, making it easier to compare the quality of the dataset.
All the assets used in the environment were sourced either from Omniverse Nucleus, the database and collaboration engine of Omniverse, or simply instantiated using Replicator APIs. Figure 11 and Figure 12 show an example of the images created

7.1.1 Ground

The ground plane in the scene consisted in a physics-enabled ground plane with an industrial texture applied.

7.1.2 Pallet

A single pallet model, obtained from the Omniverse Nucleus database, was used across all simulations.
Its position remained fixed throughout the iterations, although it may have disappeared in some iterations.
Four invisible planes surrounded the pallet, forming a funnel-like structure.
These planes served as a dual function:

- Keeping the boxes confined to the pallet area.
- Enhancing the randomness in box arrangement when they lean against the plane.

7.1.3 Disturbance element

Disturbance elements were introduced to make the scene more realistic by adding randomness, replicating the conditions of a real-world industrial environment.
The system was designed so that the CNN recognized only the boxes and the pallet, avoiding misidentification of disturbance elements as an object of interest.

In the specific case of this dataset, the disturbance element was a metallic table, which it may appear visibly at the edges of the images or cast shadows on the boxes or nearby areas due to the lighting.

7.1.4 Lights

Various lighting types were used to illuminate the scene and make it more realistic:

- Linear lights: Provided general illumination, creating areas of lights and shadows in the environment.
- Flash lighting: Acted as a disturbance element, producing a high-intensity light beam in certain areas.

At each iteration, the following parameters were modified:

- Intensity
- Pose
- Visibility (whether the source of light is present in the scene)
- Colour

7.1.5 Camera

The camera was positioned directly above the pallet, focusing on the centre of the scene.
Coordinates:

- x-axis: fixed at 0.
- y-axis: fixed at 0.
- z-axis: Variable, with a uniform distribution between 2.8 m and 3.5m.

Orientation:

- changed randomly to simulate real-world scenarios where the pallet may not be perfectly centred in the image frame.

7.1.6 Boxes

The boxes were cubes with dimensions following a normal distribution, ranging from 0.1 m to 0.4 m on each side.
Initially, the box textures were highly colourful and featured unusual patterns, while later in the project more standard textures were used, resembling real-world industrial paper boxes.

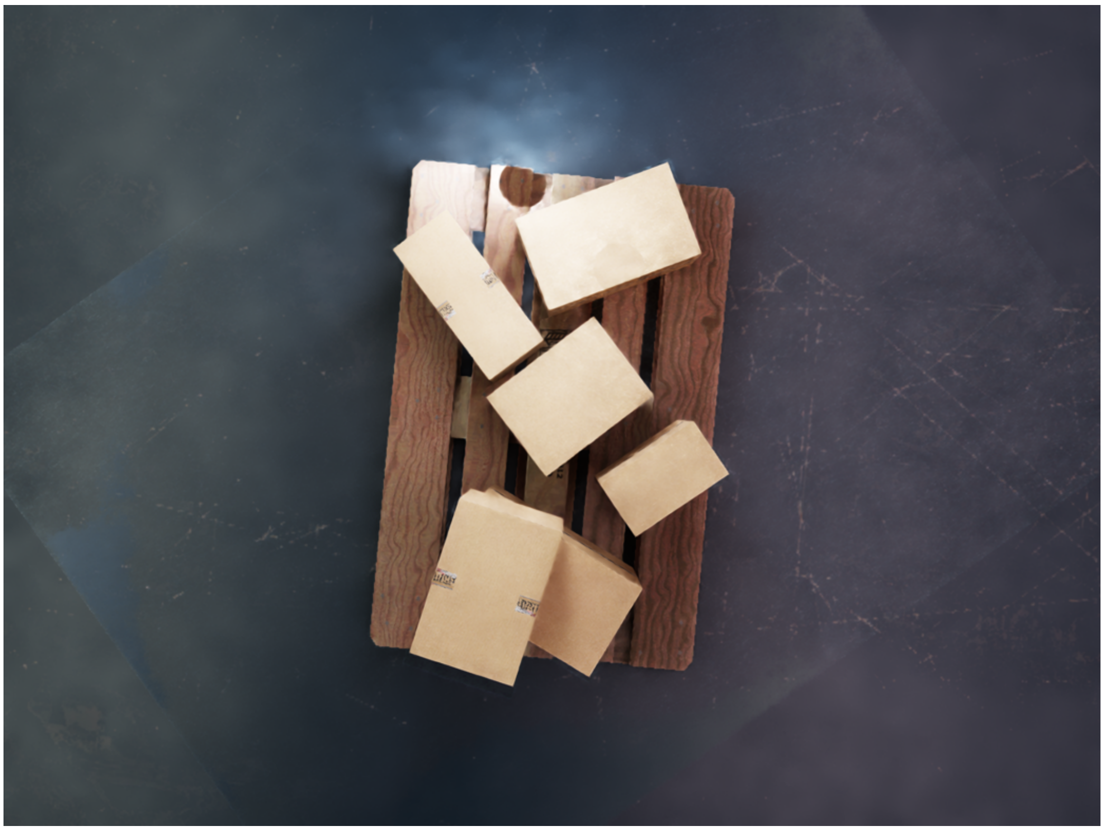


Figure 11 - Scene with overlapping boxes and light beam as disturb

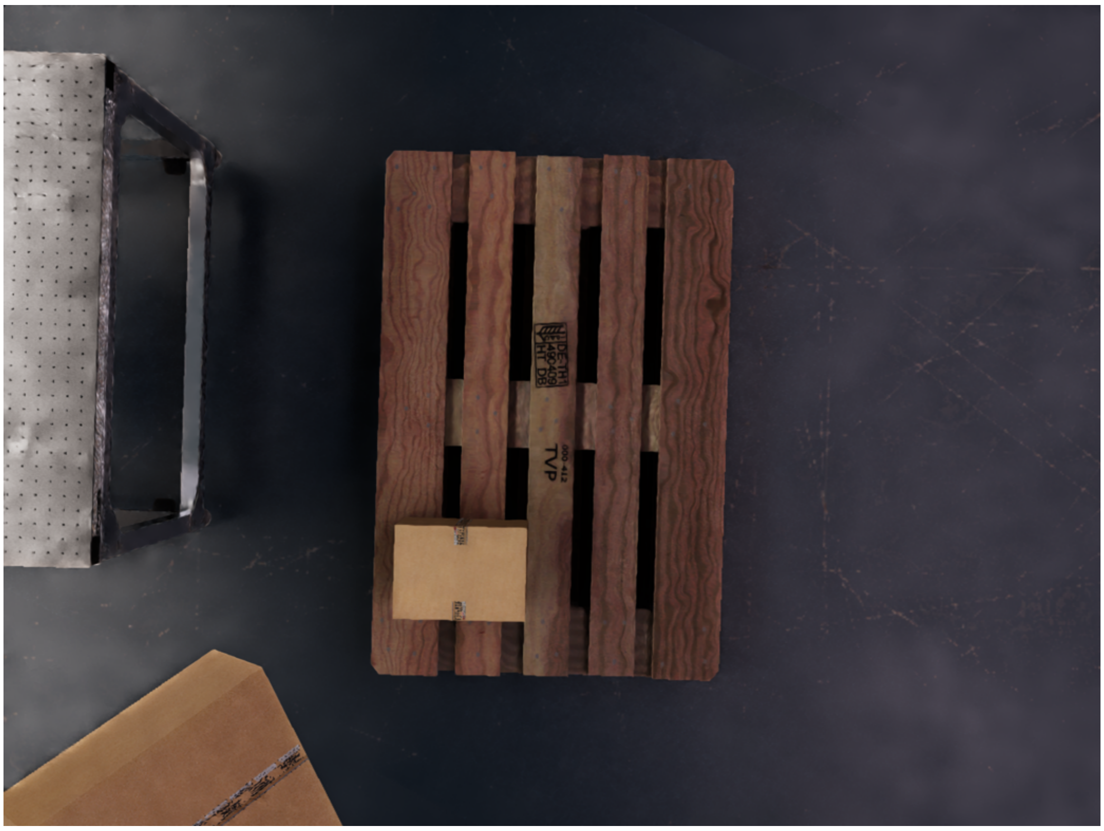


Figure 12 - Scene with a box outside the pallet and the table visible in the right edge of the picture

7.2 Simulation

The dataset was generated using a Python script, which was divided into four main stages:

1. Instantiation of scene elements:
    - Plane: The ground plane was created.
    - Pallet: The model of the pallet with its walls was imported and placed at a fixed position.
    - Camera: Positioned to point toward the centre of the pallet area.
    - Lighting: Scene lighting is added.
    - Disturbance Elements: Additional random objects were imported and positioned in the scene.
2. Box configuration:
    - The number of boxes in each iteration was set using the quantity array (e.g., 0, 1, 2, 5 and 10 boxes)
    - A random texture was applied from a predefined folder.
    - Boxes were introduced at a height and dropped, allowing physics simulation to randomize their arrangement.
3. Physics simulation: A loop ensured that the boxes fell and stabilized on the pallet before images are captured. The 1000-step counter ensured sufficient time for the simulation to settle.
4. Randomization and Image Capture:
    - Camera Pose: Adjusted to simulate various camera positioning.
    - Lighting Parameters: Randomized for intensity, colour, and position to introduce diversity in lighting conditions.
    - Disturbance Elements: Their placement was modified to alter shadows and visual noise in the scene
    - Image Capture: The scene was rendered and saved as an image.

Algorithm 2 represents the pseudocode of the simulation.

```
# Element Instantiation
plane = create_plane()  # Creates the ground plane
pallet = import_pallet()  # Loads the pallet model
camera = create_camera()  # Sets up the camera
light = create_light()  # Adds lighting
disturb_element = import_disturb_element()  # Adds disturbance elements (e.g., table)

# Define quantity of boxes
quantity = [0, 1, 2, 5, 10]  # Different numbers of boxes for each case

for qnt in quantity:
    for i in range(qnt):
        box = import_box()  # Load box model
        box <- randomize_texture()  # Apply random textures to each box

    for i in range(50):  # Generate 50 images per configuration
        for j in range(1000):  # Counter for physics simulation
            # Wait until the boxes stabilize after being dropped
            if j == 999:
                # Randomize camera, lighting, and disturbance elements
                camera <- modify_pose()
                light <- randomize_parameters()
                disturb_element <- modify_pose()

                # Save the image
                image <- save_image()
```

Algorithm 2 – dataset creation

8 Convolutional Neural Network Model

8.1 Mask R-CNN

The model selected for this project was a pretrained Mask R-CNN model available on the Pytorch website[37].

The idea was not to choose the highest-performing model for achieving the best possible results but to use a generic object detection model sufficient to evaluate the performance of the synthetic dataset created using NVIDIA Isaac Sim.

Training a model from scratch would have demanded significantly more time, a larger dataset, and could have introduced variability in results, as the starting point would have been random.

Mask R-CNN is a robust framework capable of performing instance segmentation, providing object masks along with bounding boxes as its output.

The development of Mask R-CNN builds upon Faster R-CNN by adding a third branch to its architecture.

While Faster R-CNN outputs a class label and a bounding box for each detected object, Mask R-CNN extends this functionality by generating an object mask as well.

The Mask R-CNN framework consists of two main stages:

1. Region Proposal Network (RPN): This stage proposes candidates bounding boxes for potential objects in the image.
2. Second stage: In this stage, the model processes each proposed region to predict:
    - The object class.
    - The bounding box offset.
    - A binary mask for each Region of Interest (RoI)[38].

The model was pretrained using *MaskRCNN_ResNet50_FPN_V2_Weights.COCO_V1*.

8.1.1 COCO Dataset

Common Object in Context (COCO) is a large-scale object detection, segmentation, and captioning dataset.

COCO dataset was developed with the goal of advancing the state-of-the-art in object recognition by gathering images of complex everyday scenes containing common objects in their natural context.

Objects are labelled using per-instance segmentations to aid in precise object localization.

The dataset contains 330k images, with a total of 1.5 million labelled instances of 91 stuff types and 80 object categories.
The dataset is limited to category labels that are commonly used by humans when describing objects (e.g., dog, chair, person, etc.) and, eventually, some object categories may be parts of other categories (e.g., a face may be part of a person)[39].

8.2 Dataset definition

The dataset was constructed following the specifications of the case study to maintain consistent metrics and enable the mixing of real and synthetic images during testing phases.

8.2.1 Synthetic Dataset Outputs

The output of the synthetic dataset included:

- RGB image
- Instance segmentation
- Normal image

These outputs were organized into a custom dataset format for compatibility with the Mask R-CNN model.

8.2.2 Custom Mask Creation

Using the instance segmentation and normal image, it was possible to create a custom mask that segmented each face of every box in the scene.
The process can be summarized as follows:

1. Instance Segmentation: The instance segmentation output was divided into N masks, where N is the number of boxes in the scene. During early testing, these masks were used to train the model to recognize entire boxes within the scene.
2. Face Segmentation: In the second version of the code, functionality was added to segment the individual faces of the boxes instead of the boxes as a whole.
    a. Each mask was multiplied by the corresponding normal image, producing a new mask where each box face has a distinct value, representing (x, y, z) values of the scene.

   b. By splitting these news masks into sets containing only one unique value, the final mask with segmented box faces was obtained.
3. Bounding Box Computation: after generating segmented masks, bounding boxes were computed to create the ground truth data needed for training the model.

8.2.3 Dataset Preparation and Augmentation

Data augmentation was used to increase dataset diversity by generating transformed versions of the original images. Random transformations applied in this project included:

- Colour Jitter
- Grayscale
- Horizontal Flip
- Vertical Flip

By incorporating these steps, the dataset achieved a balance of realism and variability, ensuring robust model training.

8.3 Metrics

The training phase involved iterating over the dataset for a predefined number of cycles, called epochs.
In this thesis, the model undergone 30 to 50 iterations.
To evaluate the prediction accuracy different metrics were analysed.

8.3.1 Precision

Precision is the proportion of correctly identified positives (True Positives, TP) out of all predicted positives (True Positives + False Positives, FP):

$$P = \frac{TP}{TP + FP}$$

A high precision indicates strong confidence in classifying a sample as Positive, producing few False Positives, and leading to more reliable predictions.
A model focused only of Precision could miss a substantial number of True Positives.

8.3.2 Recall

Recall is the proportion of True Positives (TP) out of all actual positives (True Positives + False Negatives, FN):

$$R = \frac{TP}{TP + FN}$$

High recall reflects the ability of the model to identify a large proportion of True Positives but optimizing only for Recall could increase the number of False Positives.

8.3.3 Precision-Recall Curve

The Precision-Recall curve illustrates the trade-off between Precision and Recall across different thresholds for converting prediction scores into classes.
This allows designers to select optimal threshold to maximize both metrics.
When the score is equal or above the threshold, the sample is classified as belonging to a class, otherwise, it is classified as the other class.
Precision-Recall curve gives a better idea of the overall accuracy of the model.

8.3.4 F1 Score

F1 Score is a common metric for evaluating the balance between Precision and Recall, being the harmonic mean of the two variables:

$$F1 = 2\frac{P \cdot R}{P + R}$$

A high F1 score indicates good balance between Precision and Recall, while a low F1 score suggests an imbalance, with one metric dominating the other.
During the simulations of this thesis project, F1 score was used to determine the best threshold to be used in inference.

8.3.5 Average Precision

Average Precision (AP) summarizes the Precision-Recall curve as a single value, computed as the area under the PR curve:

$$Average\ Precision\ (AP) = \int_{r=0}^{1} p(r)dr$$

With $p(r)$ representing the function of precision at different recalls.

8.3.6 mean Average Precision

For multi-class classification tasks, the mean Average Precision (mAP) is calculated as:

$$mAP = \frac{1}{N}\sum_{i=1}^{N} AP_i$$

Here, N represents the number of classes, and AP_i is the Average Precision for the i-th class[7].

8.3.7 Intersection over Union (IoU)

IoU is a metric used to quantify the overlap between the predicted bounding box and the ground truth bounding box. It is defined as the ratio of the intersection area to the union of the two boxes (Figure 13).

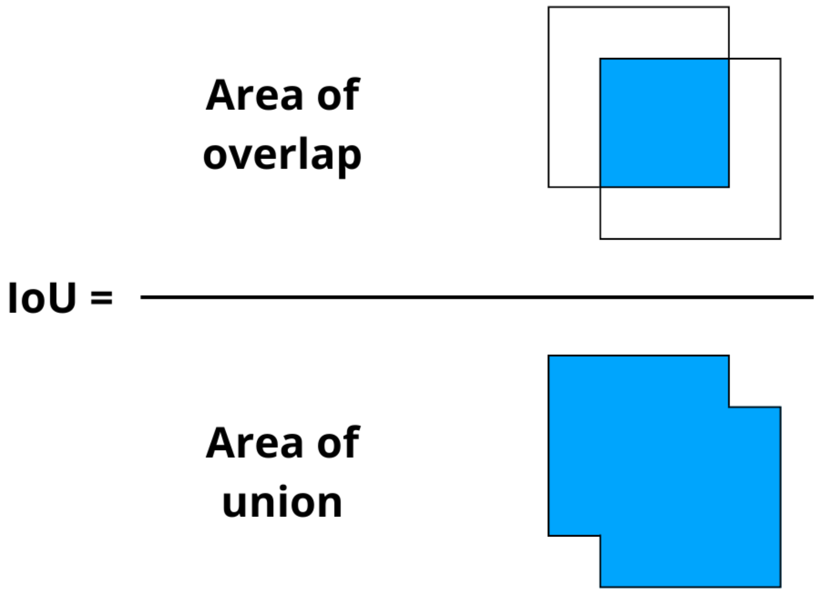


Figure 13 - Formula to compute IoU and a visual representation of overlap and union of bounding boxes

IoU measures the similarity between predicted and ground-truth region for an object detected by the model[40].

