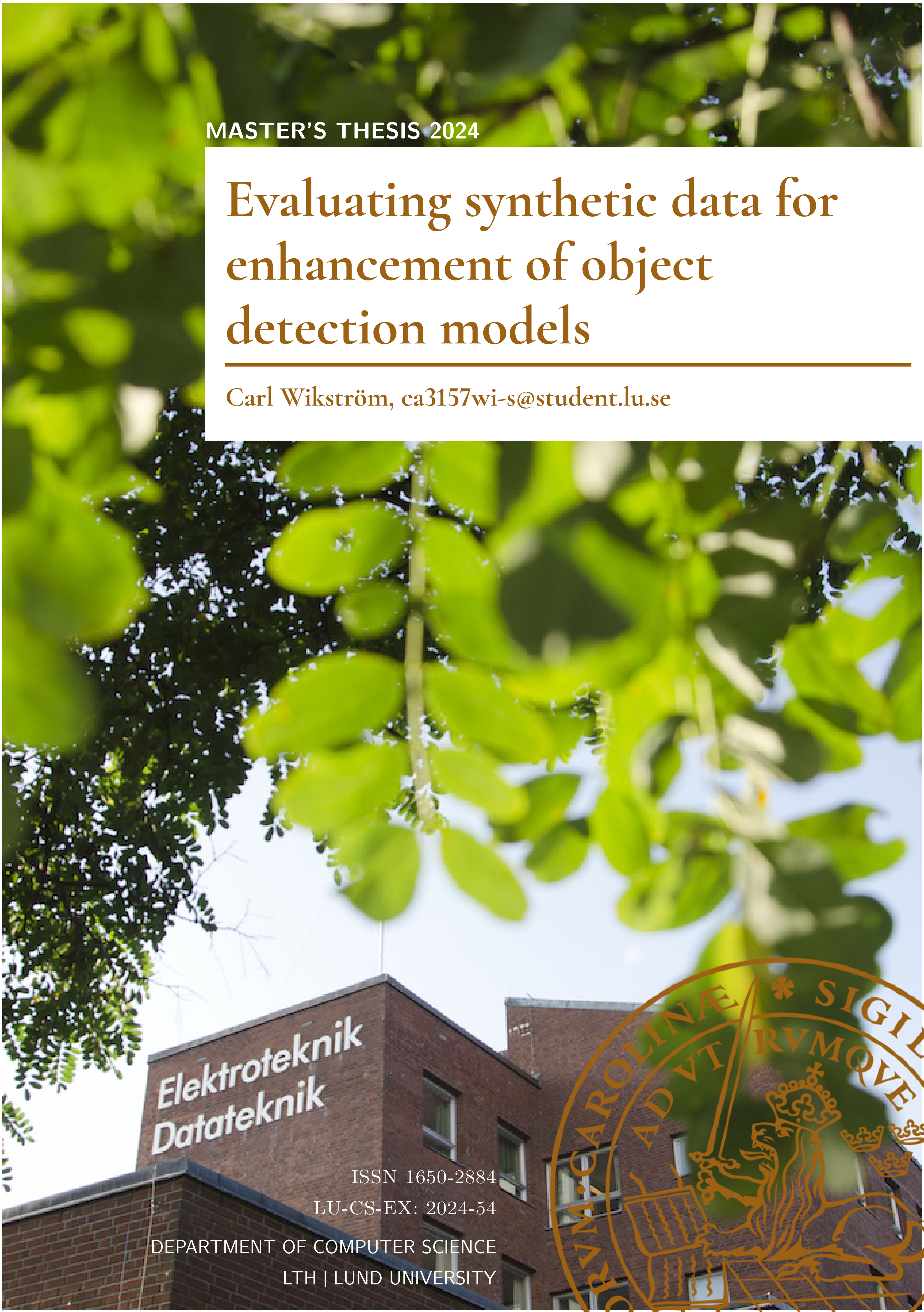


MASTER'S THESIS 2024

Evaluating synthetic data for enhancement of object detection models

Carl Wikström, ca3157wi-s@student.lu.se

ISSN 1650-2884
LU-CS-EX: 2024-54

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

LU-CS-EX: 2024-54

Evaluating synthetic data for enhancement of object detection models

Carl Wikström, ca3157wi-s@student.lu.se

Evaluating synthetic data for enhancement of object detection models

Carl Wikström ca3157wi-s@student.lu.se

August 30, 2024

Master's thesis work carried out at Verisure Innovation.

Supervisors: Patric Fröjd, `patrik.frojd@verisure.com`
Volker Krueger, `volker.krueger@cs.lth.se`

Examiner: Jacek Malec, `jacek.malec@cs.lth.se`

Abstract

This thesis investigates the potential of using synthetic data to enhance object detection models in security surveillance contexts. Synthetic data was generated with the Unity Perception package and evaluated using real-world security footage. The study compared these results with those from a model trained on an equivalent number of COCO dataset images. Findings indicate that while synthetic data alone did not match the performance of real data, fine-tuning the model with a small target domain dataset allowed synthetic data to perform equivalently. The research highlights the benefits of synthetic data, especially when real data is scarce, and shows promise for detecting specific behaviors, such as crawling versus standing individuals. This study contributes to the ongoing discussion about the practicality of synthetic data in surveillance applications, underscoring both the challenges and opportunities of leveraging synthetic datasets for enhancing object detection models in diverse security scenarios.

Keywords: Object detection, Synthetic Data, Unity Perception, YOLOv9-c, Security Surveillance

Acknowledgements

I would like to extend my sincere thanks to my supervisor at Verisure, Patric Fröjd, for his guidance, dedicated time, and constant availability throughout this project. His mentorship has been instrumental in the continuous development and experimentation during the thesis.

I would also like to thank my supervisor at the Faculty of Engineering at Lund University, Volker Krueger, for his help and assistance during the thesis. His expertise and support have been crucial in guiding me through this thesis.

Contents

Chapter 1
Introduction

This chapter serves as a brief introduction to the thesis and the central problem that will be investigated. It highlights the problem that companies like Verisure are facing with constraints posed by data sparsity and privacy regulations. The central questions that the thesis aims to address are introduced along with previous research within the topic.

1.1 Background

Verisure is one of Sweden's largest security companies providing services to companies and private residents alike. Their services include complete home alarm systems that entail smart cameras, sensors and smoke detectors that are interconnected to give a complete security solution. Previously, 1 dimensional infra red signals have been part of the solution that triggers their alarms. These solutions sometimes faced challenges in distinguishing between the motion of, for example, a dog and a crawling person, which has driven the development of more advanced detection methods. Today a more sophisticated approach, namely object detection models, are investigated for use in their motion detection systems deployed in their security cameras.

Object detection, a critical task in computer vision, has evolved significantly with the advent of deep learning, offering impressive capabilities in recognizing and localizing objects within images[15]. These models are today used in numerous areas, such as medical technology, autonomous driving and security [23][19]. The performance of these models is however heavily influenced by the quality and diversity of the training datasets, with biases and skewness often resulting in challenges when it comes to generalization across varied real-world scenarios [34]. One such situation is within security where many specific scenarios are crucial to identify, but where the sparsity of data is a constraining factor. Because of privacy regulations such as GDPR, real world data gathering is very restricted. Large scale public datasets can be used to train a general object detection model that performs adequately on general tasks but when faced with a difficult task that has limited training data it often does not perform as well. Verisure is continuously working to overcome the challenges posed by data sparsity. Collecting real-world data can be both expensive and time-consuming, making it challenging to gather sufficient data for training robust object detection models for various security scenarios. One particular scenario of interest is the detection of crawling people. This scenario is critical for security surveillance, yet it is difficult to capture enough real-world instances to train an effective model. Figure 1.1 illustrates a typical Verisure use case where a state-of-the-art public model like YOLOv9-e mistakes a human for a dog.

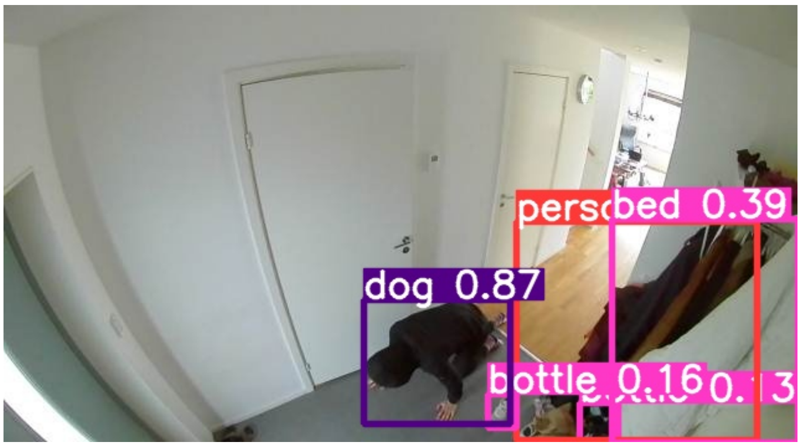


Figure 1.1: The public YOLOv9-e models prediction of staged Verisure use case image

In this context, synthetic data can offer a promising solution. By generating synthetic images, it is possible to create a diverse and extensive dataset that can help train object detection models to recognize and accurately identify objects in situations where real data is scarce or difficult to obtain. This thesis aims to evaluate the effectiveness of synthetic data in training object detection models for security surveillance applications.