It also sets a threshold during training to determine whether a prediction qualifies as a positive detection.

There is not a unique value of IoU to be used during object detection, but it depends on the specific task and dataset:

- A higher IoU threshold reduces the number of False Positives but may also miss some True Positives.
- A lower IoU threshold captures more True Positives but can increase False Positives

For this project, an IoU threshold of 75% was used, as it provided a reliable balance for the face-detection task (Figure 14).

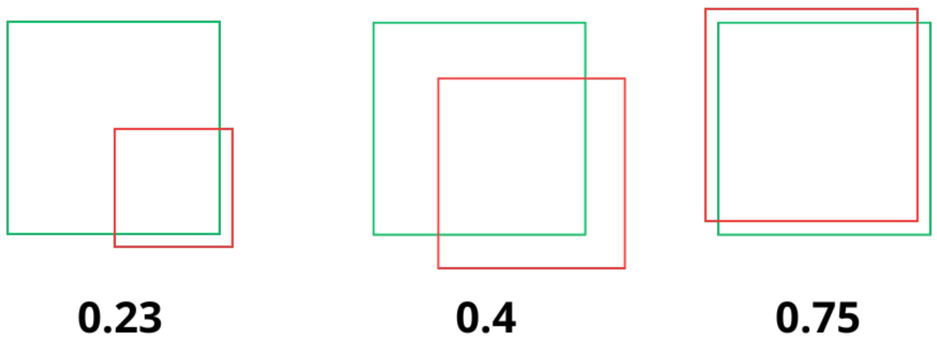


Figure 14 - Example of different IoU values. Green boxes represent ground-truth while red boxes represent predictions

9 Synthetic dataset evaluation

9.1 Introduction

The synthetic dataset was extensively tested during the early phases of the thesis to evaluate the performances of the chosen CNN model and refine the Synthetic Data Generation (SDG) process.

The first training set consisted of boxes with various colourful and intricate textures applied to their surfaces.

This approach aimed to test the ability of the model under challenging conditions, as distinguishing the sides of the boxes was sometimes difficult even for human observers.

A key aspect of this phase was ensuring that the validation set included textures that were absent in the training set.

This choice allowed for evaluating the generalization capability of the model to unseen data.

The images used during these initial training sessions were not directly comparable to the real dataset in terms of characteristics.

The primary goal at this stage was to gain familiarity with synthetic data and identify areas for improvement.

9.2 Random textures evaluation

The results, depicted in Figure 15, present the mAP trend over the entire training process for both bounding boxes and segmentation at 75% of IoU.

The maximum value of mAP for bounding boxes is 0.785, obtained at epoch 8.

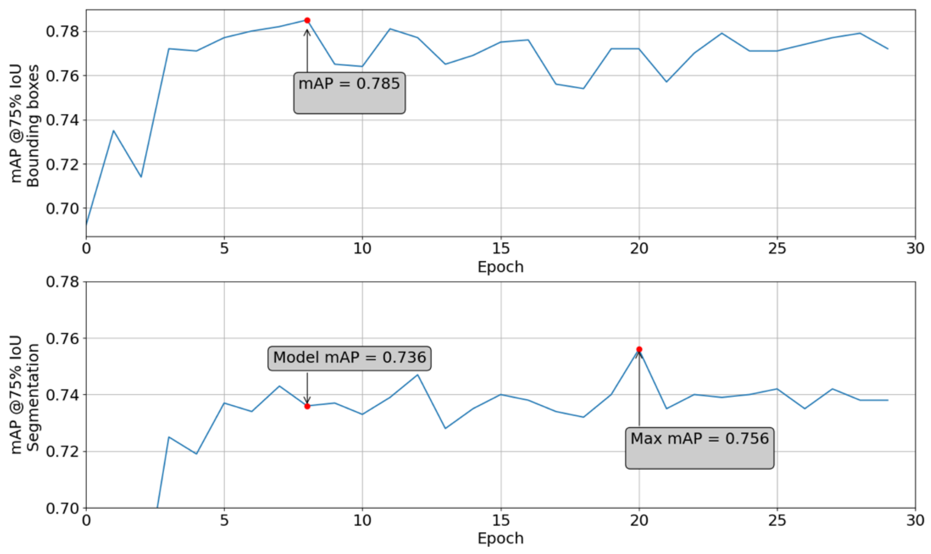


Figure 15 - mAP for the synthethic dataset for bounding boxes (upper graph) and segmentation (lower graph)

The chosen model was the one with the highest performance for bounding boxes, while the graph showing mAP for segmentation in inserted for the sake of completeness.

Figure 16 shows the values of Precision and Recall for the best-performing model on the validation dataset.

The best tradeoff returned a value of 0.779 for Precision and 0.620 for Recall.

The threshold for recognizing a box face by the model is determined using the F1 score, whose maximum value is 0.691.

The threshold value is 0.937, which is subsequently applied to deploy the model in production with new images.

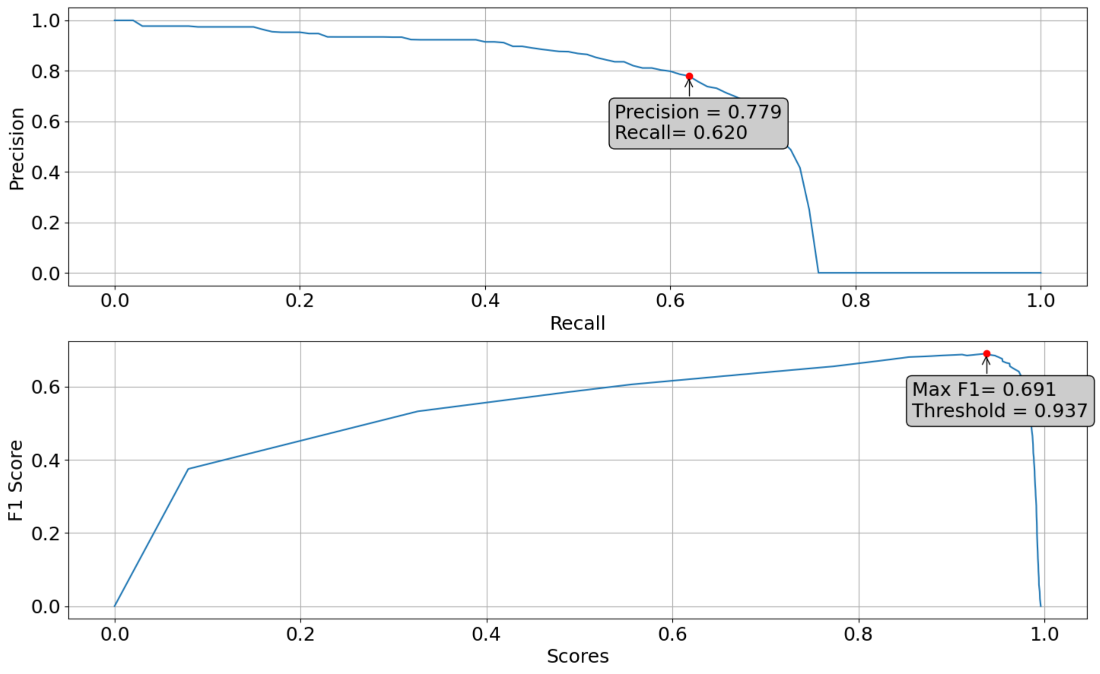


Figure 16 - Precision and Recall (upper) and F1 Score (lower) for the best-performing model on the validation dataset

Once the model was available, test was conducted to compute the mAP at 75% IoU on the testing set, providing a baseline numerical evaluation.

The results indicated a mAP of 0.859 for bounding boxes and 0.860 for segmentation on the test set.

Qualitative testing was also performed by analyzing individual images and evaluating the performance of the model, as illustrated in Figure 17.

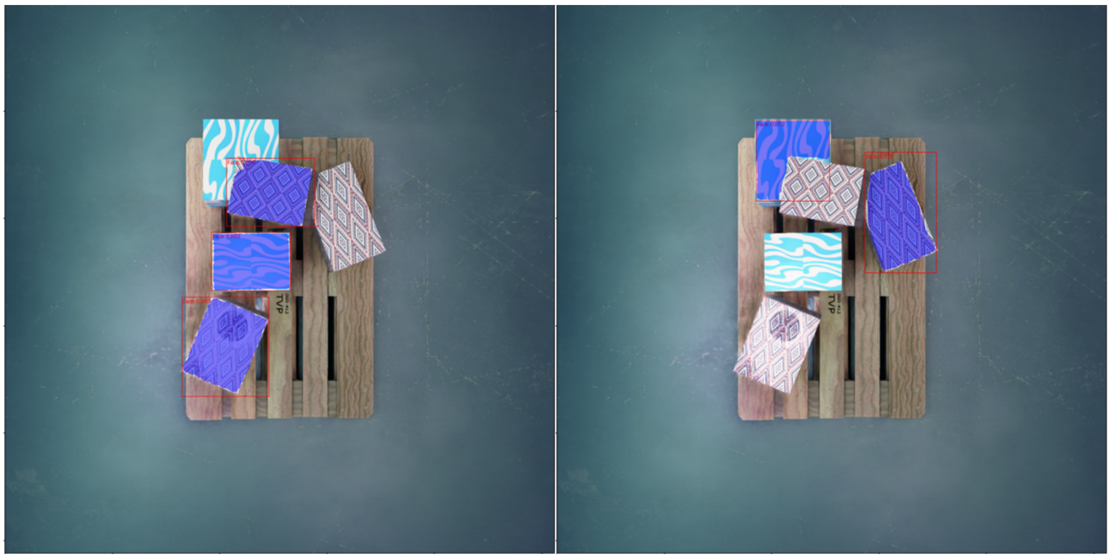


Figure 17 – Qualitatively example of object detection and instance segmentation with the model chosen. Blu areas are the segmentation masks, while red boxes are the bounding boxes.

The model demonstrated the ability to detect each face of a box while filtering out those with small areas.

These results are achieved using a testing set containing new images; however, the box textures used were limited to those already encountered during the training phase.

9.3 New textures in test set

The testing continued by creating a series of test datasets containing images with “new” textures that had not been encountered during training.

The percentage of images with new textures varied from 20% to 100%, with an increment of 20% for each dataset.

Tests were carried out with the best-performing model for bounding boxes found at Section 9.2

As shown in Table 2 and in Figure 18, the mAP for both bounding boxes and segmentation exhibited a linear decline as the percentage of new textures increased.

The performance degradation was more pronounced for segmentation compared to bounding boxes.

As said previously, the results for segmentation are highlighted for sake of completeness, as in the other testing phases segmentation had been turned off.

Dataset	Bounding Box	Segmentation
Old textures	0.859	0.860
20% New textures	0.810	0.802
40% New textures	0.794	0.770
60% New textures	0.768	0.727
80% New textures	0.741	0.684
100% New textures	0.702	0.645

Table 1 – mAP values for bounding boxes and segmentation on testing sets with an increasing percentage of new textures

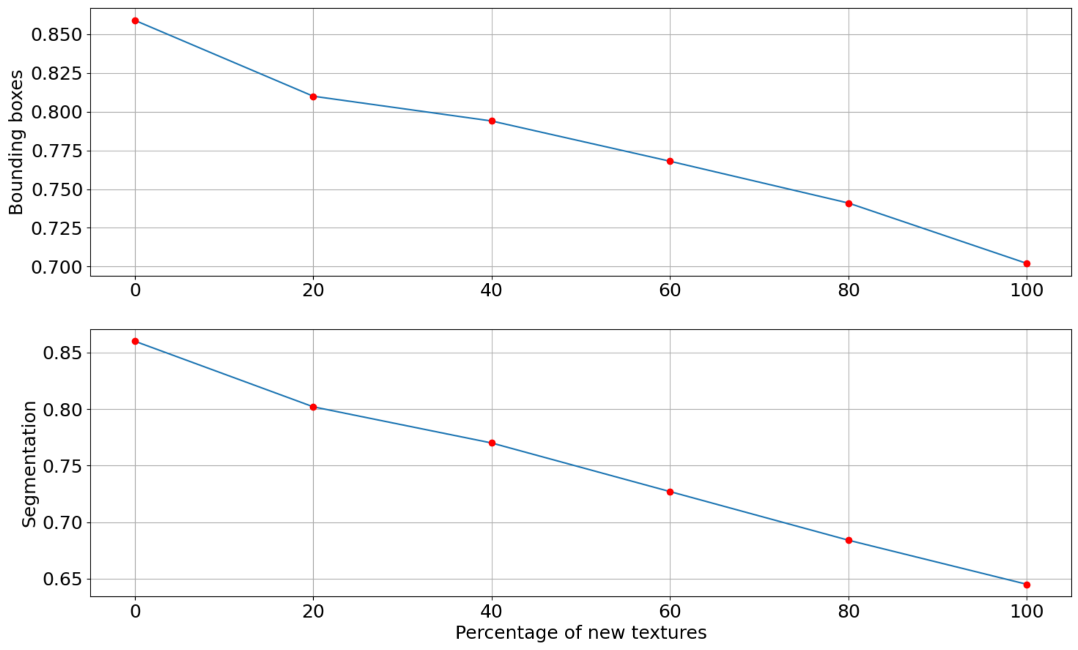


Figure 18 - Graph showing the decreasing trend for bounding boxes (upper) and segmentation (lower) when new textures are added to the testing set

10 Real dataset evaluation

10.1 Introduction

The real dataset (Figure 19 and Figure 20) was evaluated as a reference for analysing the performances of the synthetic dataset.

The labelling process for the real dataset differed slightly from that of the synthetic dataset, as it was performed manually rather than using a heuristic model.

The key differences in box annotation were:

- Only the largest visible face of each box was annotated.
- If a face of a box was occluded by another, the lower face was annotated only if more than 90% of it was visible.
- A box was not annotated if fewer than three corners were visible in the image.

The pallet was also treated differently in terms of annotation, with the following rules:

- Only the visible sections of the pallet were annotated.
- Overlapping labels between the pallet and the boxes were avoided.

The training process lasted for 30 epochs.

For the real dataset, segmentation was not considered, as no masks were provided for the faces of the boxes.

However, it is important to note that the model was still pre-trained for segmentation tasks.

Additionally, pallet annotations were excluded due to differences in annotation rules between the real and synthetic datasets, as discussed in Section 11.1.

Unlike synthetic images, captured in RGB format, real dataset images were grayscale.

Consequently, the random grayscale transform was turned off during training.

Boxes used in the real dataset were common industrial boxes and feature various textures, including:

- COMAU boxes
- Wine/Beer boxes
- Food boxes

A notable characteristic was the occasional presence of transparent duct tape to seal the boxes.

This could create varying reflective behaviour under high-light conditions.

The issue was highlighted during the real system training, as it can lead to undesired behaviour.

Specifically, the model might mistakenly recognize two separate faces instead of a single face.

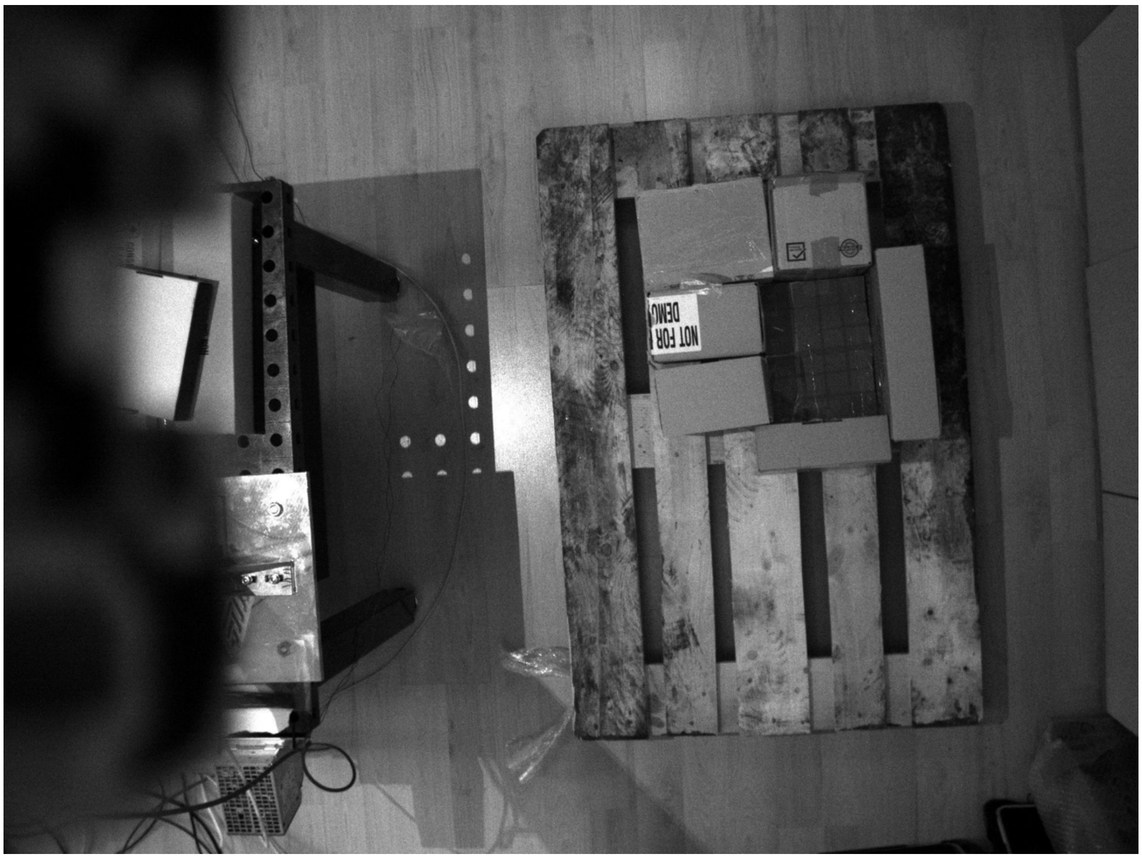


Figure 19 - Example of image from the real dataset. This image presents different disturbing elmements.

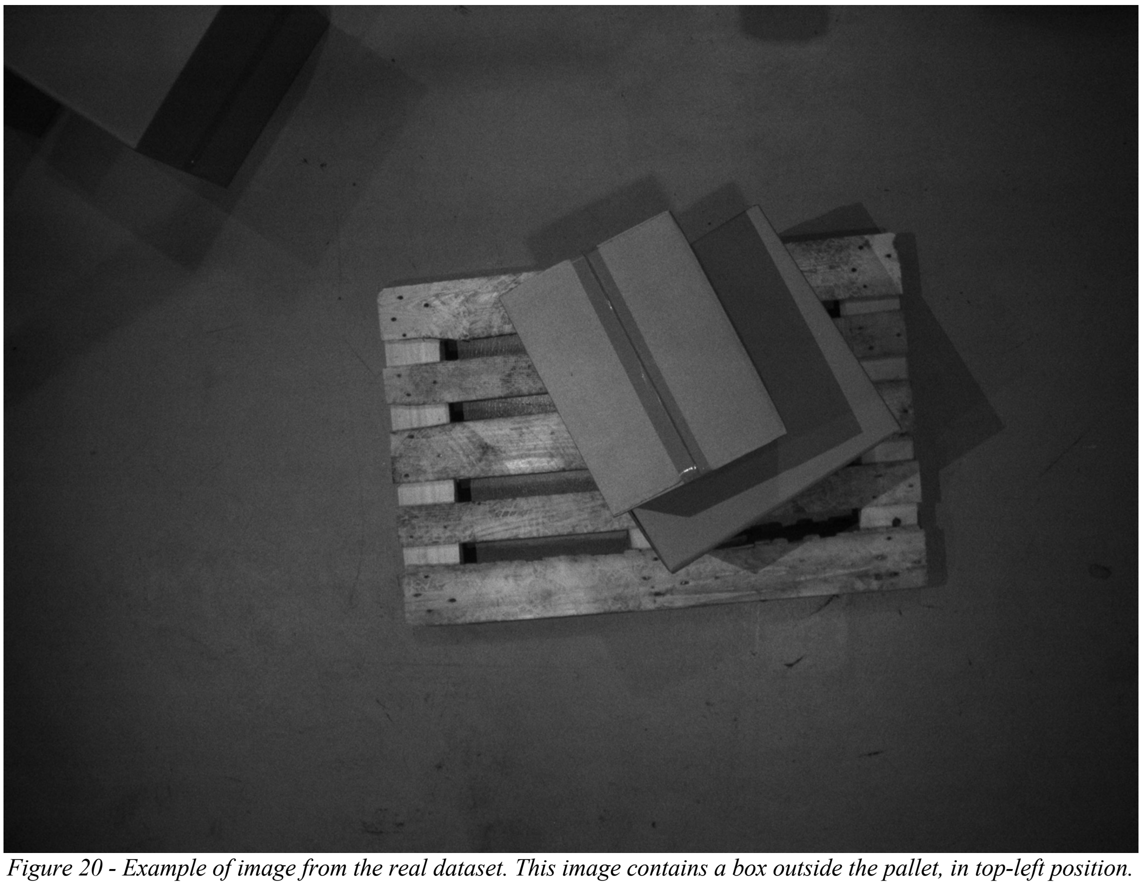


Figure 20 - Example of image from the real dataset. This image contains a box outside the pallet, in top-left position.

10.2 Results

The mean Average Precision (mAP) for training dataset over 30 epochs is shown in Figure 21, with a maximum value of 0.932, achieved at epoch 6.

The mAP followed an asymptotic behaviour, with values stabilizing and oscillating between 0.920 and 0.930 starting from epoch 5.

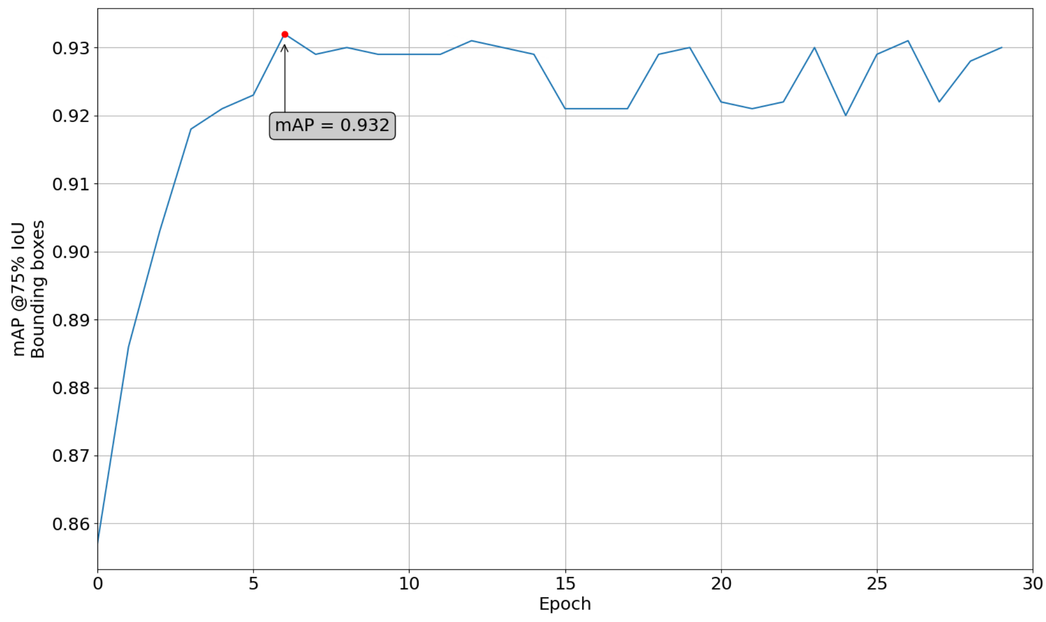


Figure 21 - mAP in training for real dataset

Figure 22 shows the values of Precision, Recall and the F1 score.

Precision and Recall values of the best model, computed using the validation set, were both high, aligning with the mAP trend observed during training.

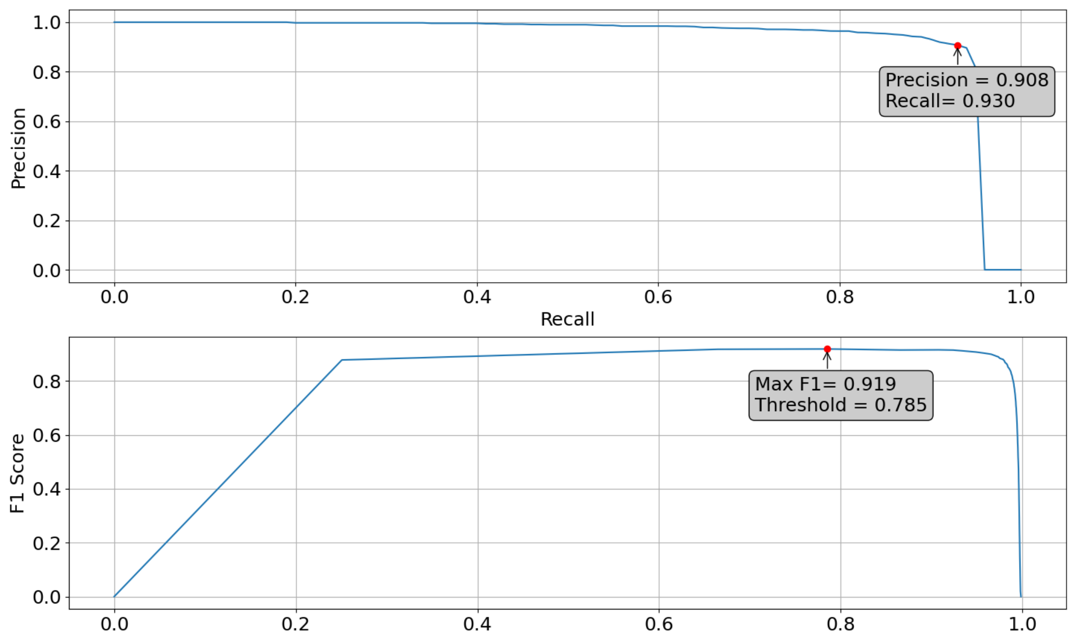


Figure 22 - Precision-Recall curve over validation dataset (upper) and corresponding F1 Score curve (lower)

11 Mixed datasets

11.1 Introduction

Before combining the real and synthetic datasets, certain precautions were necessary due to the differences in labelling conventions.
The pallet was not labelled, because the labelling differed from real dataset to synthetic dataset. To align the labelling of the synthetic data with the rules applied to the real dataset a heuristic would have been necessary, but development was simply not worth it.
The augmentation function was also modified, since the real dataset was composed of grayscale images, while the synthetic was made of RGB images.
To have a uniformity, a grayscale transformation was added to all the synthetic images, to decrease gap-domain between the two sources of data.
Mixing the two dataset was essential to evaluate the behaviour of the model when trained with data from distinct domains.
However, adjustments to the code were necessary, as real and synthetic data required different handling during pre-processing and training.
The testing process involved two main scenarios:

- Addition of real images to a synthetic dataset.
- Addition of synthetic images to a real dataset.

Both tests sought to demonstrate the potential of synthetic data to enhance the training process of CNNs in industrial systems.
By reducing dependency on real data, synthetic data can offer a practical alternative in scenarios where data collection is challenging or resource intensive.
The results of these tests are presented in the following section, while a more detailed discussion is provided in Chapter 12.

11.2 Synthetic data with real data

The objective of this testing process was to simulate a scenario where the available real data was insufficient and to determine a minimum percentage of real data that, when added to a synthetic dataset, yielded satisfactory performance.

This testing involved incrementally adding a percentage of real images to the synthetic dataset and evaluating the performance of the model after each addition.

The synthetic dataset consisted of 250 images, while the number of real images added ranged from 0 to 65.

Each simulation is conducted over 30 epochs, and the best-performing model was selected based on the mAP for bounding boxes, calculated at 75% of IoU.

11.2.1 Synthetic images without real images

Figure 23 highlights the mAP over 30 epochs, with highest value of 0.540 achieved at epoch 14.

The mean Average Precision exhibited an irregular pattern, characterized by local minima and maxima, without a clear or consistent trend.

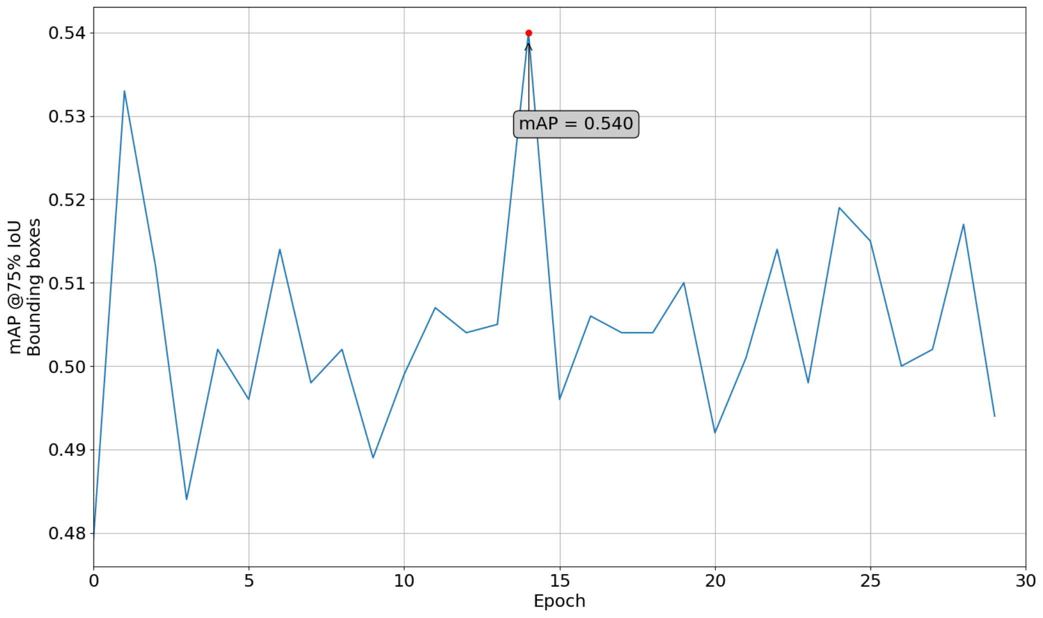


Figure 23 - mAP during training with a dataset containing only synthetic images

Figure 24 shows the values of Precision and Recall for the best model and the corresponding F1 score, computed on the validation set.

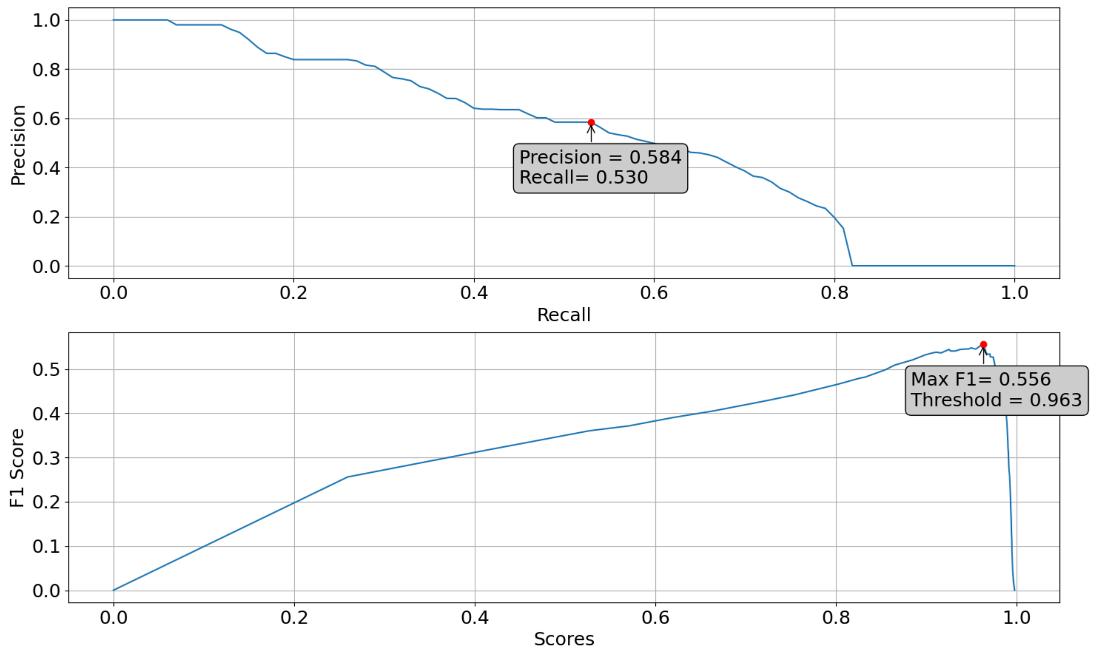


Figure 24 - Precision-Recall curve computed over validation dataset (upper) and corresponding F1 score (lower)

As anticipated, the results were not very high due to the domain differences between synthetic and real images.

Both Precision and Recall were low, indicating a high number of FP and FN compared to TP.