In response to the challenges faced in traditional data collection, generative AI has emerged as a possible alternative. Generative Adversarial Networks (GANs) are a collection of AI algorithms frequently explored within this domain. Introduced by Goodfellow et al. in 2014 [10], GANs consist of two neural networks: the generator and the discriminator, which compete against each other. The generator aims to create data that statistically mimics the training dataset, while the discriminator endeavors to distinguish between generated and real images. This adversarial approach enables GANs to produce highly realistic images resembling real-world data. However, while GANs have demonstrated success in various applications, they are not always effective. One scenario where generative models often struggle is in generating entire human bodies [16]. Such models may produce images with inaccuracies in human anatomy, resulting in non-realistic outputs that may not generalize well to real-world data.

Another popular alternative to generative AI is the use of game engines to generate large sets of labeled data [5] [26] [31]. This approach has evolved to an impressive level in recent years, and in 2021, Unity released their perception package that offers a highly customizable toolkit and a high degree of randomization [1]. Advantages of using a game engine in generating synthetic data include the level of control over the generated data and the fact that the data is perfectly labeled upon creation.

However, a potential challenge when using this synthetic data is the so-called domain gap, which can result in poor model performance [30]. The domain gap refers to the discrepancy between the distributions of synthetic data and real-world data. This gap can arise from various factors in the data and can lead to models trained on synthetic data underperforming when applied to real-world tasks. While numerous ways of bridging this domain gap exist, they are not without challenges and will not be explored in this thesis.

In this thesis, the focus will be on evaluating the effectiveness of synthetic data generated through game engines for training object detection models in security surveillance footage. By exploring state-of-the-art models and methodologies, the aim is to provide a proof of concept for the use of synthetic data to enhance Verisure's object detection capabilities. Shedding light on the opportunities and challenges in utilizing synthetic data for security applications.

1.2 Research questions & limitations

The thesis aims to answer the following questions:

- How does synthetic data compare to real data when training an object detection model?
- To what extent can synthetic data enhance the performance of an object detection model in specific security surveillance scenarios?
- How does the performance of an object detection model vary with different types of synthetic data generated for security surveillance scenarios?

The study is limited to examining human detection using the YOLOv9c model. The chosen model and results achieved are specific to this research and do not represent or reflect the models or outcomes used by Verisure. The objective isn't to optimize for the best model performance possible due to computational limitations imposed by dataset sizes and model complexity. Instead, the focus lies on assessing synthetic data, examining its performance relative to real data, and determining its viability in place of real data and in scenarios with limited access to real data.

1.3 Previous work

The subject of generating synthetic data for machine learning models is capturing widespread interest right now. There exist many studies that look at different aspects and explore different methods. This thesis will explore two main areas within computer vision and object detection, namely, generating synthetic data and domain adaptation.

1.3.1 Generating synthetic data using 3-d models

Generating synthetic data using 3-d models such as a game engine is not a new concept and similar applications to our proposed usage have been investigated. Neuhausen et al. explored using synthetic data as a substitute for real data in human recognition tasks on construction sites and found that the synthetic data only slightly decreased performance [28]. Similarly, Lee et al. used a game engine to generate synthetic data to extend the training dataset and found that it substantially enhanced the performance of small object detection in computer vision tasks [20]. These papers illustrate the applicability of synthetic data by generating data that mimics the key features of the real data. The workers are clearly distinguishable by their outfits and the small objects remain consistent in appearance regardless of the environment. Ebadi et al. illustrated a more general approach to the human detection problem using synthetic data [8]. They showed that a Unity perception model can be used to improve the results of a general human detection model when testing on the COCO-2017 test set. However, it's important to note that while this result is interesting for the general human detection problem, it does not guarantee its effectiveness for a more specific target domain. This limitation is highlighted by Madan et al., who argue that training on a diverse set of images may improve out-of-distribution (OOD) testing but can potentially degrade in-distribution performance [24].

1.3.2 Domain adaptation

With the use of synthetic data in training object detection models, the need for sufficient domain adaptation has become clear. Different approaches address this problem, with a distinction between synthetic-to-real refinement and model-based domain adaptation being necessary [29]. Synthetic-to-real adaptation modifies the synthetic data to better align with real-world data. For example, Menke et al. used a GAN-focused approach, while Zhang et al. used a feature adaptation model to reduce data distribution mismatch [25][41]. Although GAN-focused approaches show promise, they are computationally intensive and will not be explored in this thesis. The goal of synthetic-to-real transformation is to bridge the domain gap by making synthetic data mimic the target domain. Enhancing the realism of synthetic images during generation helps ensure they closely replicate the visual properties of the target domain, such as lighting, texture, and noise patterns found in real-world surveillance footage. This preemptive adaptation during image generation has the potential to significantly improve the model's performance when tested on real data. Model-based domain adaptation instead focuses on the training of the model. While ensuring the characteristics and quality of synthetic data is important, the manner in which it is utilized can be equally critical. In this approach, the synthetic data is not altered but the focus is instead placed on the training process and the model structure to ensure adaptation to the target domain [29]. Hinterstoisser et al. claim that state-of-the-art architectures usually consist of pre-trained feature extractor layers followed by task-specific layers [13]. They propose freezing the weights of these feature extractors when training on synthetic data, arguing that they are already sufficient for visual tasks and may be affected poorly by domain shifts. Their approach leads to improved object detection performance compared to full retraining, nearly matching models trained on real data. These findings are, however, countered by numerous other studies that show comparable or even worse results using the technique [38] [37]. Thus, the effectiveness of this approach may vary depending on the dataset used.

Chapter 2
Theory

In this thesis, our primary objective is to assess the efficacy of synthetic data in improving object detection models, particularly within security surveillance scenarios in computer vision. To achieve this, we have selected the YOLOv9 detector as our model of choice for experimentation and validation. YOLOv9 is chosen due to its state-of-the-art performance on benchmark tests, excelling in both speed and accuracy, particularly for real-time applications. The following chapter provides a theoretical foundation essential for understanding the underlying principles and techniques used in object detection models. Starting with neural networks and deep learning, it explores their role in computer vision tasks. Convolutional neural networks (CNNs) are introduced, emphasizing their importance in capturing complex patterns in images. The chapter then delves into object detection, outlining its goals and introducing two-stage and one-stage detector architectures. Evaluation metrics for object detection models are also introduced. Finally, it discusses synthetic data generation and domain adaptation.

2.1 Neural networks & Deep learning

Neural networks have been a key part in the recent advances within machine learning. They are used within various areas such as large language models, computer vision tasks, time-series analysis and many more. When training a neural network an important distinction is made between supervised learning and unsupervised learning. Unsupervised learning is used when there is no clear distinction between input and output in the data. This approach is utilized within a variety of problems areas, such as clustering, dimensional reduction and anomaly detection. In supervised learning the target class y is known and the model can be trained using this relationship between input x and output y.
The building blocks of a neural networks consists of neurons. A neuron takes an input x^* and transforms it into an output y^* through:

$$y^* = g(w^T x^* + b) \tag{2.1}$$

Here w are the weights and b is a bias term for the neuron and the function g is what is called the activation function. The objective of a neural network when using supervised learning is to learn a mapping from an input x to an output y. This is done by learning the optimal value for θ in the function approximation $f(x;\theta)$. Neural networks of this sort are usually represented by composing together many different functions f^*. The resulting chain-like structure of functions is what composes a network that has one layer for each function f^* in the chain. The final layer in the network is called the output layer. The desired end result of the network is as previously stated to produce a approximation of the mapping from the input x to output y. In training of the network it is only specified that this mapping from input x to output $f(x)$ from the output layer is to be approximated. Nothing in the training of the network specifies what the output from the layers in the middle should be, because of this these layers are refereed to as hidden layers.
One common type of layer in neural networks is the fully connected layer. In a fully connected layer, each neuron is connected to every neuron in the previous layer. This means that each neuron in a fully connected layer receives an input that is the weighted sum of all the outputs from the previous layer, followed by the application of the activation function. Fully connected layers are often used towards the end of neural networks, especially for classification tasks, as they help in converting the learned features into a final decision.
The effectiveness of neural networks ultimately depend on the training process. The neurons in each layer of the network are connected by adjustable weights and each layer has an activation function. The objective of the training process is to minimize the loss for all training samples:

$$L(\theta) = \frac{1}{N}\sum_{i=0}^{N} l(y_i, f(x_i, \theta)) \tag{2.2}$$

The weights in the network are updated through a process called backpropagation. The backpropagation is used to minimize this loss function by adjusting the network's parameters. The process involves calculating the gradients of the loss function with respect to the weights and biases in the network. These gradients indicate the direction and magnitude of the adjustments needed to minimize the loss. The parameters are then adjusted for each iteration based on the gradient of the loss function and another parameter called learning rate. The learning rate is multiplied by the gradient of the loss to decide how much the parameters are changed each iteration.

2.2 Deep learning

Deep learning refers to the use of neural networks with many hidden layers in order to find complex relationships and patterns in data. With the growth of big data, deep learning has become increasingly important in order to fully utilize the value that can be found in the data [27]. The term deep stems from the depth of the network, indicating the number of hidden layers in the network. This depth is what allows deep neural networks to achieve success within many different areas such as image processing, speech synthesis, natural language processing and sound generation [18]. While deep neural networks often show impressive results, the training of the networks can be quite complicated. During the backpropagation

vanishing or exploding gradient problems can arise and lead to models that perform very poorly. These problems occur because the gradients become successively very large or small as they propagate back in the network. To tackle this problem techniques such as batch normalization have been suggested. Batch normalization was introduced by Ioffe et al. in 2015 and is designed to improve training stability and convergence of the network during training [17]. It does so by normalizing the inputs of a layer by adjusting and scaling them for each batch. This ensures that inputs to each layer are more consistent across batches which improves stability during training.
Another common problem when training deep neural networks is overfitting to the training data. This occurs when the model learns noise and random fluctuations in the training data which leads to poor generalization to new data. One technique used to reduce overfitting is L2 regularization. This technique simply adds a penalty term to the total loss that is proportional to the weights. This term includes a constant λ that controls the magnitude of the penalty added according to:

$$L(\theta) = L(\theta) + \lambda ||w||^2 \tag{2.3}$$

2.3 Convolutional neural networks

Convolutional neural networks, or CNNs, are a type of deep learning models that have shown impressive results within computer vision especially. First introduced by Lecun et al. CNNs utilize the mathematical convolution operation in order to capture spatial hierarchies and intricate patterns. A convolution consists of a convolutional kernel that is slid across the input feature map. The kernel is moved across the input feature map with a step size that is refereed to as stride. This operation is illustrated in Figure 2.1 where a 3x3 kernel is used.