11.2.2 Synthetic images with 25 real images

For the dataset containing 25 real images, the mAP for bounding boxes during training over 30 epochs is presented in Figure 25.

The highest mAP achieved by the model was 0.754, obtained at epoch 20.

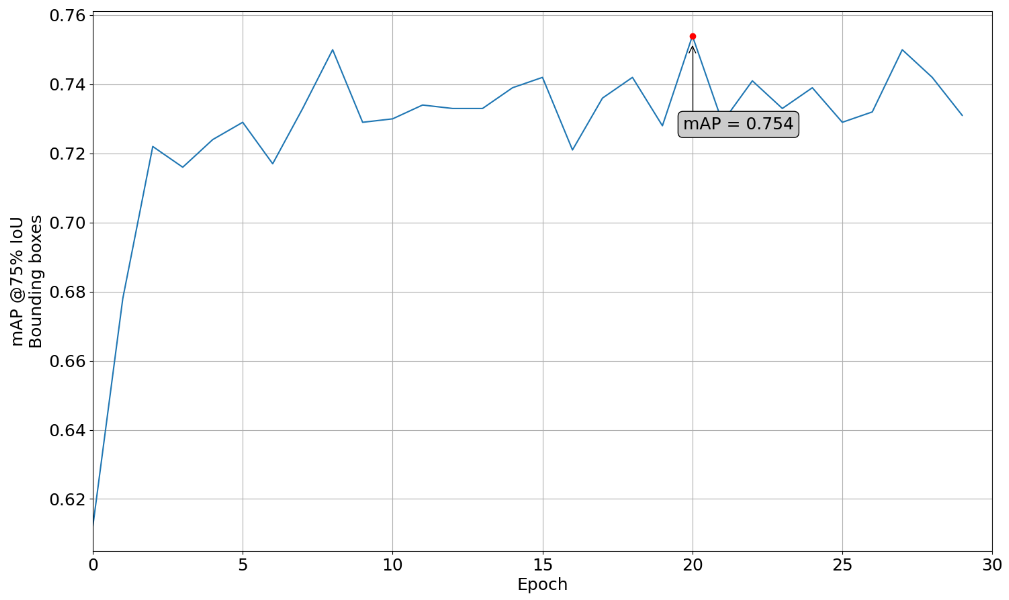


Figure 25 - mAP during training with 25 real images and 250 synthetic images

The behaviour of the function became more regular, starting to resemble the trend observed in the benchmark with only real data.

In Figure 26, the Precision and Recall values for the best-performing model are displayed, highlighting the optimal trade-off between the two metrics.

The F1 score for the model was 0.728, corresponding to a threshold of 0.901.

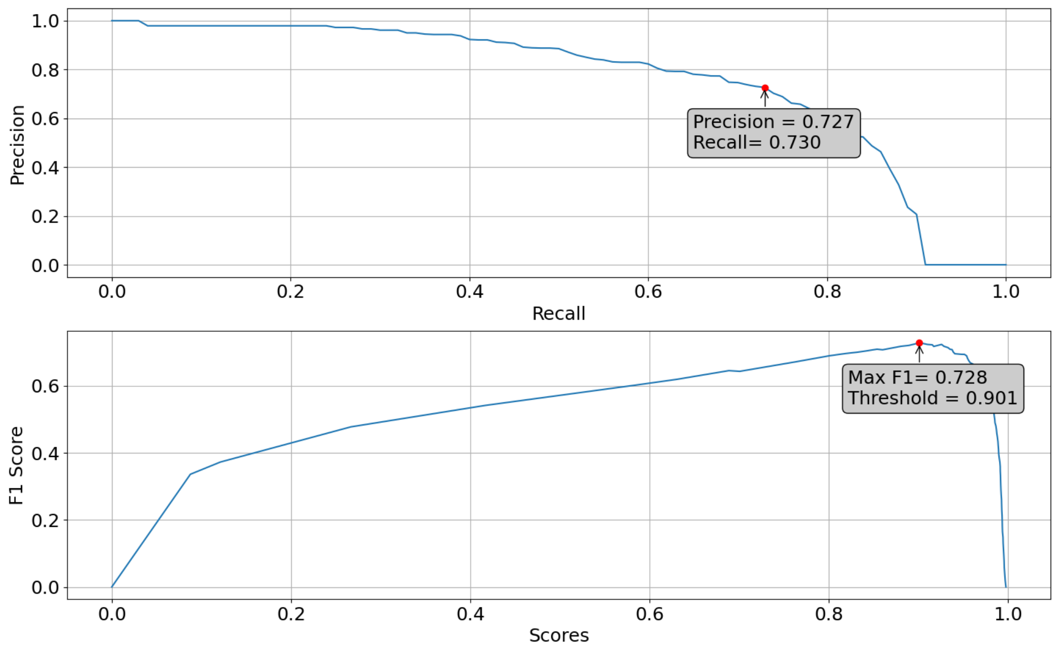


Figure 26 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

Both Precision and Recall values showed improvement, with an Area Under the Curve (AUC) higher than that observed in the test using only synthetic images.

11.2.3 Synthetic images with 50 real images

Figure 27 illustrated the mAP during training for the dataset containing 50 real images combined with synthetic images, with the highest mAP value of 0.783.

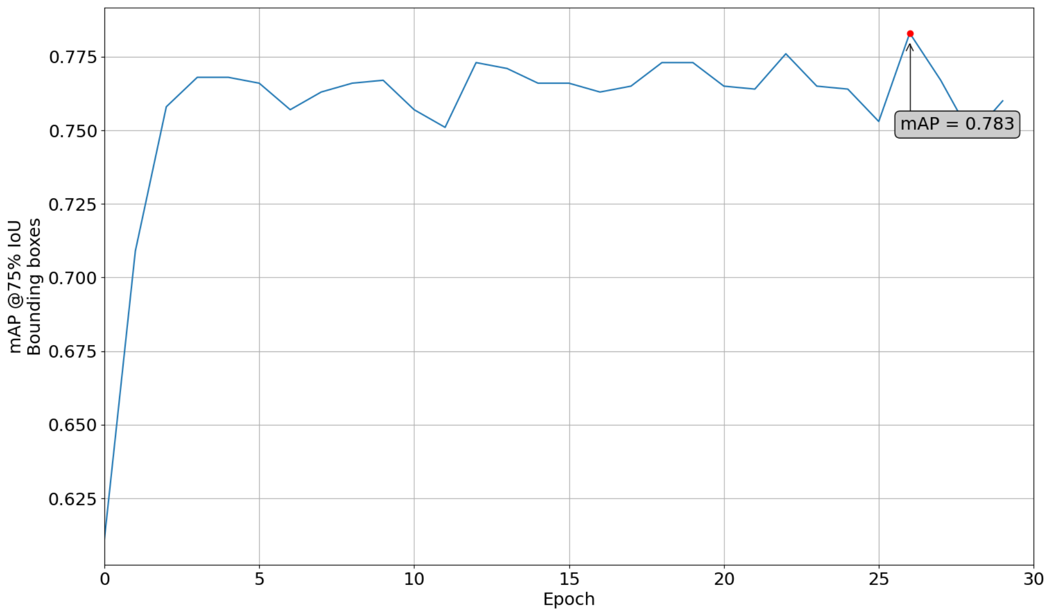


Figure 27 - mAP during training with 50 real images and 250 synthetic images

Figure 28 presents the best trade-off between Precision and Recall for the best performing model on validation dataset, along with the corresponding F1 score.

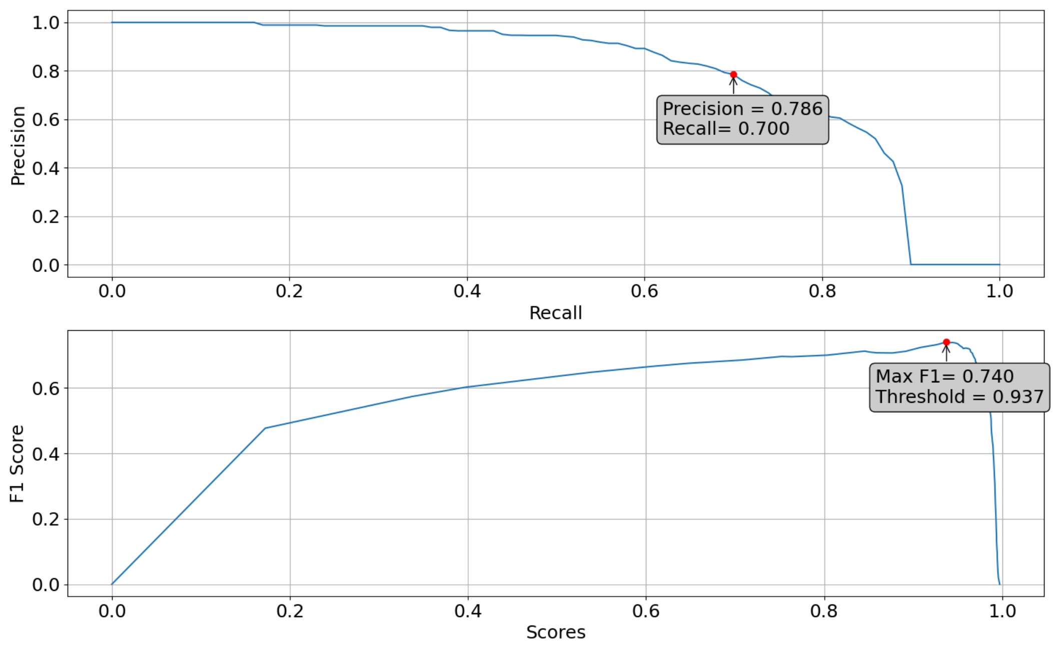


Figure 28 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

11.2.4 Synthetic images with 65 real images

In Figure 29 is shown the mAP during training with 65 real images in the dataset, with the maximum value of 0.829 achieved at epoch 9.

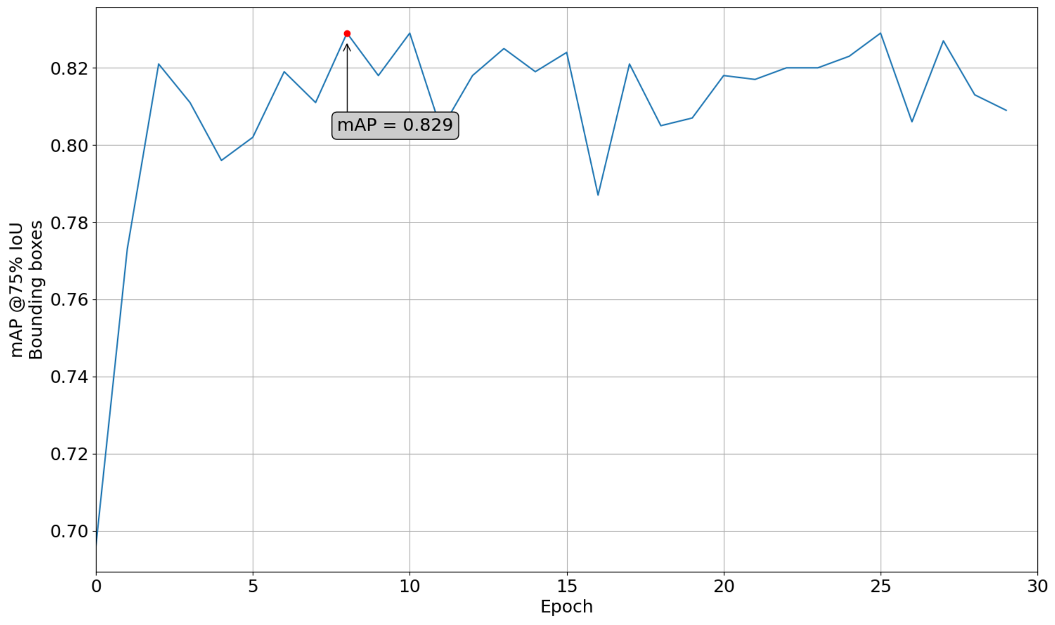


Figure 29 - mAP during training with 65 real images and 250 synthetic images

Figure 30 shows a value of 0.829 for Precision and 0.790 for Recall for the best-performing model on validation dataset.

The corresponding F1 score was 0.943.

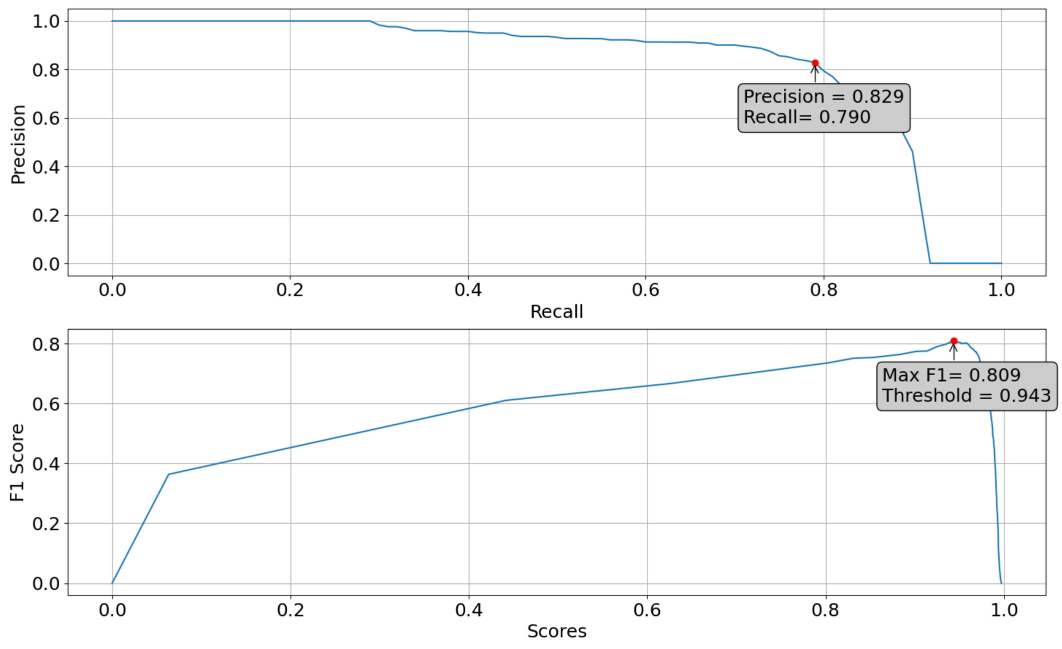


Figure 30 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

11.3 Real images with synthetic data

The second testing phase with the mixed dataset simulated a scenario where a dataset of real images required an injection of synthetic images to enhance its size and achieve improved performance.

In this phase, testing was conducted by adding synthetic images to a base set of 500 real images. The objective was to measure the impact of synthetic data on the performance of the model. The number of synthetic images added to the dataset varied incrementally, ranging from 0 to 250.

11.3.1 Real images

The first test in this phase involved a set of 500 real images, randomly selected from the 1300-image benchmark dataset described in Section 10.1.

The mean Average Precision behaviour during training exhibited an asymptotic trend, stabilizing by epoch 7, oscillating around 0.920, as shown in Figure 31.

The results showed that the model achieved a mAP of 0.926, a value very close to the benchmark mAP of 0.932.

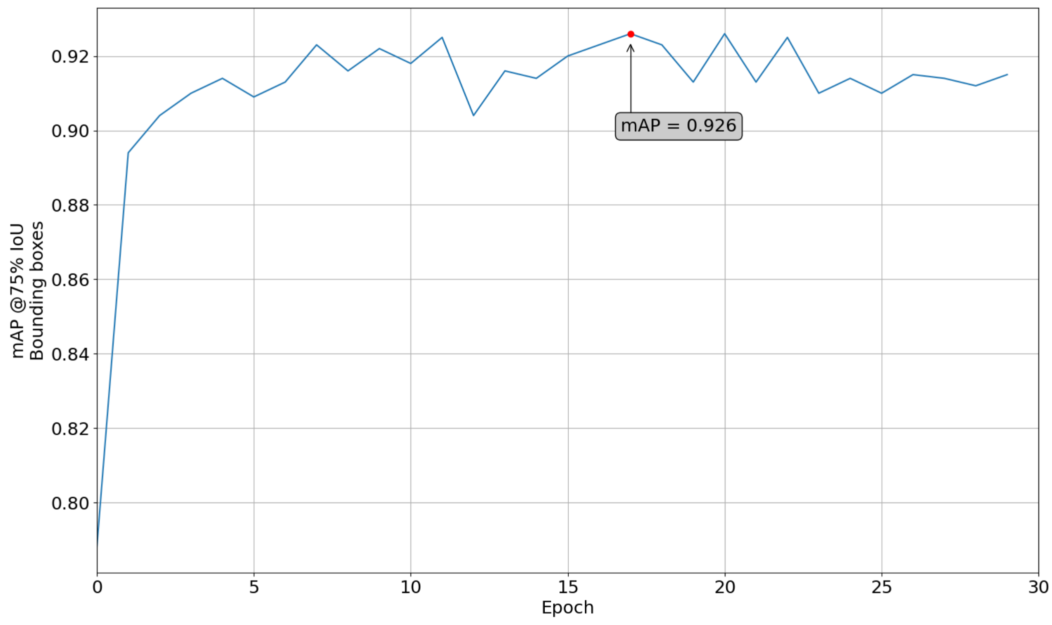


Figure 31 - mAP during training with a dataset containing 500 real images

The Precision and Recall metrics for the best-performing model on validation dataset are displayed in Figure 32.

The optimal trade-off between these two metrics yielded a Precision value of 0.891 and a Recall value of 0.920, with a corresponding F1 Score of 0.905.

.

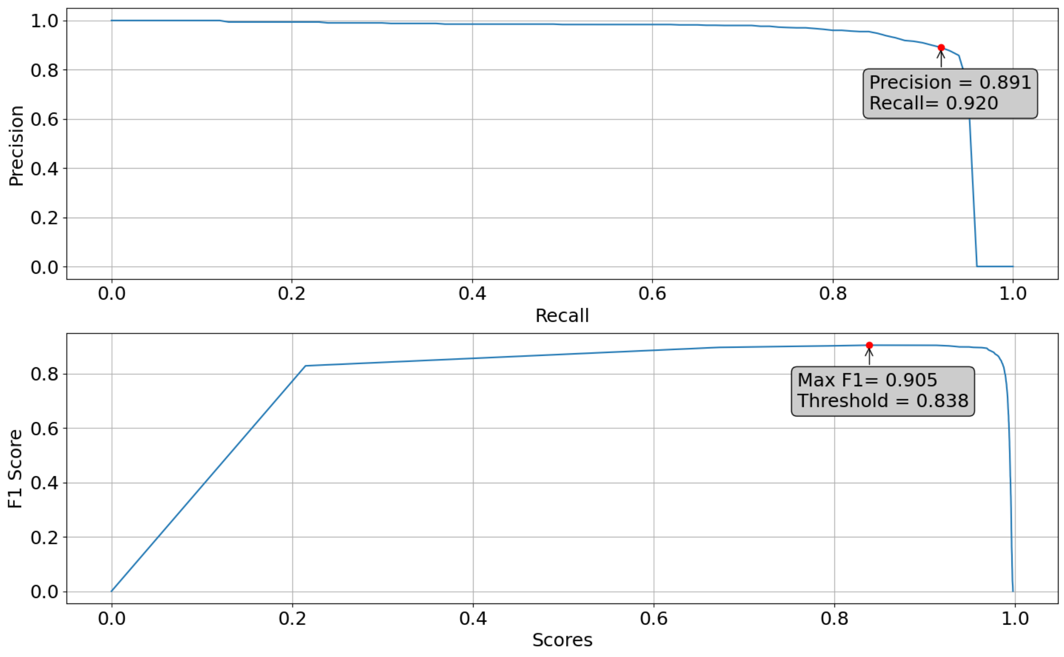


Figure 32 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

11.3.2 Real images and 100 synthetic images

The results from the addition of 100 synthetic images to the dataset of 500 real images are shown in Figure 33, where the best model achieved a maximum mAP of 0.908 on training set.

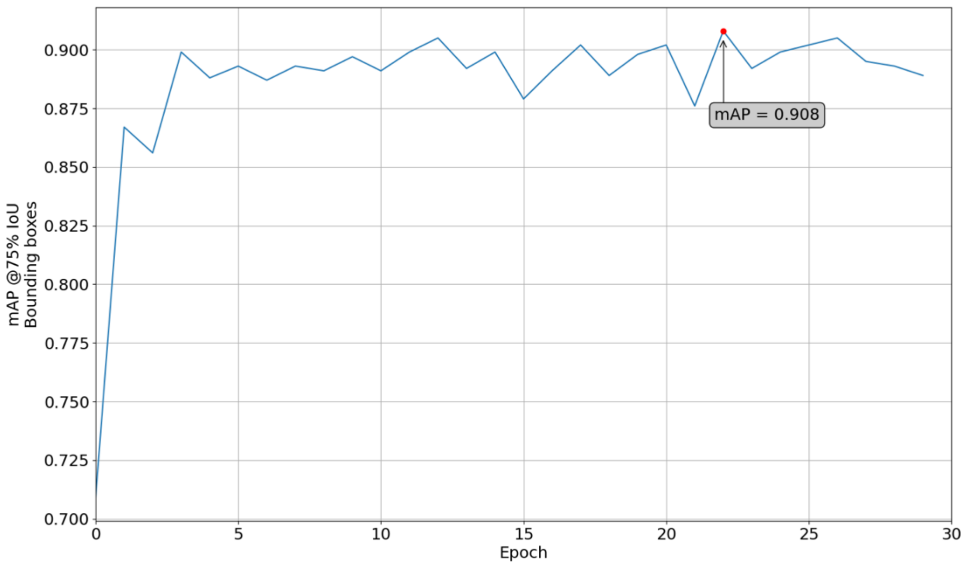


Figure 33 - mAP during training with a dataset containing 500 real images and 100 synthetic images

The best trade-off between Precision and Recall for the model selected results in a Precision of 0.900 and a Recall of 0.890 for the validation dataset is shown in Figure 34.

The corresponding F1 Score was 0.895.

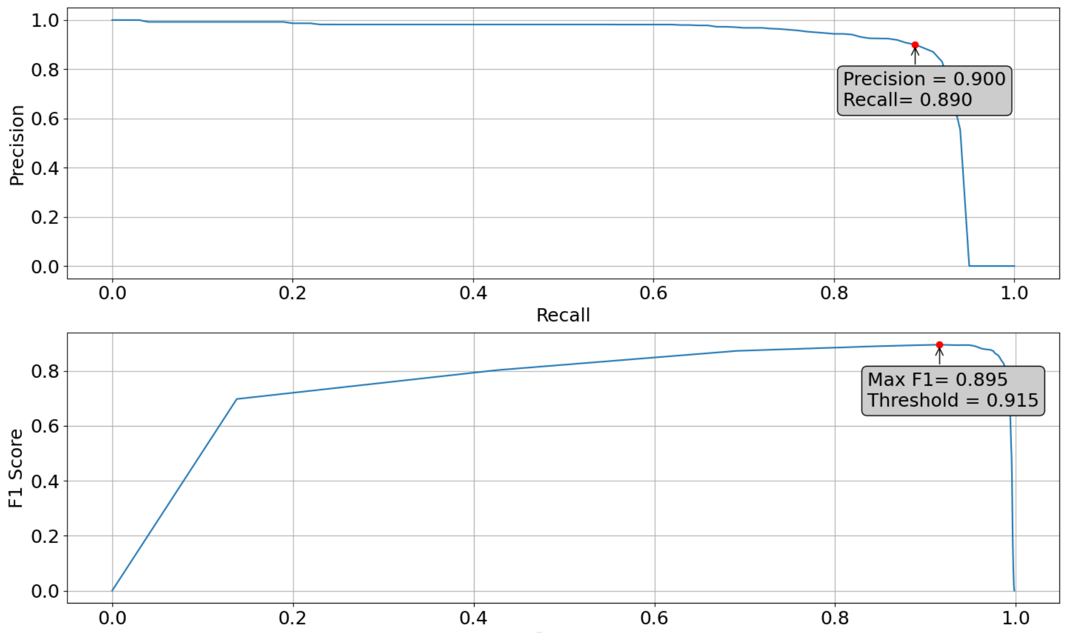


Figure 34 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

11.3.3 Real images and 250 synthetic images

The addition of 250 synthetic images to the dataset did not result in a significant change in the mAP value, as the maximum was 0.923 during training, as shown in Figure 35.

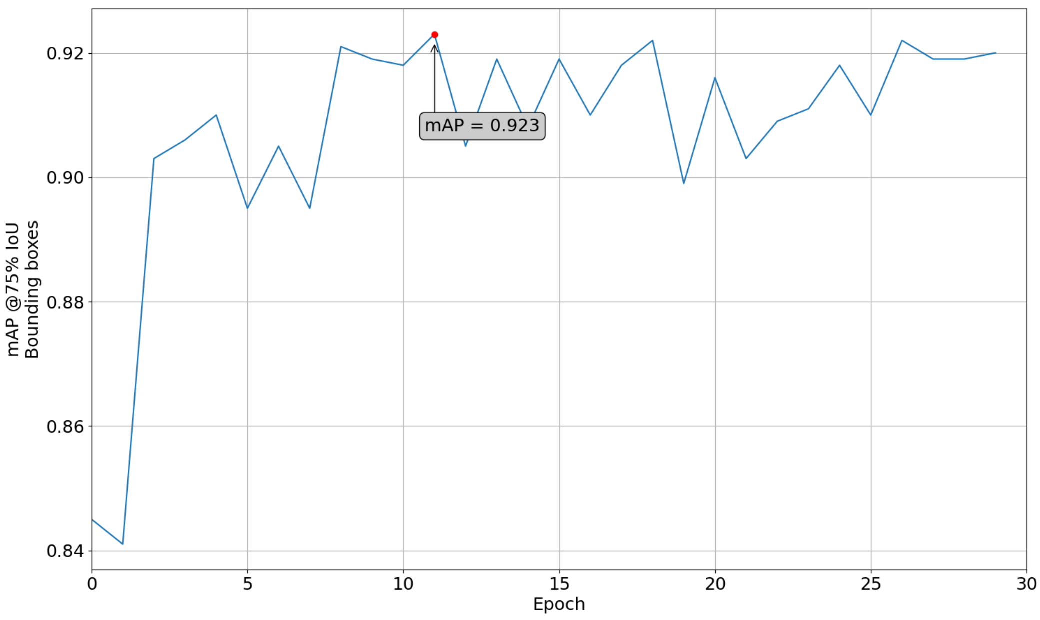


Figure 35 - mAP during training with a dataset containing 500 real images and 200 synthetic images

The maximum values for Precision and Recall for the best-performing model on validation dataset were respectively 0.923 and 0.880 and the corresponding F1 score is 0.901, as shown in Figure 36.

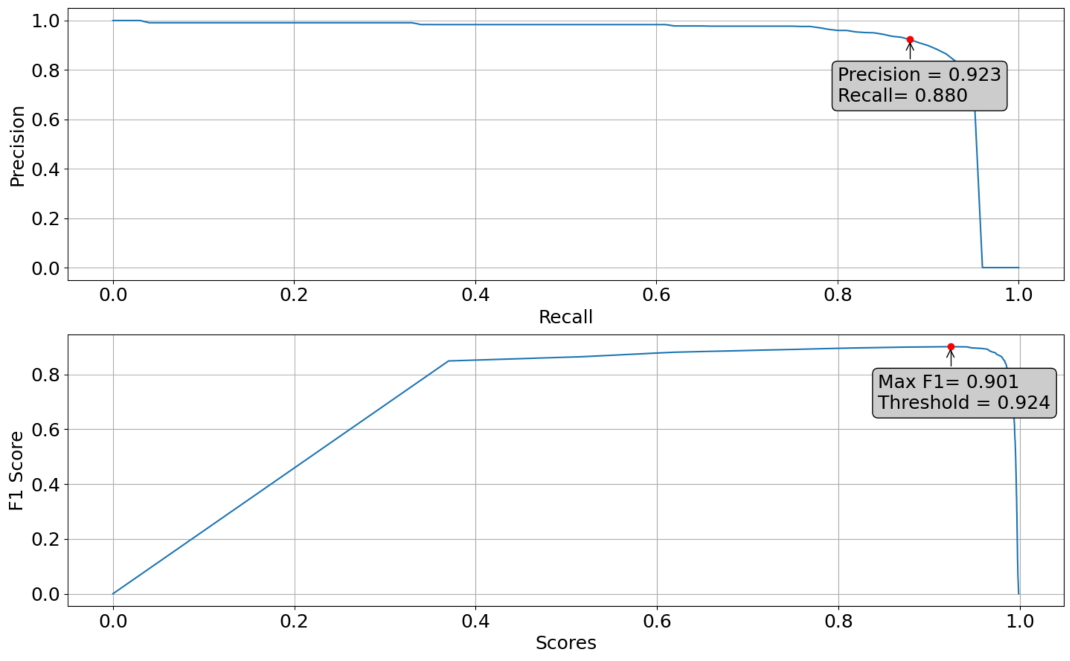


Figure 36 - Precision-Recall curve over validation set of the best model (upper) and corresponding F1 Score (lower)

12 Discussion

12.1 Synthetic data

Testing with only synthetic data was conducted to refine the SDG process and build expertise with the CNN model.

During testing, the test dataset was modified by injecting of new textures, to evaluate how these changes impacted the performances of the model.

As shown in Table 1 and Figure 16, the results revealed a linear decrease in performance when new textures were introduced.

This decline was attributed to the added complexity of recognizing unseen textures.

The performance drop was more pronounced in segmentation tasks than in bounding box detection.

This difference likely stems from the increased difficulty of pixel-level classification for box faces compared to the simpler task of identifying the entire face with a bounding box.

Despite these challenges, the performance of the model remained satisfactory for both for the test set with known textures and the set containing entirely new ones.

When deployed in production, the model gave the following errors:

- Merging of two faces: as shown in Figure 37, the model merged two adjacent box faces into one
- Splitting a single face: As depicted in Figure 38, the model occasionally split a single face into two separate faces.

These errors were likely due to the complex and colourful textures used in testing.

In many cases, these textures made it difficult even for human observers to determine the exact location of a face or to discern whether it represented a single face or two closely positioned boxes.

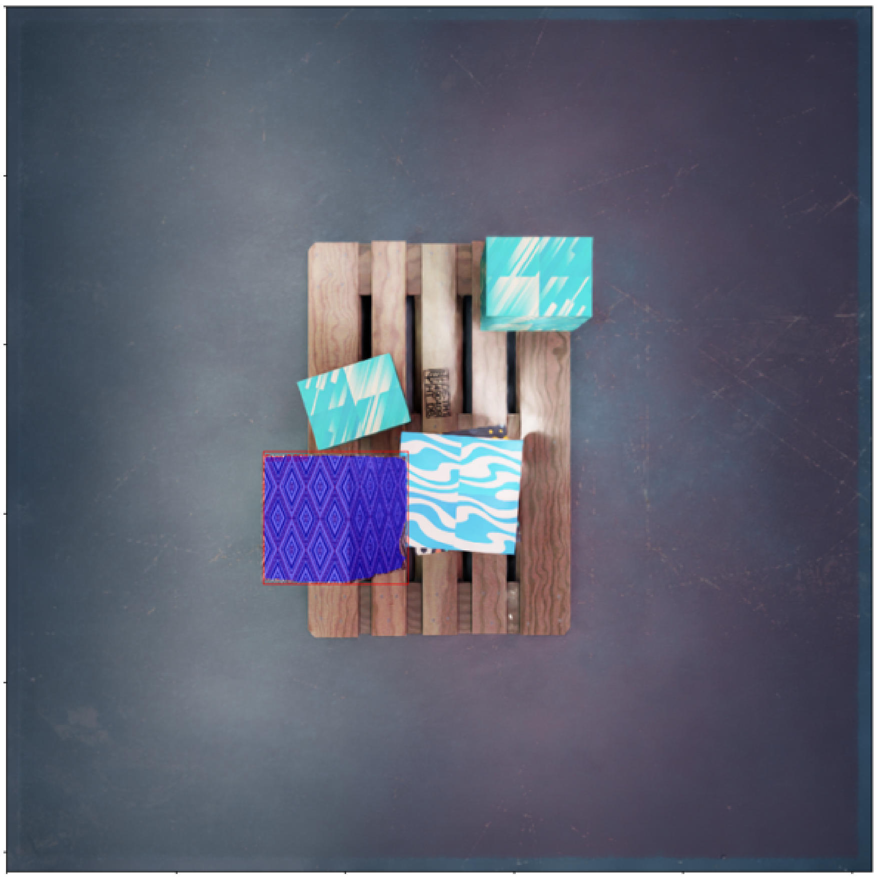


Figure 37 - Merged faces

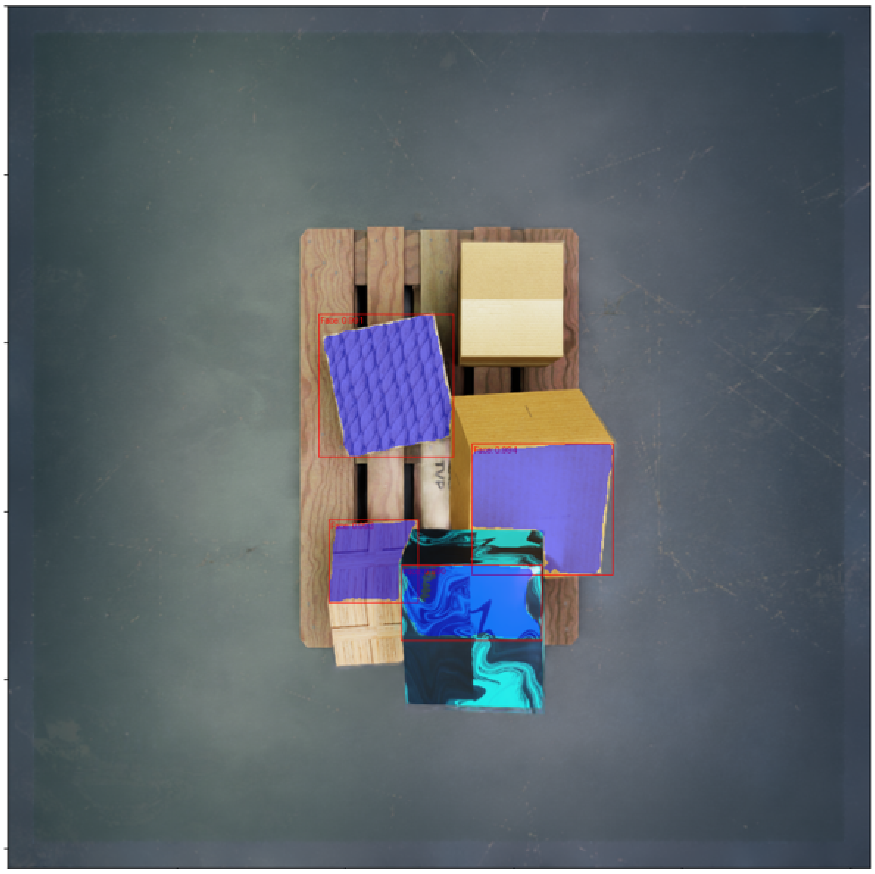


Figure 37 - Splitted faces

12.2 Real data

The real dataset was tested to establish a benchmark for comparison in subsequent simulations, enabling the evaluation of the performances of synthetic images relative to a model trained with real images.
The performance with real data were high, both when using the full dataset (1000 images for training and 300 for validation) and with smaller subsets.
The quality of real data was evident, as the model performed satisfactorily even with a small amount of data.
This suggests that pretraining the CNN significantly helped, as a complete training would have required a larger dataset.
The labelling for the real dataset followed the goal of the project of recognizing the largest face of a box and pick it correctly with a gripper.
It is important to note that the labelling process for real data differed from that of the synthetic dataset.
Creating a heuristic to generate such annotations would have been complex and would have handling many corner cases, which would have deviated from the main objective of the thesis, which focused on studying NVIDIA Isaac Sim technologies.
The MI.RA / Depalletizer system dataset uniquely served as a reference to compare with the synthetic data performance.