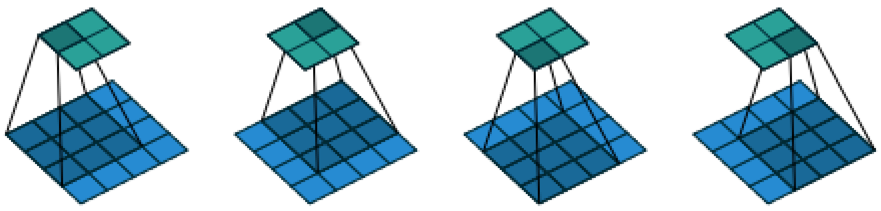


Figure 2.1: (No padding, unit strides) Convolving a 3 × 3 kernel over a 4 × 4 input using unit strides (i.e., i = 4, k = 3, s = 1 and p = 0). (Credits : Dumoulin & Visin [7]).

Another important building block in CNNs is the pooling operation. The pooling operation is used in order to reduce the size of the feature map. Much like the convolution operation, a function window is moved across the feature map. This is done to summarize subregions of the input, often using an average or a max value operation. A 3x3 average pooling operation performed on a 5x5 grid can be seen in Figure 2.2.

CNNs are extensively used in computer vision because of their performance within image processing tasks [36][2][12]. The reason they are so efficient to use in image processing is

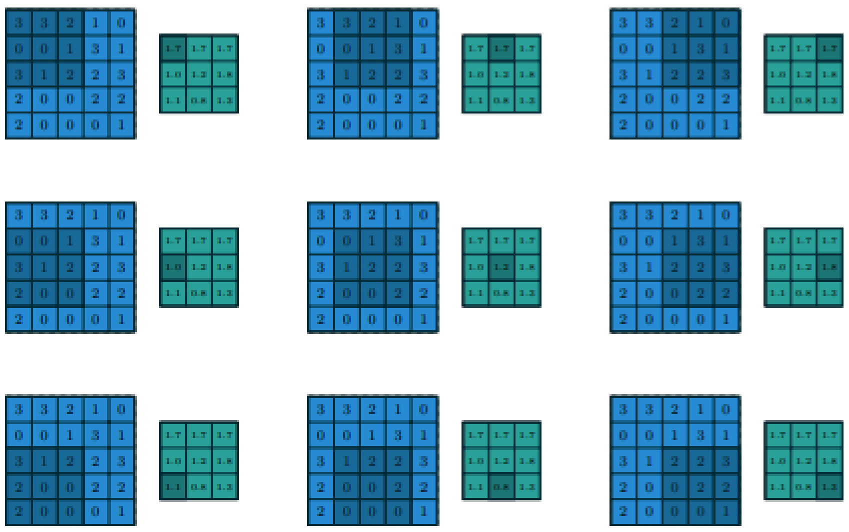


Figure 2.2: Computing the output values of a 3 × 3 average pooling operation on a 5 × 5 input using 1 × 1 strides. (Credits : Dumoulin & Visin [7]).

their ability to learn hierarchical representations of features. This is achieved using a deep network but also by using **filters**. The resulting specific feature map comes from applying a convolution using a specified kernel on the input data. These kernels can detect different features in the images, resulting in a model that, with enough depth, can identify complex patterns and features. An illustration of how filters process images is showed in Figure 2.3.

2.4 Object detection models

Object detection is a critical task within computer vision that, with the development of deep CNNs, has seen significant advances recently. The objective of an object detection model is twofold, firstly it has to locate relevant objects in the image, secondly it then has to correctly classify the object. The localization task consists of identifying and delimiting an area of the image that contains a relevant object. This is done by encapsulating the object inside a rectangular bounding box. The bounding box is often described by a 4-tuple $[x_1, y_1, x_2, y_2]$ where the values describe the position and size of the box. The classification task consists of determining which category a given bounding box, also called region of interest (ROI), belongs to. The ROI is put through a deep CNN that captures patterns, textures and spatial information that is relevant to the object within the ROI. The last part of the model is a classification layer, this is often a fully connected layer and its task is to map the found features to class probabilities. The output layer often utilizes a softmax activation function that normalizes the outputs to a probability distribution among the classes.

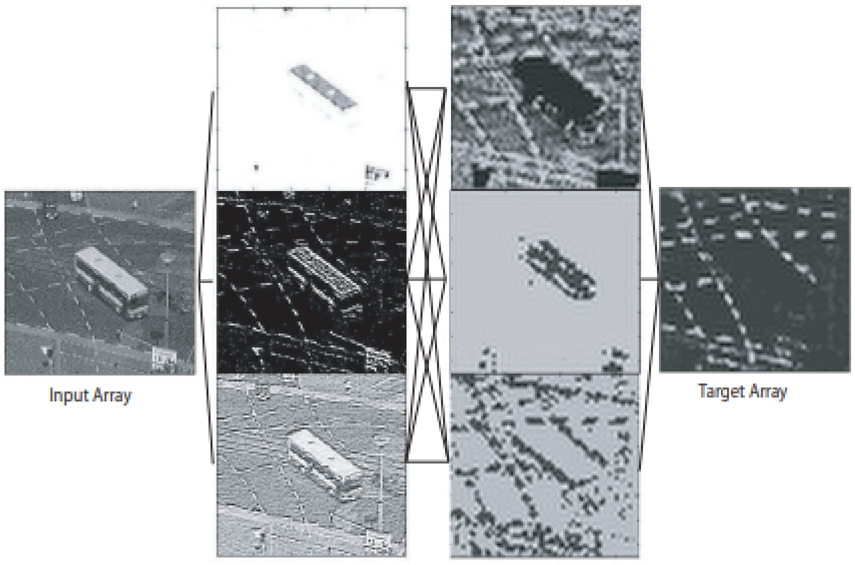


Figure 2.3: Input, feature, and output arrays of a convolution network applied to detecting road markers. (Credits : Browne & Ghidary[2]).

2.4.1 Two-stage detectors

When talking about state-of-the-art object detection models there are mainly two types of architectures being used. Firstly we have two-stage detectors like Faster R-CNN and Mask R-CNN that as the name suggests divide the problem into two parts [33][11]. The first part consists of a Regional Proposal Network (RPN) that identifies regions of interest and sends these down the pipeline. The second part of the model handles the actual object classification and bounding box regression. These models typically achieve higher accuracy results than one-stage detectors but are also typically slower [4][35].

2.4.2 One-stage detectors

The second architecture is the one-stage object detection model. One-stage models such as YOLO (You Only Look Once) and SSD (Single Shot MultiBox Detector) treat the problem as a single regression task instead, directly predicting bounding box coordinates and class probabilities in one pass through the network [32][22]. These models are typically significantly faster than two-stage detectors, with the trade-off being that they often are not as accurate.

2.4.3 YOLOv9

As speed is an essential part of real time object detection, the YOLO model will be the architecture that is utilized in this thesis. The first YOLO model was published by Redmon et al. (2015) and since then continuous improvements have been made [32]. The most recent

published model is the YOLOv9 that set new benchmarks on the MS COCO data set [40]. The model uses two new concepts called Programmable Gradient Information (PGI) and the Generalized Efficient Layer Aggregation Network (GELAN) and aims to tackle the problem of information loss in deep networks.
The information bottleneck principal illustrates the problem of information loss in deep networks. This phenomenon can be illustrated mathematically through equation 2.4, where I denotes mutual information, f and g are transformation functions with the parameters θ and ϕ.

$$I(X, X) >= I(X, f_\theta(X)) >= I(X, g_\phi(f_\theta(X))) \tag{2.4}$$

In a deep neural network g_ϕ and f_θ represent two consecutive layers. We can then see that the risk of information loss grows with the depth of the network. To handle this problem a new technique called Programmable Gradient Information (PGI) is utilized. PGI uses a component called Auxiliary Reversible Branch to help generate reliable gradients and update network parameters. The branch can achieve this because of its reversible architecture. If the functions g and f in equation 2.4 are reversible then no information loss occurs between layers. PGI also utilizes a Multi-level Auxiliary Information component that integrates networks between feature pyramid hierarchy layers. Essentially this component provides the main branch with information from different levels of the feature hierarchy as deep feature pyramids can risk losing important information otherwise. Another important component to tackle the information loss problem is the Generalized Efficient Layer Aggregation Network (GELAN). While Multi-Level Auxiliary Information component adds extra mechanisms to to provide the main branch with aggregated gradient information from different levels of the feature hierarchy, GELAN integrates different layers of the model to gather a complete picture of the data's details.

2.4.4 Transfer learning

Training a deep learning network can be extremely computationally demanding, especially when using large datasets. Luckily one does not always need to train a model from scratch when testing on new data. Instead a technique called transfer learning can be utilized where weights from a previously trained network can be re-used for a new task. The network is then fine-tuned on the new target dataset, often with the backbone weights of the network frozen. This allows the new model to utilize the feature extraction that has been learned by a previous model and the resources required for training is greatly decreased.