12.3 Mixed data

12.3.1 Synthetic data with real images

The tests described in Chapter 11.2 were conducted to simulate a common scenario where a synthetic dataset is used to substitute for a real one.
The training dataset consisted primarily of synthetic images, with a varying number of real images added to evaluate how their inclusion improved the capabilities of the model.
The results, summarized in Table 2, demonstrated the performance of the model across training sets containing an increasing number of real images, proving valuable insights into the optimal balance between real and synthetic data.

Synthetic images	Real images	mAP @50% IOU	mAP @75% IOU
0	65	0.913	0.724
0	1000	0.978	0.936
250	0	0.709	0.583
250	25	0.852	0.752
250	50	0.880	0.815
250	65	0.916	0.842

Table 2 - mAP over testing set of the best model

Table 3 shows the values of maximum Precision and maximum Recall with the validation set for each best model.

Synthetic images	Real images	Max Precision	Max Recall	F1 Score	Threshold
0	1000	0.908	0.930	0.919	0.785
250	0	0.584	0.530	0.556	0.963
250	25	0.727	0.730	0.728	0.901
250	50	0.786	0.700	0.740	0.937
250	65	0.829	0.790	0.809	0.943

Table 3 - Precision and Recall values

The analysis of the results revealed that testing a model trained exclusively on synthetic images against a real dataset yielded lower performance compared to a model trained on real images. This outcome was expected, as the model was trained on the feature of synthetic data, while the validation set consisted of real images, the two distinct domains.
An important factor to consider was the difference in annotation practises:

- Real images followed the rules outlined in Chapter 10.1.
- Synthetic images included annotations for all visible faces of each box.

This discrepancy could have influenced the results, as the model could have correctly recognized a face that was not annotated in the real dataset.
In such cases, the recognition was market as a False Positive when it was, in fact, a True Positive.
When real images were incrementally added to the training set, the behaviour of the model shifted, becoming to be more "restrictive".
This change was evident in the qualitative analysis of the result because some faces of the boxes were not detected.
Performance improved progressively, both for IoU at 75% than IoU at 50% thresholds.
Precision and Recall values also increased with the addition of real images.
Figure 39 shows a comparison among the PR Curves of the simulations, with a highlight of a case in which a training was conducted with only 65 real images, without synthetic ones.
It is possible to recognize the improvements in performances, with the Area Under the Curve (AUC) increasing at each injection.

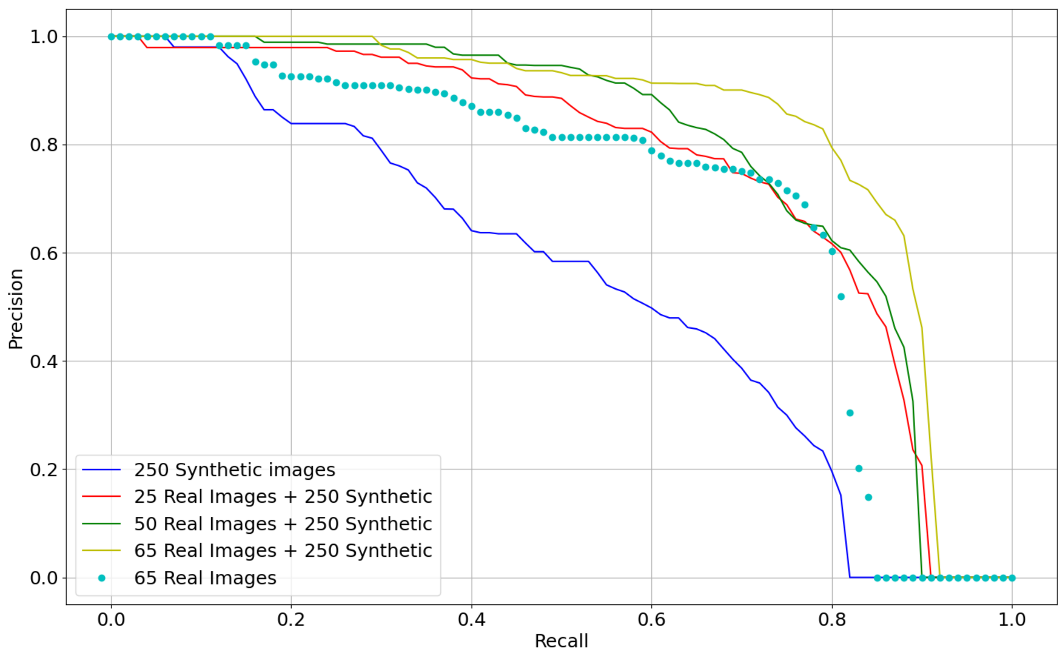


Figure 39 - PR Curve comparison

If the PR curve gets close to the top-right corner of the plot it means that both Precision and Recall are high across most threshold settings, with the ideal point representing a classifier with no false positives or negatives.

The significant result is that the gradual injection of real images in the synthetic dataset improved the performance of the model, highlighting that 65 real images alone did not give the same performance of 65 real images plus 250 synthetic images.

By analysing the results from each testing step, it was evident that the model adapted to the annotation rules of the real dataset, learning to disregard certain faced that should not be recognized.

12.3.2 Real images with synthetic data

The test conducted in Section 11.3 simulated a scenario where synthetic data could have been used to enhance the performance of a real dataset (e.g., in case where certain corner cases may not be present in the real data).

The idea was to create a subset of real images from the complete dataset used in Section 10.2 and evaluate the effect of adding synthetic images.

However, the addition of synthetic images did not lead to an improvement in mAP, with the performance trend remaining nearly stable, as shown in Figure 40.

This figure presents a comparison of the mAP during training process for the three different datasets.

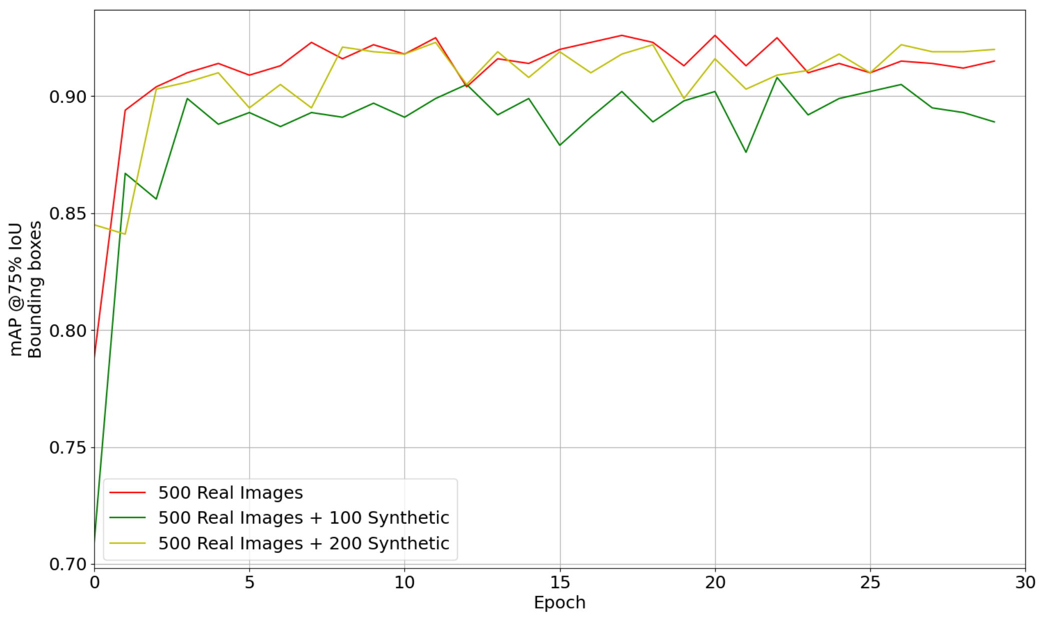


Figure 40 - Comparison of the mAP during training for the three models (highlighted in the legend of the graph)

Figure 41 presents a comparison of the Precision-Recall curves over the validation dataset, for the three best-performing models.

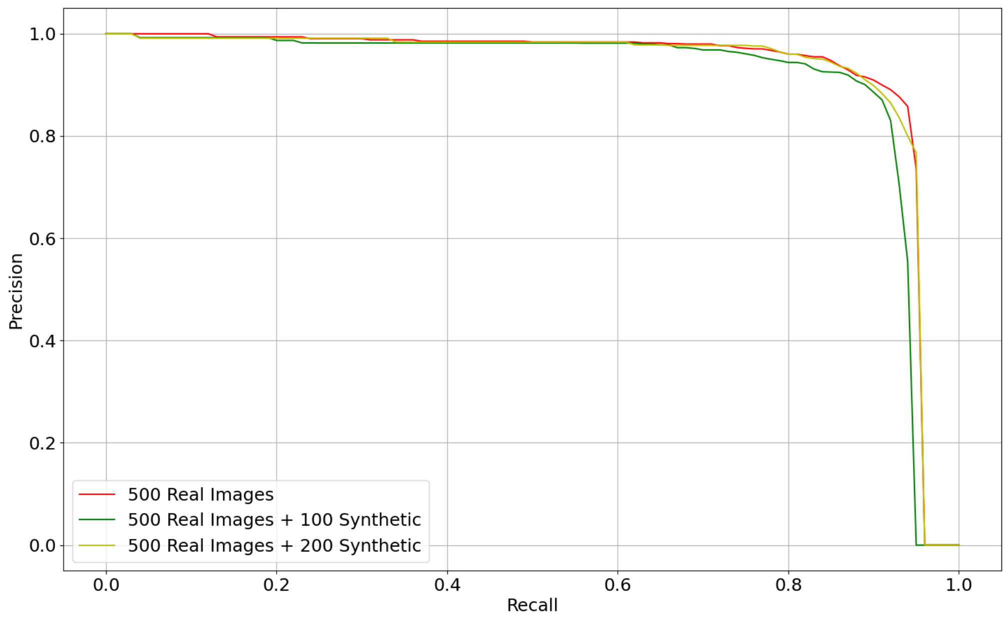


Figure 41 - Comparison of the Precision-Recall curve of the three best models (highlighted in the legend of the graph) over the validation dataset

The three curves are nearly identical, especially the red curve (500 Real Images) and the yellow curve (500 Real Images + 200 Synthetic Images).

The reason could be explained by two main factors:

- The real dataset was of high quality, capturing a wide range of real-life situations and corner cases.

- The CNN, being pre-trained for object detection, was already capable of recognizing patterns and shapes. As a result, only a small amount of real data was needed for fine-tuning.

The key takeaway is that the goal of the thesis was not to improve the quality of the benchmark dataset, but to investigate the performance of a synthetic dataset created via NVIDIA Isaac Sim. Despite the addition of synthetic images to the dataset, there was no significant decrease in performance.

In fact, the results remained stable, indicating that a well-constructed synthetic dataset can effectively support the CNN training process in an industrial environment.

13 MI.RA/OnePicker Dataset

13.1 Introduction

The goal of this thesis was to establish a method for gaining experience with synthetic data and photorealistic simulation environments.
After the results in Chapter 11, the objective of the company was to create a dataset for another application.
Instead of testing the dataset directly, the focus was on performing a qualitative analysis.
In the future, the company plans to test the dataset using their proprietary technologies

13.2 Dataset definition

To better simulate real-world scenarios, the 3D objects used in the simulation were digital twins of those from YCB set
This set includes everyday objects with diverse shapes, sizes, textures, weight, and rigidity, making it ideal for benchmarking robotic manipulation tasks[41].
The desired outputs from the simulation were:

- RGB image
- Instance Segmentation
- Depth camera image
- Point cloud
- Camera parameters

13.3 Environment description

The environment consisted in a plastic bin placed on the ground plane table, with a camera mounted above the bin.
To simulate a broader range of scenarios, both the lighting conditions and the bin model were randomized.
Figure 42 and Figure 43 show an example of the RGB output.