2.5 Evaluation metrics

In order to compare the performance of the models we need to define a couple of quantitative metrics that are commonly used when comparing object detection models. These metrics provide insights into the accuracy and robustness of the models in detecting the objects within images.

2.5.1 IoU

The first metric is the Intersect over Union (IoU) that measures how well the model can distinguish the located object from its background. It is defined as the overlap between the ground truth bounding box and the predicted bounding box and can mathematically be formulated as:

$$IoU = \frac{A \cap B}{A \cup B} \tag{2.5}$$

The IoU metric is always between 0 and 1 where a value closer to 1 is better as the model can more accurately distinguish the exact outline of the object. IoU helps set a threshold value where we say that we need at least an IoU value of, for example, 0.5 to say that the object is correctly classified. In Figure 2.4 we can see an example of an IoU value in the range of 0.5, this is often the lowest value to look at as a lower value means the predicted bounding box is quite far of.

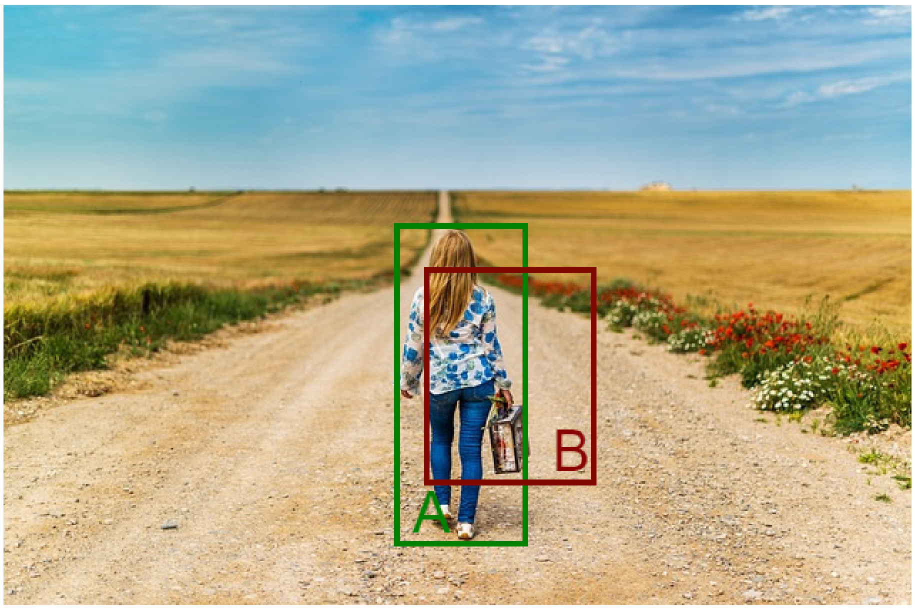


Figure 2.4: Example of an IoU value in the range of 0.5

2.5.2 Precision & Recall

Two other metrics that are important when evaluating the models are precision och recall. In order to define these metrics we first have to define the four cases when predicting objects:

- True Positive (TP) Correctly located object.
- False Positive (FP) Falsely located an object that is not there.
- True Negative (TN) Correctly does not locate anything in the backgound, typically not used in object detection.

- False Negative (FN) Failed to locate the object that is there.

Given these cases we can now define the precision and recall metrics for out situation when we are locating humans in images. Precision is the amount of positive predictions, located humans, that are correctly predicted. It is defined by the following formula:

$$Precision = \frac{TP}{TP + FP} \tag{2.6}$$

The recall metric is similar to the precision metric but instead gives us how many of the positives, humans in the image, were located. The recall metric is defined by the following formula:

$$Recall = \frac{TP}{TP + FN} \tag{2.7}$$

Recall and precision are complementary metrics that together can give a good estimation of the model performance.

2.5.3 Mean average precision

Average precision (AP) and Mean average precision (mAP) are very important metrics when evaluating the performance of an object detection model. The AP delves into the trade-off between precision and recall. It does this by calculating the area under curve (AUC) for the precision-recall curve. The precision-recall curve is simply the precision and recall values calculated at different threshold values. The curve illustrated the trade-off that one makes between precision and recall, the optimal value is not always clear but usually it is the point that gives the largest AUC. A precision-recall curve is illustrated in the Figure 2.5.
The Mean average precision (mAP) is an average of the AP values for many or all different classes and gives a comprehensive performance measure for the model. This value is often calculated at different IoU thresholds, illustrated by the $@x$ in mAP@x. This value can be set as in mAP@0.5 in Figure 2.5 or we can calculate the average value for many different IoU thresholds in a given range, mAP@0.5:0.95.

2.6 Synthetic training data generation

Synthetic data refers to artificially generated data that simulates real-world scenarios, providing an alternative to real data for training machine learning models. Various techniques are available for generating synthetic data, with game engines and generative models such as GANs being two popular options. Each method has its advantages and disadvantages, making them suitable for different situations.

Game engines, such as Unity[1] and Unreal[2], offer significant benefits, particularly the ability to include ground truth annotations during data generation. This capability is essential for supervised learning, as it provides precise labels for training models. Furthermore, game engines provide extensive control over the environment, enabling precise adjustments

[1] `https://unity.com`
[2] `https://www.unrealengine.com`

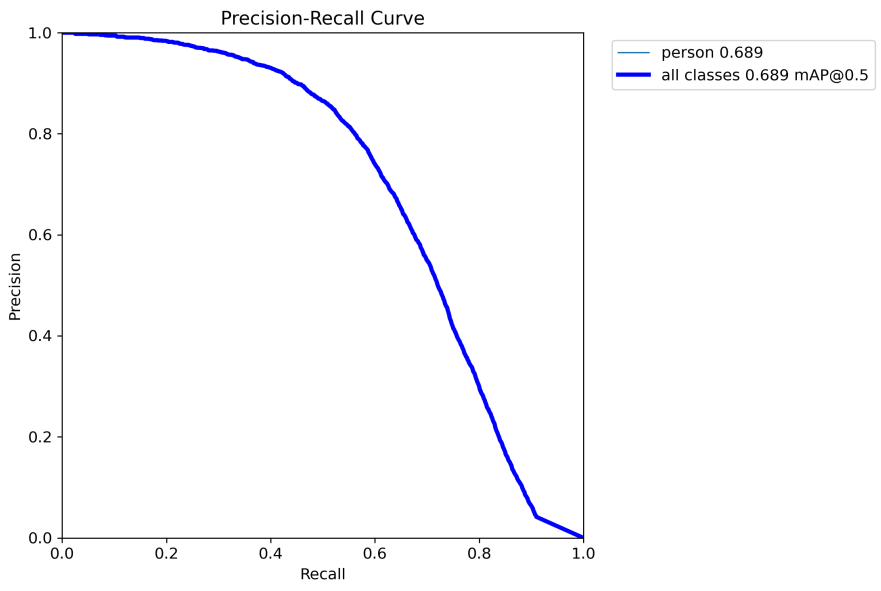


Figure 2.5: Example of a precision-recall curve for a one class object detection problem

in parameters like position, rotation, texture, and lighting. This flexibility facilitates domain randomization, enhancing the robustness and generalization capabilities of models trained on synthetic data. However, a notable disadvantage of game engines is the domain gap, which refers to the visual differences between synthetic and real-world data. Variations in appearance, texture, lighting conditions, and other environmental factors can challenge the generalization of models to real-world scenarios.

Generative Adversarial Networks (GANs), on the other hand, excel in generating highly realistic images that are often close to indistinguishable from real-world data. This realism can potentially reduce the domain gap and aid in better generalization to real-world data. Despite this advantage, GANs have significant drawbacks, including the lack of inherent ground truth annotations, which complicates the generation of labeled datasets. Additionally, training GANs is computationally intensive and requires substantial expertise to achieve optimal results.

Given these considerations, Unity was chosen for this thesis due to its practical advantages and comprehensive capabilities. Unity's extensive control over data generation and built-in support for annotations make it a superior choice for creating robust synthetic datasets. Additionally, considering that GANs have already been extensively studied and may not offer novel insights for our thesis, and due to their complexity and computational intensity, they are less suitable for our thesis needs. Therefore, Unity presents a more practical and efficient solution for synthetic data generation in this context. Furthermore, Unity Perception was chosen over other game engines because of its user-friendly interface, versatility, and integration with the broader Unity ecosystem, which enhances the accessibility and usability of the platform for our thesis purposes.

2.6.1 Unity Perception

The Unity Perception package is an extension of the Unity editor, specifically designed for generating synthetic datasets including ground truth annotations [39]. Its versatility and capabilities cater to a wide range of computer vision tasks, including 2D/3D object detection, semantic segmentation, and instance segmentation. One of its key strengths lies in its support for domain randomization, which allows for the creation of diverse and realistic synthetic datasets by manipulating various environmental factors.

When utilizing the Unity Perception package to create synthetic datasets, one has full control over the data generation process. Through the Unity editor interface, users can position a camera and manipulate objects within the scene with precision. These objects can be randomly generated in terms of position, rotation, texture, lighting, and other parameters, enabling the creation of highly diverse and customizable datasets.

An exemplary scene generated using the Unity Perception package can be seen in Figure 2.6, illustrating the placement of random objects in front of a canvas. This image is extracted from the PeopleSansPeople package, which not only generates humans in various poses but also incorporates random objects against an image background sourced from the COCO dataset [9].

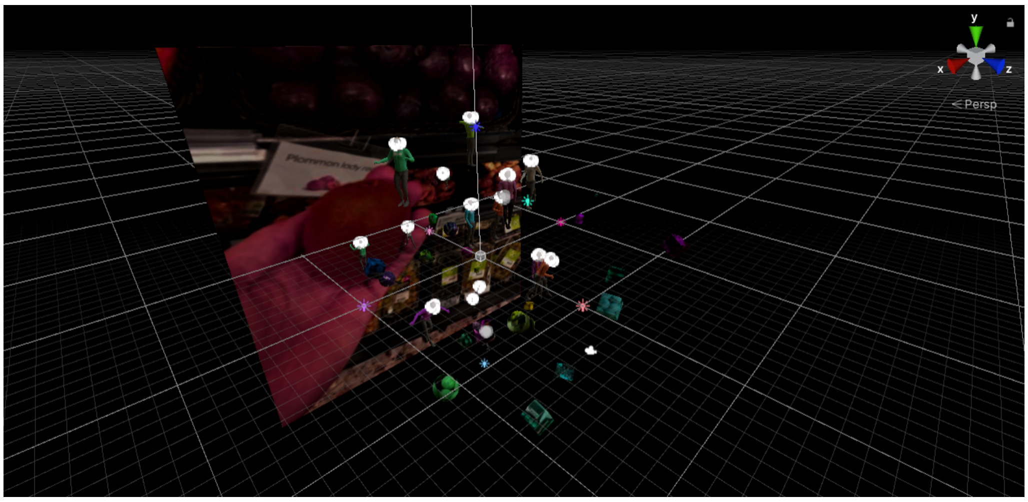


Figure 2.6: Example image taken from the editor of the PeopleSans-People Unity Perception package.

Unity Perception has been demonstrated to be useful in enhancing model performance, particularly when compared to training solely on real-world data. Studies by Borkman et al.[1] and Ebadi et al.[8] have provided empirical evidence of its efficacy in augmenting model training and improving model robustness.

Unity Perception stands out as a practical and efficient tool for generating synthetic data in computer vision. Its user-friendly interface, comprehensive control over data creation, and seamless integration with Unity make it a valuable asset. These features make it accessible and suitable for various research goals, including our focus on improving object detection models using synthetic data.

2.7 Domain adaptation

When developing deep learning models for computer vision tasks, synthetic data offers significant advantages. Yet when it comes to certain tasks, networks trained on synthetic data often experience a significant decline in performance when tested with real-world data. This discrepancy arises from the inherent difference in the data, also known as the domain gap. To address this challenge, several new techniques for adapting the synthetic data to the target domain have been developed. In this thesis, the primary focus will be on adapting the synthetic data to the target domains. Furthermore, various training strategies for the object detection models will be explored to examine their efficacy in bridging the domain gap.

2.7.1 The domain gap

The domain gap stems from differences in the distribution of the source and target images. This discrepancy can be attributed to any number of factors such as differences in lighting, pixel-level details, background, or object appearance. Even shifts that appear mild to a human observer can have a surprisingly large effect on the model performance [14]. Consequently, when models trained on synthetic data are applied to real-world scenarios, they often struggle to generalize due to the disparity in source and target domains.

The concept of the domain gap also encompasses the notions of in-distribution and out-of-distribution data. In-distribution data comprises samples that closely resemble the training data, exhibiting similar visual characteristics, contextual cues, and environmental conditions. These samples are representative of the scenarios encountered during the model training process, contributing to a well-defined training distribution.

Conversely, out-of-distribution data refers to samples that diverge significantly from the training distribution. These samples may exhibit unexpected variations in appearance, context, or environmental factors, which were not adequately represented during model training. As a result, models trained solely on synthetic data may struggle to generalize to out-of-distribution samples encountered in real-world scenarios. These unforeseen variations pose challenges to model robustness and performance, highlighting the importance of addressing the domain gap comprehensively. The distinctions between in-distribution and out-of-distribution data can provide a better understanding of the limitations of models trained on synthetic data and highlight the importance of strategies to mitigate their impact.

Chapter 3
Method

This chapter outlines the methodology and choices made in order to investigate how the use of synthetic data influences the performance of an object detection model. Model selection, data selection/splits and synthetic data generation will be discussed.

3.1 Object detection model & training strategies

The object detection model utilized to test our data is the YOLOv9-c model from Ultralytics [1]. The YOLOv9-c model, with 25.3 million parameters, is the second largest in the YOLOv9 family. This model was chosen due to its balance between performance and computational efficiency, which has been demonstrated in various object detection tasks [40]. The hyperparameters used in training and testing the YOLOv9-c model are primarily the default values provided by Ultralytics. The only parameter that is altered between experiments and datasets is the number of epochs. All hyperparameters, including the number of epochs, remain consistent within each experiment to ensure that the models being compared are evaluated under the same conditions. A complete list of hyperparameters used can be found in appendix A.

3.2 Data selection

In this section relevant data is presented, including how train-test splits are made and selections of classes for the models. The real data used in the thesis will come from three sources, the MS-COCO dataset[21], a cctv dataset from Roboflow[6] and Verisure's dataset. The MS-COCO dataset is utilized mainly as a training dataset for the pre-training of YOLOv9 models and the other datasets are used for fine-tuning and testing.

[1] https://docs.ultralytics.com/models/yolov9/

3.2.1 MS-COCO

The MS-COCO (Microsoft Common Objects in Context) dataset is a large public dataset intended for several different computer vision tasks. It contains more than 200,000 images that are annotated with object labels and it is widely used for benchmarking deep learning models. The pre-trained YOLOv9 model that is utilized in the experiments is trained using the MS-COCO dataset. Because of the computational resources required to train an object detection model with large datasets, limitations in the size of the datasets have been made when comparing models trained from scratch. The dataset that will be used to compare to our synthetic model trained from scratch consists of 10000 images from the COCO dataset that all contain at least one human in them. Figure 3.1 shows some sample images from the COCO dataset used.

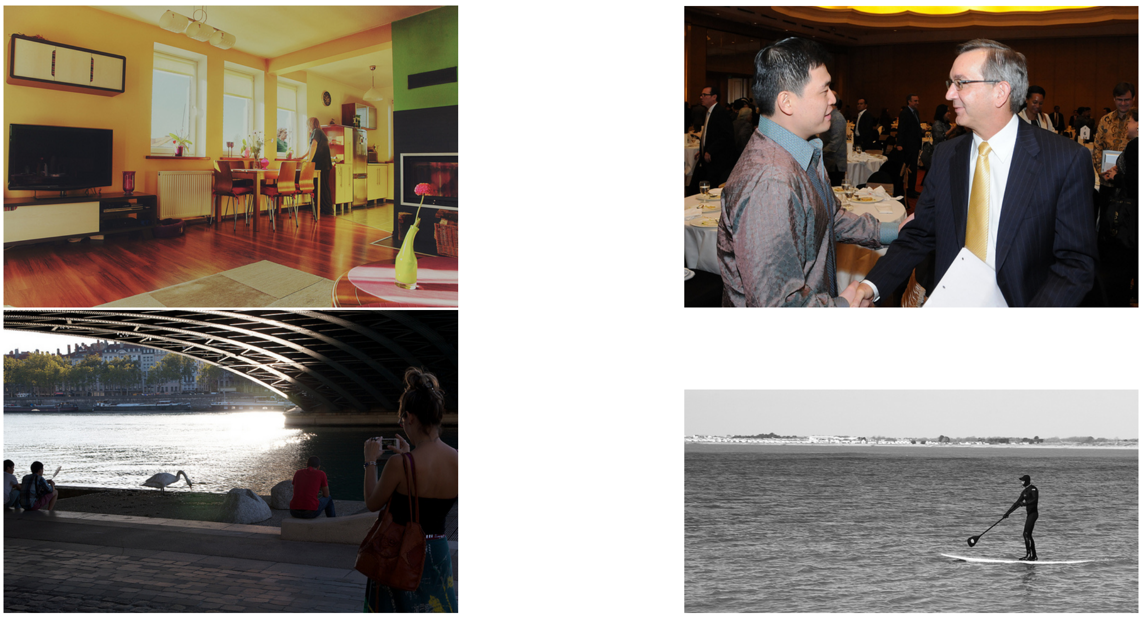


Figure 3.1: COCO dataset images

3.2.2 Synthetic datasets

We have generated and tested several synthetic datasets, incorporating mainly variations in backgrounds, human placement, and the number of humans per image. These variations are designed to evaluate the impact of different synthetic data characteristics on the performance of the object detection model. The datasets generated were all 10000 images large large, this limitation was chosen both in order to make a fair comparison to the real data and because of the computational requirements. Section 3.3 describes details of the data generation framework with example synthetic images.

3.2.3 Verisure datasets

Two different Verisure datasets were utilized in the experiments. Both datasets were collected by Verisure using testing units and staged environments for the purpose of data collection and testing. The first dataset is purely a test set consisting of 65 images of crawling people. This test set is used in experiment 4.3 and is composed of Verisure images taken in and around two different houses. The humans to detect are exclusively people crawling on all fours or on their stomach. Secondly a standing dataset that consists of 450 images, divided between two houses, with 250 images from one house and 200 from another. The images are taken from elevated positions both inside and outside the houses, ensuring a consistent visual perspective. Each image contains at least one human subject. The Verisure dataset, combined with the Roboflow CCTV dataset, provide a comprehensive foundation for evaluating the generalization capability of synthetic data to real-world scenarios. By testing the models on both datasets, we can effectively measure how well the synthetic data training translates to different real-world environments. Figure 3.2 illustrates example images from both of the Verisure datasets.