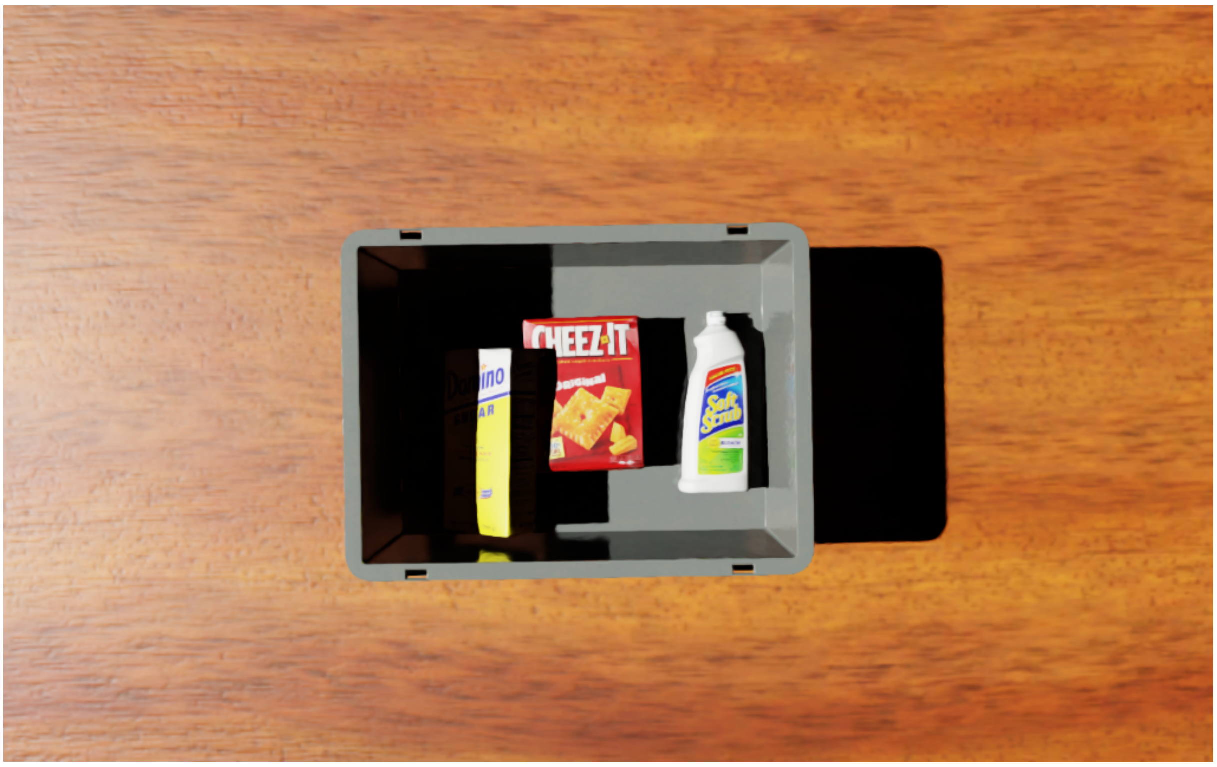


Figure 42 - Example 1 of the scene from the OnePicker dataset creation

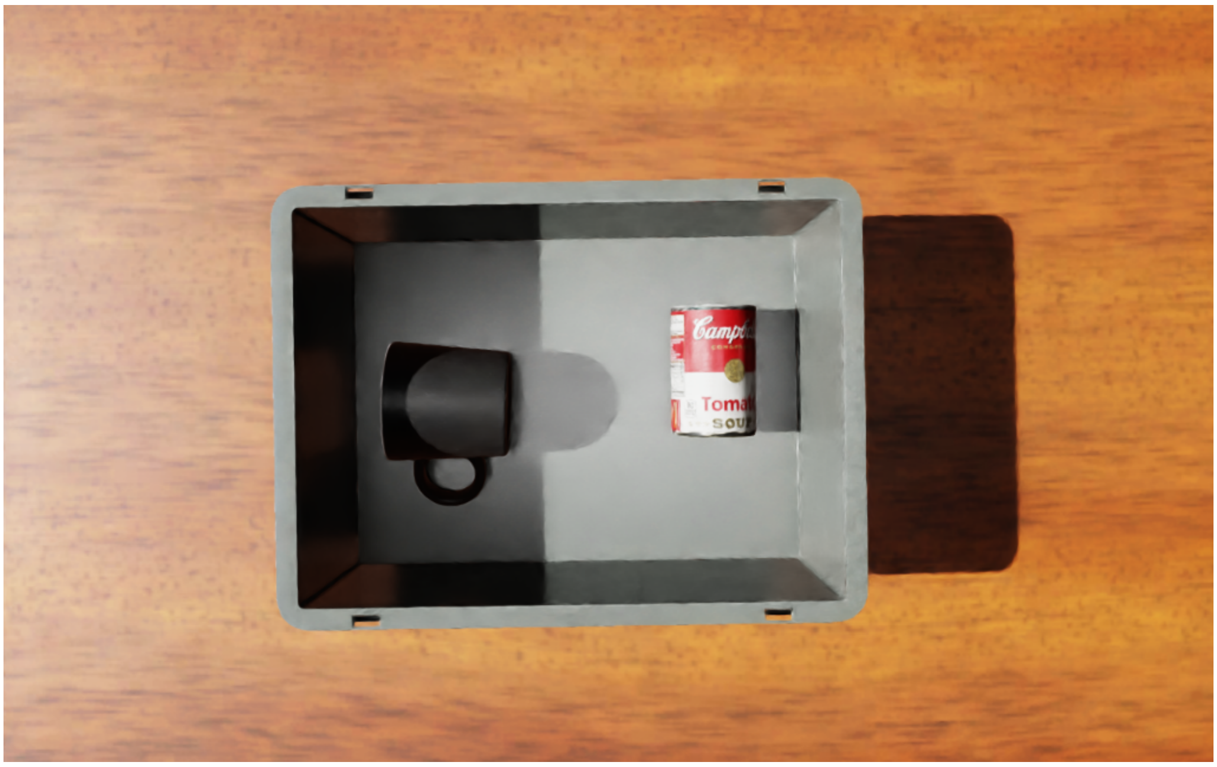


Figure 43 - Example 2 of the scene from the OnePicker dataset creation

13.4 Procedure

For this final section of the thesis, the objective was to perform a qualitatively analysis of output files generated using NVIDIA Isaac Sim.

This analysis aimed to evaluate the extent of information that can be obtained with synthetic data.

The MI.RA/OnePicker was chosen as the benchmark for a specific reason: it relies on the analysis of RGB and depth images to determine the precise spatial location of objects.

In this context, the focus was to extract the intrinsic parameters of the camera used within the simulation scene in NVIDIA Isaac Sim.

This camera information is crucial for computing the exact location of the objects in the bin, enabling the robot to use its end-effector for accurate grasping.

An image can be presented as 2D matrix where each (x, y) coordinate corresponds to a (x, y, z) point in the 3D world

The transformation from 3D world coordinates to 2D image coordinates is described by the formula:

$$w\begin{bmatrix}x\\y\\1\end{bmatrix} = P\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}, with\ P = K[R \cdot t]$$

Where:

- $w\begin{bmatrix}x\\y\\1\end{bmatrix}$ represent the 2D image coordinates, with w as scale factor
- $\begin{bmatrix}X\\Y\\Z\\1\end{bmatrix}$ represents 3D world points.
- P, the camera matrix, maps the 3D world scene into the 2D image plane, and is composed as $P = K[R \cdot t]$, where K is the intrinsic matric and $[R \cdot t]$ represents the extrinsic rotation and translation.

World points are transformed into camera coordinates using the extrinsic parameters, while camera coordinates are projected onto the image plane using the intrinsic parameters.

Extrinsic parameters consist of:

- Rotation (R)
- Translation (t)

Intrinsic parameters include:

- focal length
- Optical centre
- Skew coefficient

These are encapsulated in the intrinsic matrix K:

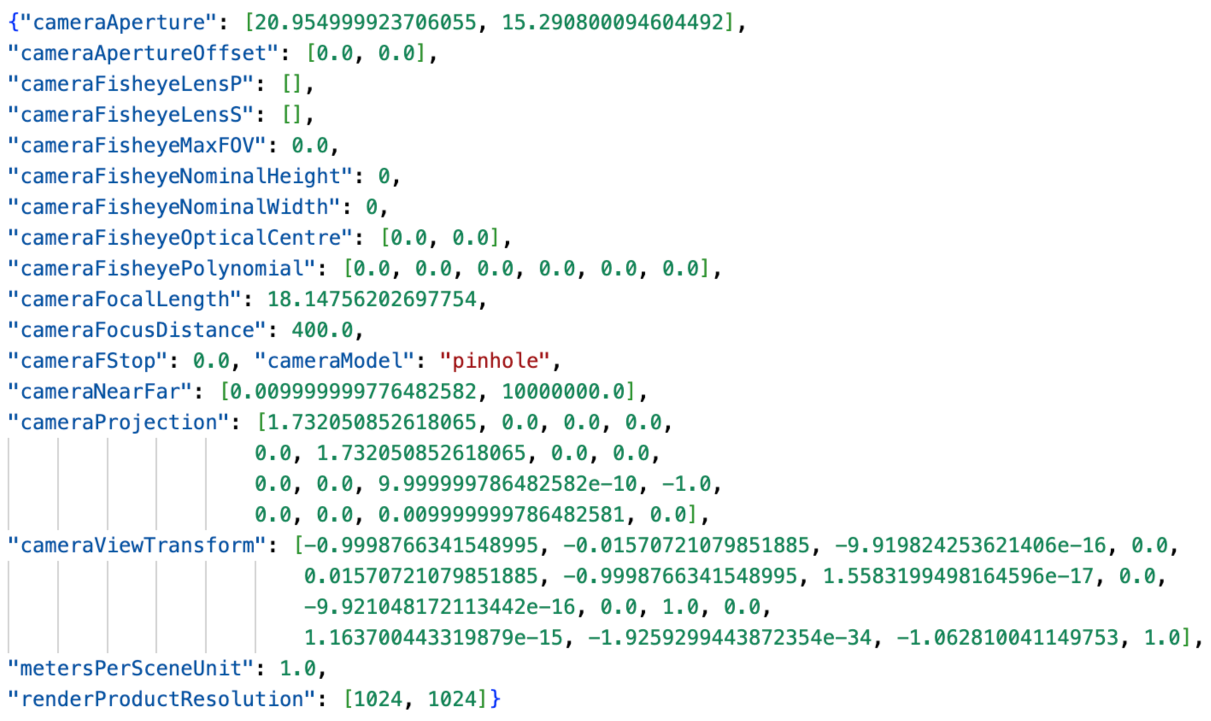
$$K = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}$$

Where:

- f_x, f_y: focal lengths along the x- and y-axes, respectively.
- c_x, c_y: principal points along the x- and y-axes, respectively.
- s: skew coefficient [42].

Using *omni.replicator.core* annotators, it was possible to save the camera parameters as a *.json* file for each simulation iteration.

An example of the output of the file is shown in Figure 44.



```
{"cameraAperture": [20.954999923706055, 15.290800094604492],
"cameraApertureOffset": [0.0, 0.0],
"cameraFisheyeLensP": [],
"cameraFisheyeLensS": [],
"cameraFisheyeMaxFOV": 0.0,
"cameraFisheyeNominalHeight": 0,
"cameraFisheyeNominalWidth": 0,
"cameraFisheyeOpticalCentre": [0.0, 0.0],
"cameraFisheyePolynomial": [0.0, 0.0, 0.0, 0.0, 0.0, 0.0],
"cameraFocalLength": 18.14756202697754,
"cameraFocusDistance": 400.0,
"cameraFStop": 0.0, "cameraModel": "pinhole",
"cameraNearFar": [0.009999999776482582, 10000000.0],
"cameraProjection": [1.732050852618065, 0.0, 0.0, 0.0,
                     0.0, 1.732050852618065, 0.0, 0.0,
                     0.0, 0.0, 9.999999786482582e-10, -1.0,
                     0.0, 0.0, 0.009999999786482581, 0.0],
"cameraViewTransform": [-0.9998766341548995, -0.01570721079851885, -9.919824253621406e-16, 0.0,
                        0.01570721079851885, -0.9998766341548995, 1.5583199498164596e-17, 0.0,
                        -9.921048172113442e-16, 0.0, 1.0, 0.0,
                        1.163700443319879e-15, -1.9259299443872354e-34, -1.062810041149753, 1.0],
"metersPerSceneUnit": 1.0,
"renderProductResolution": [1024, 1024]}
```

Figure 44 - Example of the .json file with Camera Parameters from Replicator

The parameters of interest for this section were:

- cameraAperture: an array containing horizontal and vertical aperture of the camera in millimetres.
- cameraFocalLength: The focal length of the camera, also expressed in millimetres.

Using these two parameters, the K matrix could be computed in pixel dimensions.
With this, the objective of this section was fulfilled.
The goal was to extract the relevant information about the camera in the simulated scene.
In the future, COMAU can use these parameters to derive the K matrix and subsequently compute the position of objects in the scene, enabling precise robotic manipulation.

14 Conclusions

The aim of the thesis was to evaluate the performances of NVIDIA Isaac Sim as Synthetic Data Generator for possible industrial applications, using real-world examples as benchmark.
The results obtained in Section 11.2 align with expectations.
From an analytical perspective, there was a slight performance improvement when using a mixed dataset (real and synthetic images) compared to the dataset containing the same number or real images alone.
This result could be an advantage, since the annotation of few images to be used in conjunction with synthetic images requires less time than annotating an entire dataset.
While the improvement was marginal, it is important to note that the goal was not to enhance the performance of the real dataset, but rather to compare the impact of synthetic data against real data and assess its feasibility for industrial applications.
Results from Section 11.3 demonstrated no significant improvement in performance when synthetic data were added to a dataset containing 500 real images.
This trend indicates that synthetic images were most beneficial when real data were limited.
As the quantity of real images increased, the addition of synthetic images had no significant effects on the performance, but it is important to highlight the quality of the real dataset used as benchmark.
The qualitative analysis in Section 13.4 highlighted the extensive potential of the NVIDIA Isaac Sim environment.
This was exemplified by the successful extraction of camera parameters from the simulation.
Looking ahead, COMAU has the opportunity to delve further into synthetic data generation using NVIDIA Isaac Sim.
This can help enrich their datasets, digitalize technologies for preliminary studies, conduct in-depth analyses, and support various other applications.
To maximize the utility of synthetic data, domain adaptation techniques must be developed to bridge the Sim-to-Real gap, ensuring better interoperability between simulated and real-world datasets.

Appendix – Images from testing

This appendix contains images from the testing phase of the real dataset with the model from Section 11.2.4, trained with 250 synthetic images and 65 real images.

Figure 45, Figure 46, Figure 47, and Figure 48 shows the capabilities of the model in detecting the faces of the boxes in different light conditions and with different shadows and disturbing elements.

Figure 49 contains a little issue, as the metallic mount on the table is misclassified as a face.

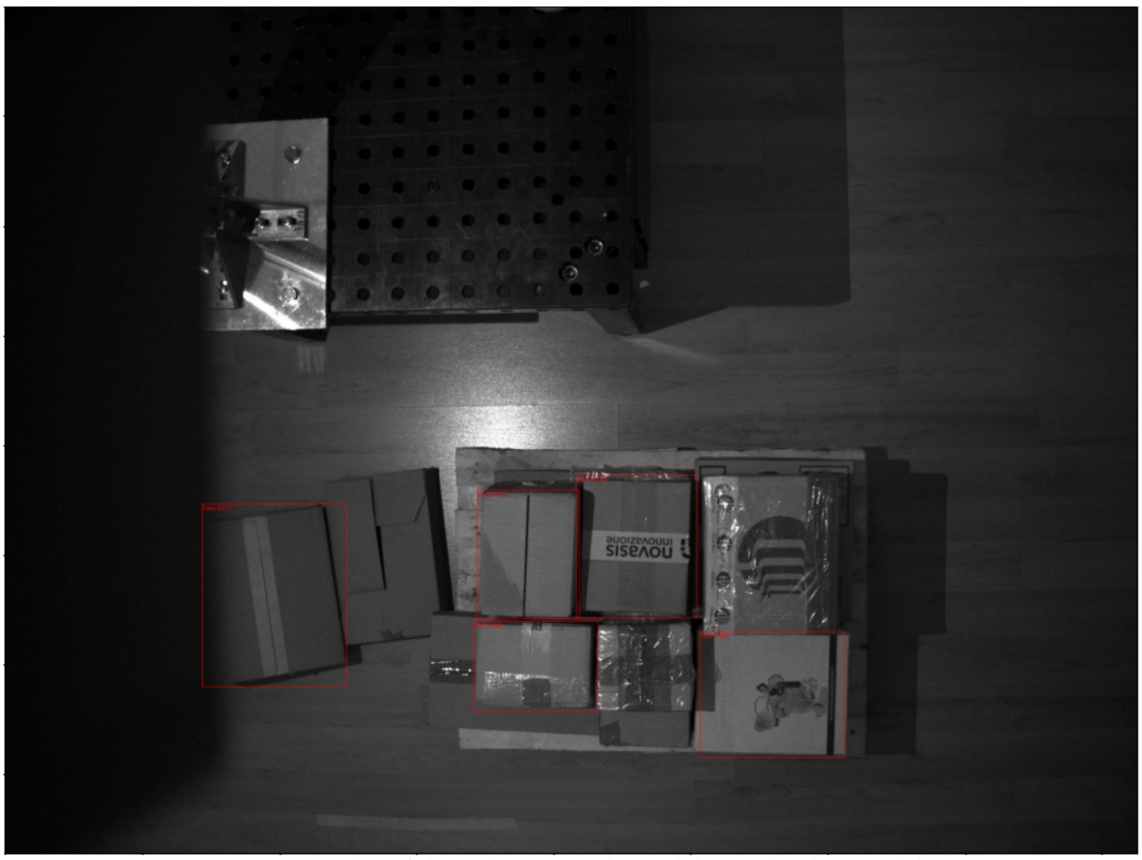


Figure 45 - Example 1 of object detection with a model trained with a mixed dataset