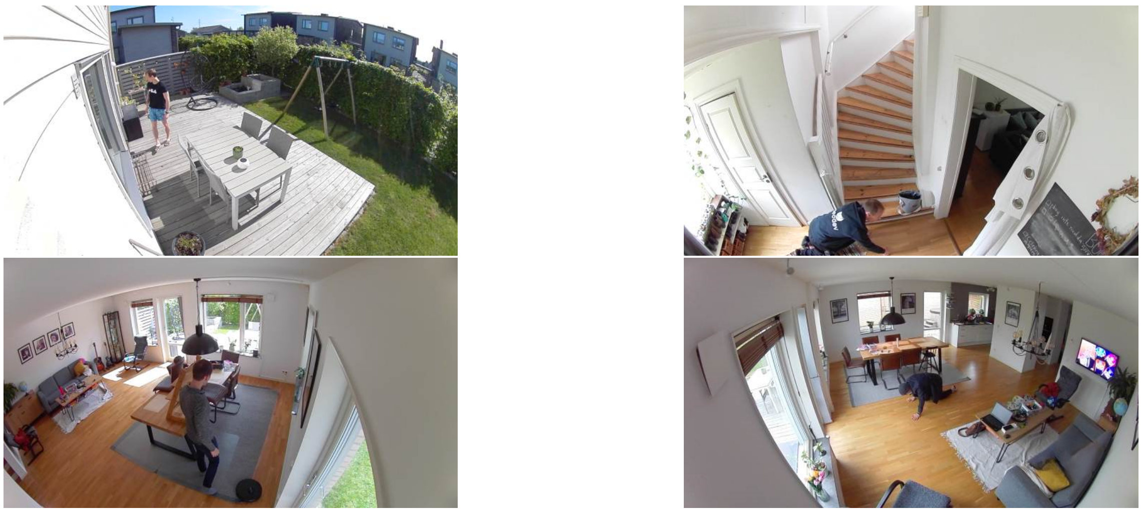


Figure 3.2: Verisure standing(left) and Verisure crawling(right) dataset images

3.2.4 Roboflow cctv dataset

This Roboflow dataset was selected due to its high variability in human placement, background, lighting, and other factors. Simultaneously the data has a consistent source – primarily security camera footage. The majority of images are captured from elevated positions, sharing a similar visual appearance to the Verisure dataset. This combination renders it a suitable test dataset for our objectives. Unlike larger datasets like MS-COCO, which exhibit a higher degree of randomization, our synthetic data aims to emulate scenarios more closely aligned with the characteristics of this dataset. The dataset consist of 700 images in total, Figure 3.3 illustrates random images taken from the dataset.

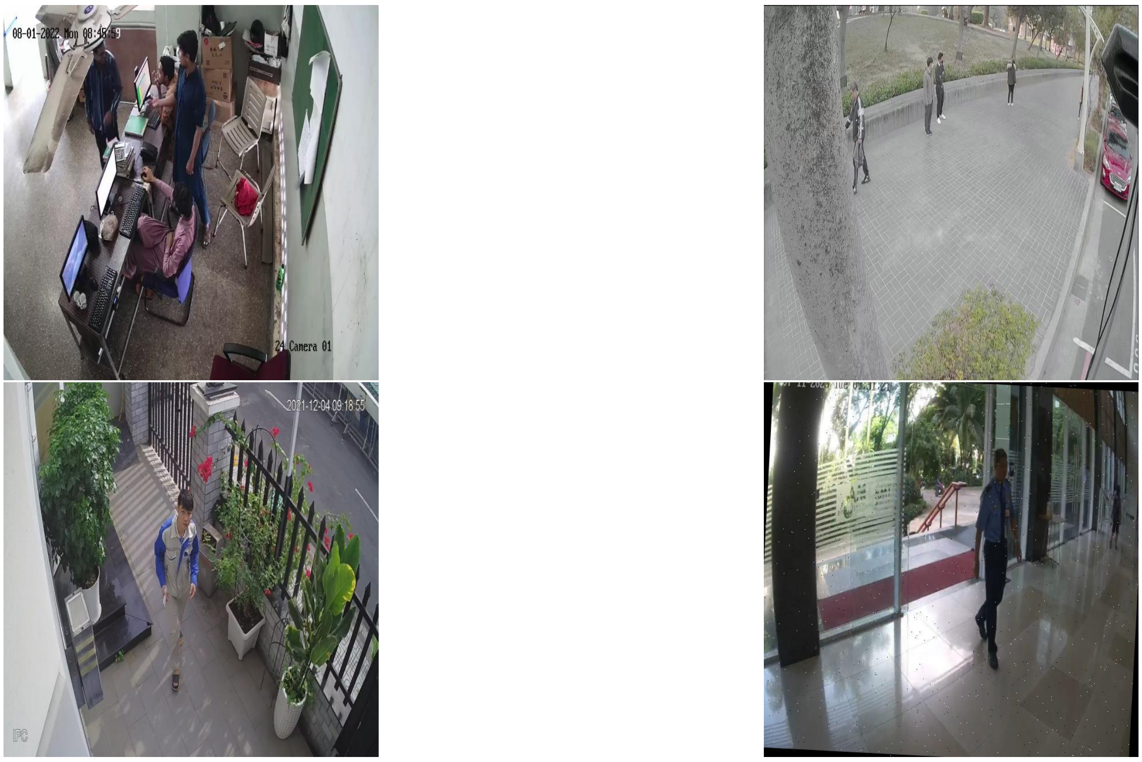


Figure 3.3: Roboflow dataset images

3.2.5 Data splits

It is standard practice when training machine learning models to split the data into three distinct subsets: the training set, the validation set, and the test set. The training set is used to train the model, the validation set is used to assess the trained model and fine-tune parameters to optimize performance, and the test set is the final set used for evaluation once model development is complete.

In this thesis, data splits were made in various ways due to the different use cases for the different datasets. For the COCO dataset and all synthetic datasets, a split of 9000 training images and 1000 validation images was used. No test set was employed as no fine-tuning of the model hyperparameters was conducted, making the validation set sufficient for evaluation.

The target CCTV and Verisure standing datasets were split using a 45/10/45 train/validation/test split, resulting in 315/70/315 images for the CCTV dataset and 200/45/200 images for the Verisure dataset. This split was chosen to ensure a sufficiently large test set, providing reliable test results, especially in scenarios where real data accessibility is limited. For the Verisure standing dataset, all images from one house were used for training and validation, while all images from the other house were used as the test set.

3.3 Synthetic data generation using Unity perception

The synthetic data is generated using the Unity Perception framework, enabling the creation of highly randomized datasets. This process simulates the diverse and complex nature of real-world data, capturing a wide range of variations in human appearance, poses, and environmental settings. A key component of our synthetic data generation process is the Synthetic Humans package in Unity [3]. This package allows for the creation of randomized human figures, each with unique characteristics such as age, height, gender, clothing, and skin color. The package also supports the randomization of human poses by sampling from a collection of animations. Custom animations can be integrated to further expand the range of possible postures and movements, enhancing the realism of the dataset. Figure 3.4 illustrates random humans generated using the Synthetic Humans package.

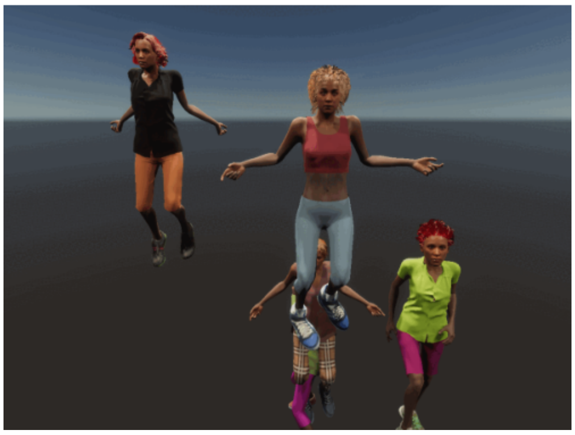


Figure 3.4: Synthetic Humans example

In addition to human Figures, the environmental context is crucial for creating realistic synthetic images. For this purpose we incorporated 730 different HDRI skyboxes from PolyHaven, which provide high-quality backgrounds for the scenes. These skyboxes include both indoor and outdoor environments which adds contextual variety to the dataset. By randomizing the selection of skyboxes for each image, the generated dataset encompasses a wide range of settings. This variability in backgrounds helps in training models that are robust to different environmental background conditions and therefore generalize better to unseen data. Example backgrounds are illustrated in Figure 3.5.

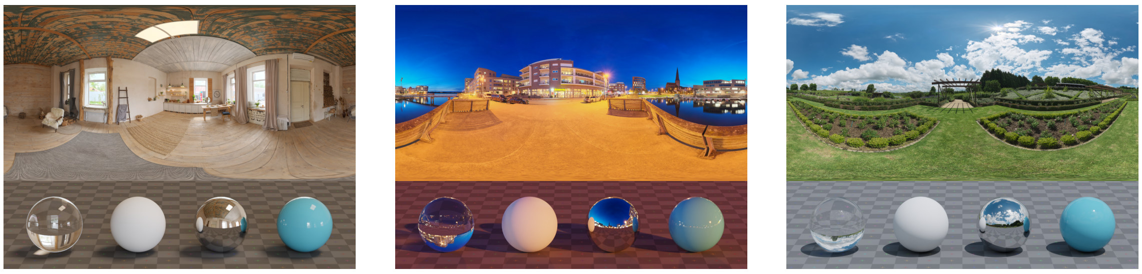


Figure 3.5: Example HDRI backgrounds taken from PolyHaven

To further enhance the robustness of our synthetic dataset, we implemented several strategies for additional randomization. We varied lighting conditions and sun angles in each image to ensure a broad spectrum of illumination scenarios. Furthermore, small objects with random textures were added to the scenes to introduce additional elements of complexity and variability. These objects are ensured to not be placed to close to the camera and are quite small as to ensure that they do not completely obscure any humans. This combination of varied lighting and additional randomized objects creates a more detailed and varied environment, ultimately improving the performance and reliability of the trained models. Figure 3.6 illustrates all components combined.