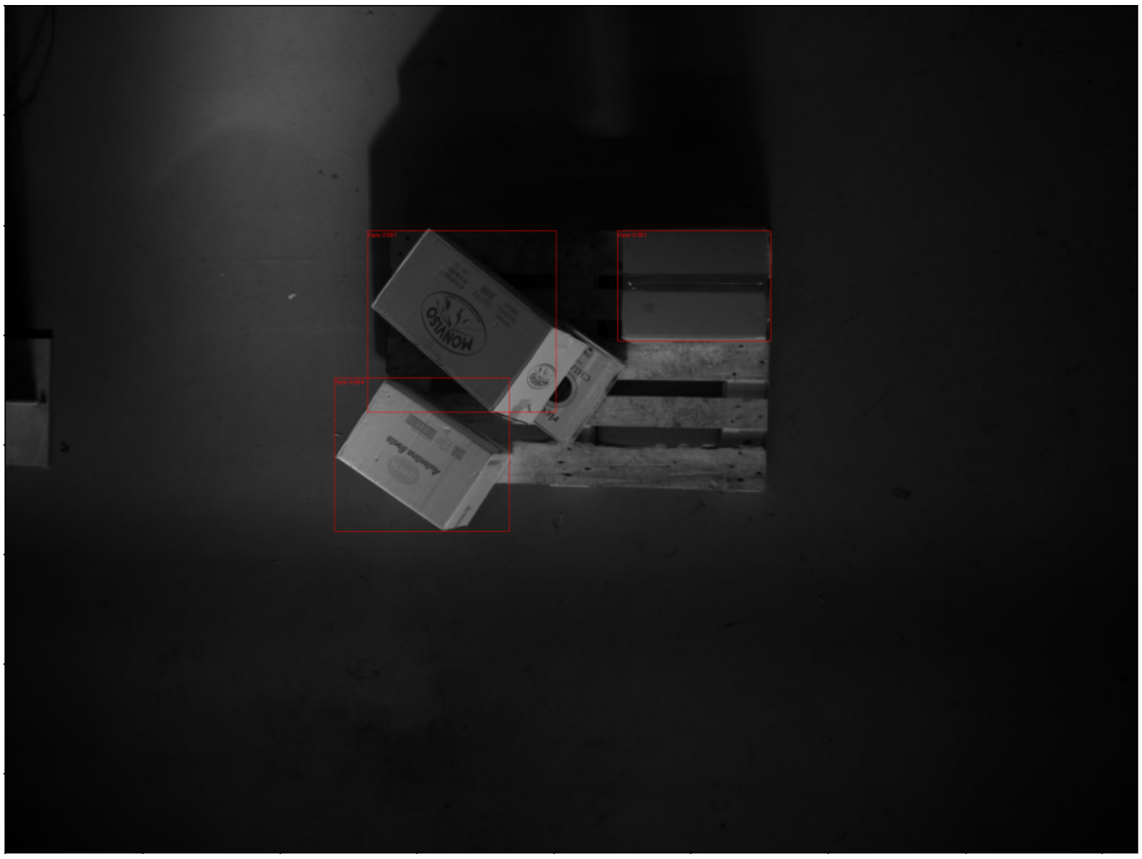


Figure 46 - Example 2 of object detection with a model trained with a mixed dataset

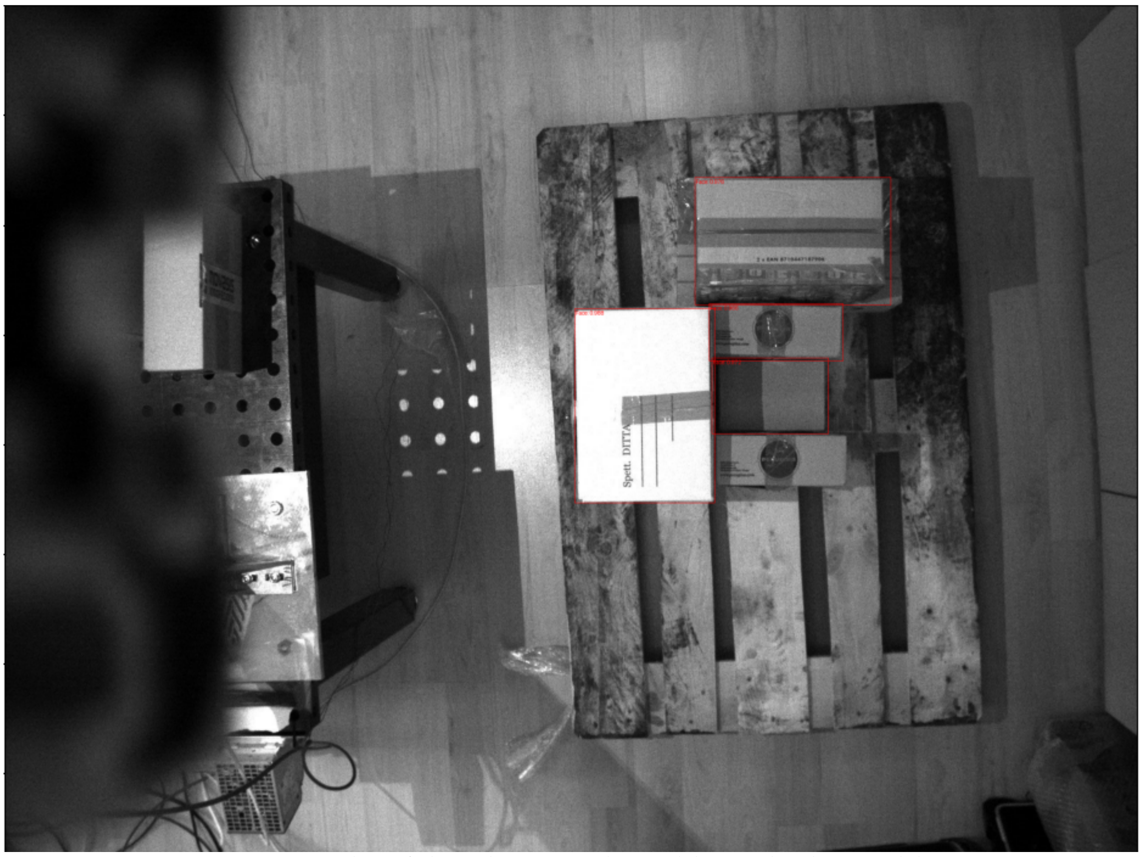


Figure 47 - Example 3 of object detection with a model trained with a mixed dataset

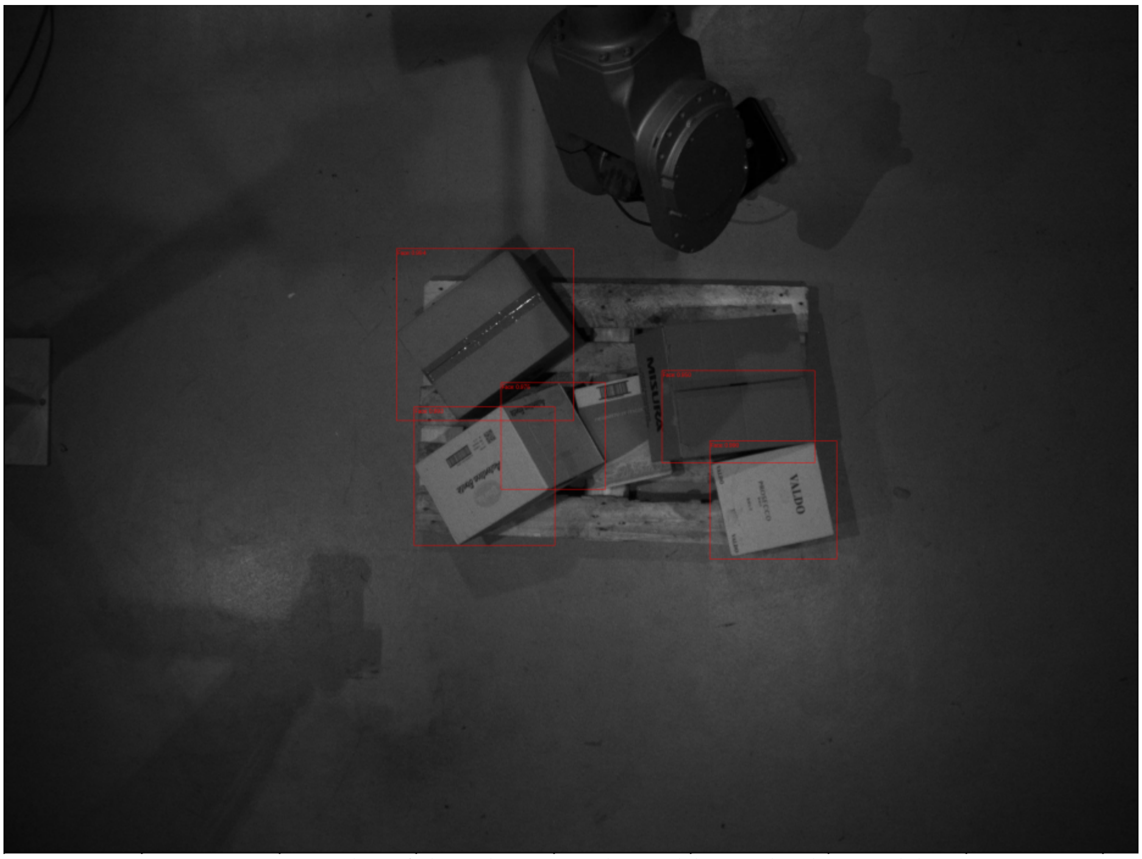


Figure 48 - Example 4 of object detection with a model trained with a mixed dataset

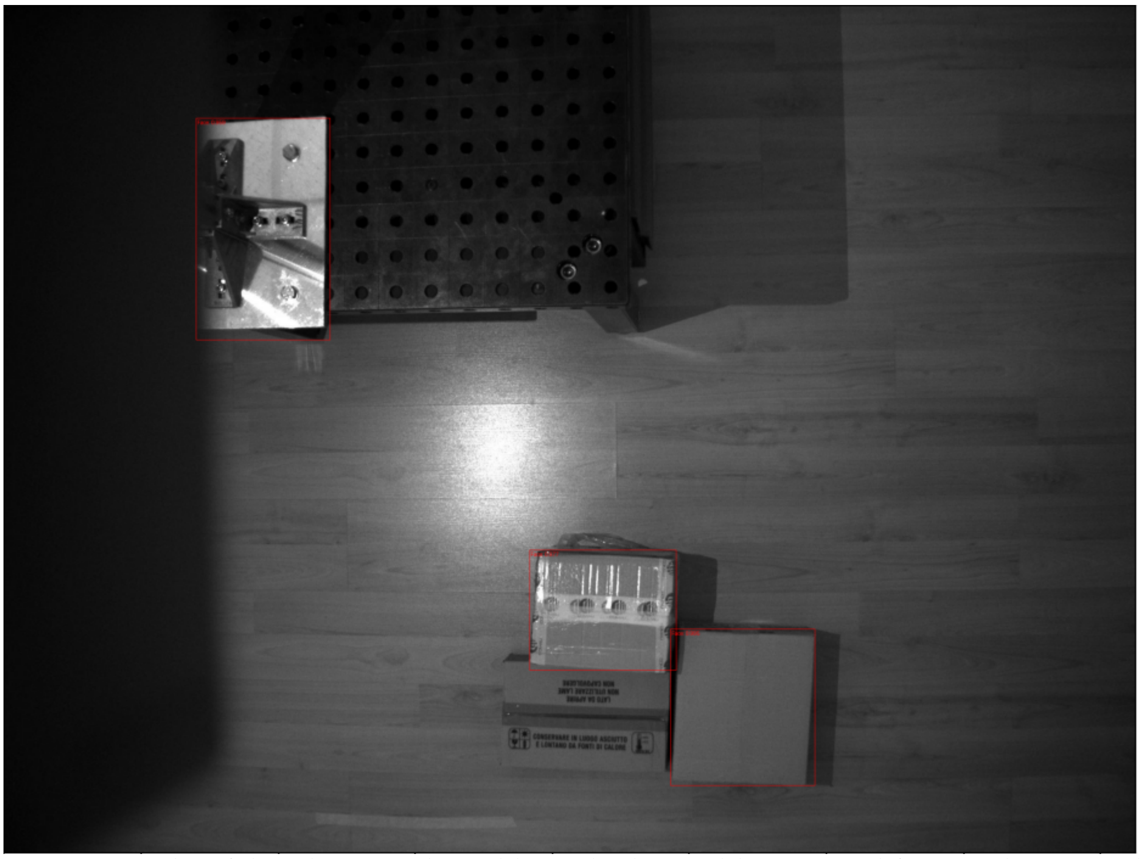


Figure 49 - Example 5 of object detection with a model trained with a mixed dataset. In top-left position the mount on the table is misclassified as a box

References

1. [ONLINE] Available at: https://www.ibm.com/it-it/topics/artificial-intelligence.
2. Joshi A V. *Machine Learning and Artii Cial Intelligence*.
3. Shanmugamani Rajalingappaa. *Deep Learning for Computer Vision : Expert Techniques to Train Advanced Neural Networks Using TensorFlow and Keras*. Packt Publishing; 2018.
4. Suresh S, Raju C, Chaitanya GSSR. *An Empirical Analysis of Leaky Integrate and Fire Neuron Model*. www.ijert.org
5. F. ROSENBLATT. *THE PERCEPTRON: A PROBABILISTIC MODEL FOR INFORMATION STORAGE AND ORGANIZATION IN THE BRAIN.*; 1958.
6. [ONLINE] Available at: https://it.mathworks.com/discovery/neural-network.html.
7. Terven J, Cordova-Esparza DM, Ramirez-Pedraza A, Chavez-Urbiola EA, Romero-Gonzalez JA. Loss Functions and Metrics in Deep Learning. Published online July 5, 2023. http://arxiv.org/abs/2307.02694
8. Ruder S. An overview of gradient descent optimization algorithms. Published online September 15, 2016. http://arxiv.org/abs/1609.04747
9. [ONLINE] Available at: https://www.ibm.com/topics/gradient-descent#:~:text=Gradient%20descent%20is%20an%20optimization,between%20predicted%20and%20actual%20results.
10. [ONLINE] Available at: https://www.ibm.com/topics/gradient-descent.
11. Pothuganti S. Review on over-fitting and under-fitting problems in Machine Learning and solutions. *Article in International Journal of Advanced Research in Electrical Electronics and Instrumentation Engineering*. Published online 2018. doi:10.15662/IJAREEIE.2018.0709015
12. [ONLINE] Available at: https://nafizshahriar.medium.com/what-is-convolutional-neural-network-cnn-deep-learning-b3921bdd82d5.
13. Yang K. *Indoor Scene Understanding for the Visually Impaired Based on Semantic Segmentation*. www.kit.edu
14. [ONLINE] Available at: https://www.ibm.com/topics/object-detection#f01.
15. [ONLINE] Available at: https://www.gartner.com/en/newsroom/press-releases/2022-06-22-is-synthetic-data-the-future-of-ai.
16. Emam K El. *Accelerating AI with Synthetic Data Generating Data for AI Projects*.
17. Emam K El. *Accelerating AI with Synthetic Data Generating Data for AI Projects*.

18. Deloitte Insights. *State of AI in the Enterprise, 2nd Edition.*; 2018.
19. MIT Technology Review Insights. The Global AI Agenda: Promise, Reality, and a Future of Data Sharing. Published online 2020.
20. Khaled El Emam, Elizabeth Jonker, Ester Moher, Luk Arbuckle. A Review of Evidence on Consent Bias in Research," American Journal of Bioethics 13, no. 4. Published online 2013.
21. Shermeyer J, Hossler T, Van Etten A, Hogan D, Lewis R, Kim D. RarePlanes: Synthetic Data Takes Flight. Published online June 4, 2020. http://arxiv.org/abs/2006.02963
22. [ONLINE] Available at: https://developer.nvidia.com/isaac/sim.
23. [ONLINE] Available at: https://developer.nvidia.com/omniverse.
24. [ONLINE] Available at: https://nvidia-omniverse.github.io/PhysX/physx/5.4.2/.
25. [ONLINE] Available at: https://docs.omniverse.nvidia.com/extensions/latest/ext_onshape.html#isaac-onshape-importer.
26. [ONLINE] Available at: https://docs.omniverse.nvidia.com/isaacsim/latest/advanced_tutorials/tutorial_advanced_import_urdf.html#isaac-sim-app-tutorial-advanced-import-urdf.
27. [ONLINE] Available at: https://docs.omniverse.nvidia.com/isaacsim/latest/advanced_tutorials/tutorial_advanced_import_mjcf.html#isaac-sim-app-tutorial-advanced-import-mjcf.
28. [ONLINE] Available at: https://www.pixar.com/openusd.
29. [ONLINE] Available at: https://openusd.org/release/intro.html.
30. Corporation N. *Exploring Digital Twins: Key Insights From Five Industry Leaders | NVIDIA*.
31. [ONLINE] Available at: https://www.youtube.com/watch?v=LUnZXBL_lqA.
32. [ONLINE] Available at: https://omniverse-content-production.s3-us-west-2.amazonaws.com/Assets/Isaac/Documentation/Isaac-Sim-Docs_2022.2.1/extensions/latest/ext_replicator.html.
33. [ONLINE] Available at: https://docs.omniverse.nvidia.com/extensions/latest/ext_replicator/annotators_details.html.
34. [ONLINE] Available at: https://docs.omniverse.nvidia.com/py/replicator/1.11.16/source/extensions/omni.replicator.core/docs/API.html.

35. [ONLINE] Available at: https://www.comau.com/it/competencies/robotics-automation/depalletizer/.
36. [ONLINE] Available at: https://www.comau.com/en/our-offer/robotics-automation/vision-assisted-picking/.
37. [ONLINE] Available at: https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html#object-detection-and-instance-segmentation-model-for-pennfudan-dataset.
38. He K, Gkioxari G, Dollár P, Girshick R. Mask R-CNN. Published online March 20, 2017. http://arxiv.org/abs/1703.06870
39. Lin TY, Maire M, Belongie S, et al. Microsoft COCO: Common Objects in Context. Published online May 1, 2014. http://arxiv.org/abs/1405.0312
40. Rahman A, Wang Y. *Optimizing Intersection-Over-Union in Deep Neural Networks for Image Segmentation*.
41. [ONLINE] Available at: https://www.ycbbenchmarks.com.
42. [ONLINE] Available at: https://it.mathworks.com/help/vision/ug/camera-calibration.html.