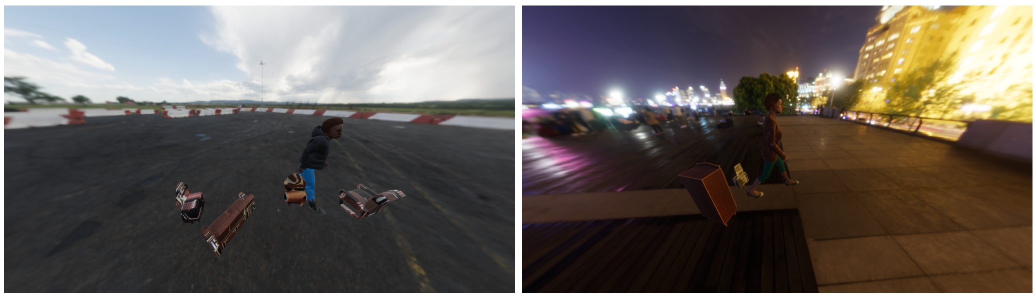


Figure 3.6: Examples of synthetic images generated

Chapter 4

Experiments & Results

In this chapter, the experiments conducted and the results obtained are presented to address the research questions posed in the thesis. The primary metric of interest is the mean Average Precision at an Intersection over Union threshold of 0.5 (mAP@0.5), with additional attention given to the mean Average Precision across a range of IoU thresholds from 0.5 to 0.95 (mAP@0.5-0.95).

4.1 Generating synthetic data

The primary objective of the synthetic data is to provide the object detection model with a diverse set of human subjects against varied backgrounds, enabling the model to effectively detect humans in real-world footage. Additionally, the goal is to make these images emulate properties commonly found in security surveillance footage to closely mimic the target domain, as training on images from a different domain will negatively impact in-domain testing performance [24].To determine the optimal composition of synthetic images for this purpose and to evaluate its performance relative to the target data, four distinct synthetic datasets, consisting of 10000 images each, were generated and tested.
The first dataset, denoted as Version 1, is designed to more closely emulate lifelike scenarios. Human subjects are strategically placed within the images at consistent depths, with minimal randomization in size and depth. This approach aims to mimic real-world scenes by placing a single human at what is most often a plausible position in the image, potentially facilitating better alignment with the target data and enhancing the model's adaptability to realistic environments. Only using a single human in each image also ensures that the model can fully capture the features of the humans which could allow the model to learn human features more accurately.

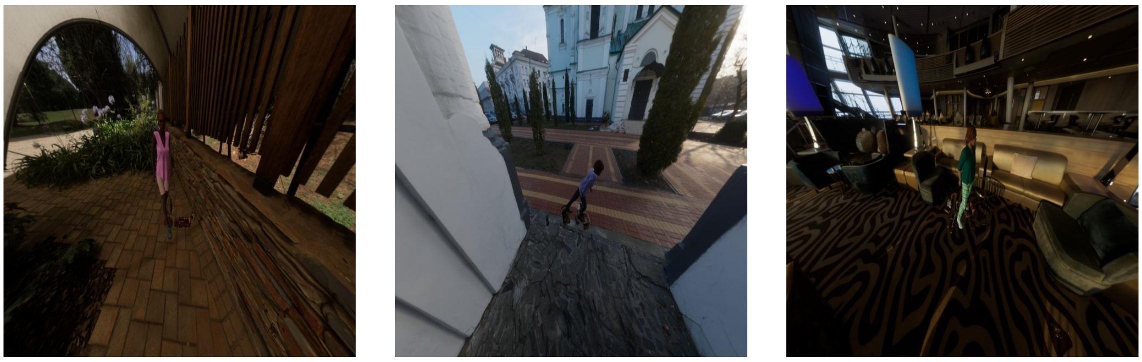


Figure 4.1: Version 1 example images

The second dataset, Version 2, introduces increased randomness in the positioning, depth and number of human subjects. This results in images that appear less realistic but feature a higher density of human subjects with some degree of overlap.

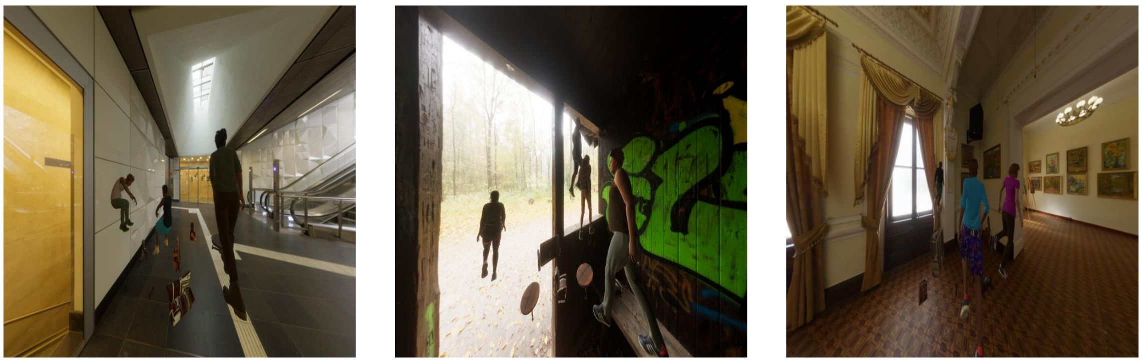


Figure 4.2: Version 2 example images

The third dataset, Version 3, introduces an even larger degree of randomization, incorporating up to 15 human subjects per image with a high degree of randomness in their placement and depth. This level of variation aims to give the object detection model humans in a large variation of scenarios and often partly blocked by other humans.

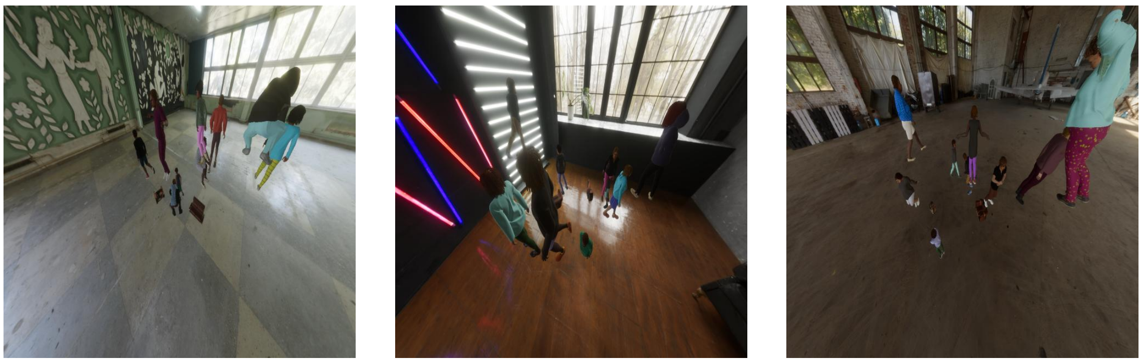


Figure 4.3: Version 3 example images

Finally, Version 4 has the same settings as Version 3 with the exception that animations of crawling humans have been added to the animation sampling pool.

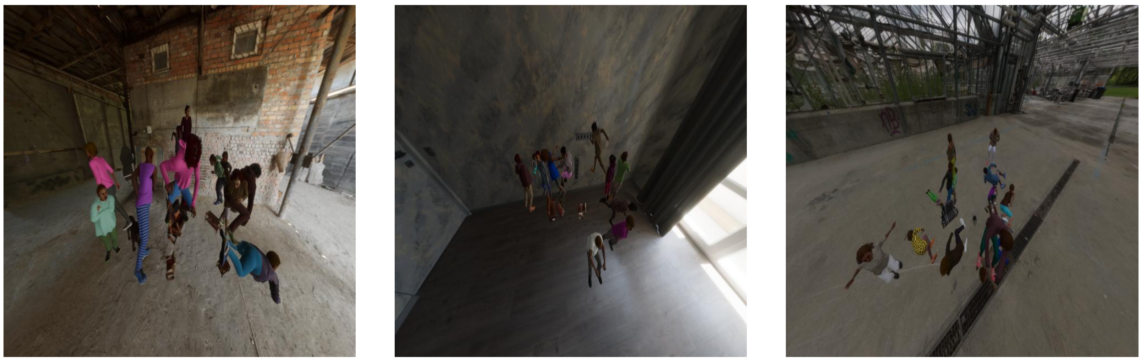
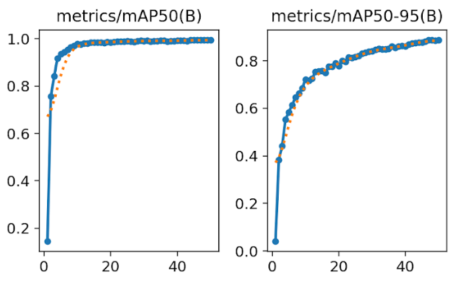
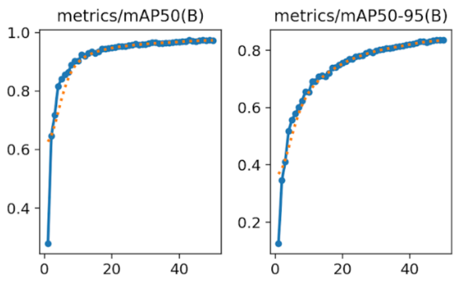


Figure 4.4: Version 4 example images

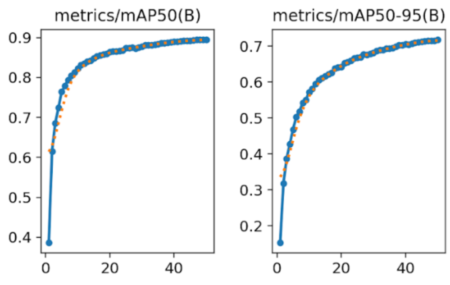
The datasets were used to train an YOLOv9-c model from scratch for 50 epochs and then evaluated on two distinct test sets. These test sets are the CCTV and Verisure Standing datasets. Results from the training processes on the validation can be seen in Figure 4.5 and the resulting test values can be seen in Figure 4.1.



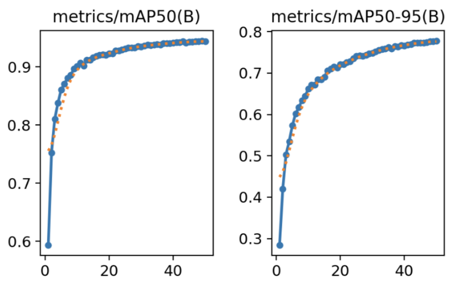
(a) V1



(b) V2



(c) V3



(d) V4

Figure 4.5: Validation results for training of YOLOv9-c models from scratch for 50 epochs

	CCTV test data		Verisure standing test data	
Synthetic data	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
Version 1	0.318	0.0959	0.35	0.16
Version 2	0.4	0.0922	0.522	0.241
Version 3	0.408	0.0989	0.598	**0.317**
Version 4	**0.437**	**0.107**	**0.629**	0.313

Table 4.1: Results synthetic data

The validation curves for mAP@0.5, shown in Figure 4.5, have all leveled out, indicating that 50 epochs were sufficient for the model to learn from the training data. As shown in Table 4.1, Version 4 has the best performance for 3 out of 4 metrics. A higher degree of randomization in the placement and poses of the humans improves the model's generalization. Notably, including crawling animations in the animation pool enhanced the model's overall performance on both standing test sets. This highlights the importance of exposing the model to a range of human poses, including partially occluded individuals.
These results will be used in the following experiments, with Version 4 being used to compare synthetic data to real data in 4.2.

4.2 Comparing synthetic and real data

To understand the effectiveness of synthetic and real data, we began with comparing models trained on both types of data without fine-tuning. We evaluated the top-performing synthetic data model against a model trained on an equivalent number of images from the COCO dataset, trained for the same number of epochs. This comparison is conducted on the CCTV and Verisure standing datasets, allowing us to compare their performance on the target datasets. Figure 4.2 presents the comparison results, these results illustrate how effectively the model learns human features across different scenarios without specific adaptation to the target domain.

	CCTV test data		Verisure test data	
Model	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
COCO	**0.474**	**0.238**	**0.871**	**0.475**
Synt-v4	0.362	0.196	0.629	0.313

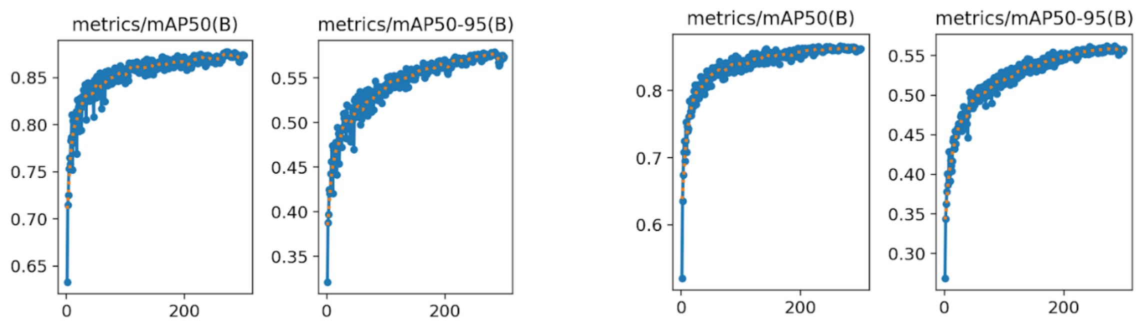
Table 4.2: Results on the target datasets

As illustrated in the Table above, synthetic data does not generalize to our test sets as well as a diverse real dataset like the COCO dataset. The performance drop-off when using only synthetic data compared to real data is substantial. There could be several reasons for this drop-off, but the simplest explanation is that the domain gap between the synthetic and test sets is larger than that between the COCO and test sets. Synthetic data fails to capture all the characteristics of real data or simply lacks sufficient variability.

Our next comparison involves assessing the same models after fine-tuning them on the target datasets. To achieve this, the models were first fine-tuned using the CCTV dataset for 300 epochs. The validation curves of the training are depicted in Figure 4.6. Additionally, we utilized the same CCTV train/val images to train a new model from scratch in order to benchmark how well a model can perform using only these images. The performance of the models on the CCTV test set is illustrated in table 4.3.

Model	mAP@0.5	mAP@0.5:0.95
COCO-FT	0.847	**0.536**
Syntv4-FT	**0.850**	0.532
CCTV-train	0.756	0.413

Table 4.3: Fine-tuned models comparison on CCTV dataset



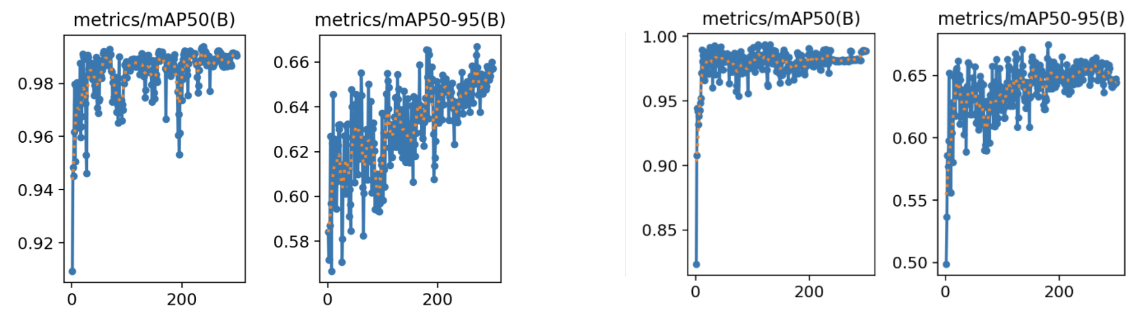
(a) COCO **(b)** Synt-v4

Figure 4.6: Validation results from fine-tuning of model using target dataset CCTV

The second target dataset to fine-tune our models on is the Verisure standing dataset. Similarly the two original models were fine-tuned for 300 epochs on this dataset, with the validation curves illustrated in Figure 4.7. A benchmark model trained from scratch using the Verisure standing data was also used. The results on the Verisure standing test set are illustrated in table 4.4.

Model	mAP@0.5	mAP@0.5:0.95
COCO-FT	**0.866**	**0.547**
Syntv4-FT	0.857	0.554
Verisure-train	0.521	0.231

Table 4.4: Validation results from fine-tuning of model using Verisure standing dataset



(a) COCO **(b)** Synt-v4

Figure 4.7: Validation results from fine-tuning of model using target dataset Verisure standing

When models trained on synthetic data were fine-tuned on the target datasets, notable performance improvements were observed. For both the CCTV and Verisure standing datasets, the models initially trained on synthetic data performed comparably to those pretrained on the COCO dataset. Both models outperformed models trained only on the target datasets, highlighting the benefits of pretraining models on a large and diverse dataset.
Notably, the validation curves for the CCTV and Verisure models differ somewhat during the training process when compared to the testing scores, likely due to the characteristics of the datasets. The CCTV test set shares the same distribution as the training and validation sets, while the Verisure test set comes from a slightly different distribution, with all test images originating from a different house than the training and validation images. The Verisure models appear to overfit the training set, which is likely why the model trained solely on the Verisure training data performs rather poorly as well (see Figure 4.4).

4.3 Improving Crawling Person Detection with Synthetic Data

The pre-trained YOLOv9-c model, which performs well on the Verisure standing dataset, shows a notable drop in accuracy when tested on the Verisure crawling dataset, as illustrated in Table 4.5. This is likely due to a lack of scenarios of crawling humans in the training data. Verisure emphasizes the challenge of finding reliable training data for certain scenarios, noting that if not properly addressed, this can impact the general performance of an object detection model. The scarcity of reliable training data can pose a significant challenge for improving model performance in detecting crawling individuals. In this section, we aim to enhance the detection of crawling individuals using synthetic data. We hypothesize that synthetic data could potentially bridge this gap and improve detection capabilities.

Test set	Precision	Recall	mAP@0.5	mAP@0.5:0.95
Verisure standing	0.972	0.87	0.956	0.567
Verisure crawl	0.753	0.548	0.675	0.41

Table 4.5: Pretrained YOLOv9-c model tested on Verisure standing and crawling data

Four different synthetic datasets were generated with the intent of enhancing the model in these scenarios. The first set consists of 100% crawling humans(crawl100), the second set 75% crawling(crawl75) and 25% standing, the third set 50% crawling(crawl50) and 50% standing, and the fourth set 25% crawling(crawl25) and 75% standing. Example images from all sets can be seen in Figure 4.8. The synthetic datasets were generated using mostly the same settings as version 4 from section 4.1, with the exception being the proportion of crawling/standing humans generated in each set.

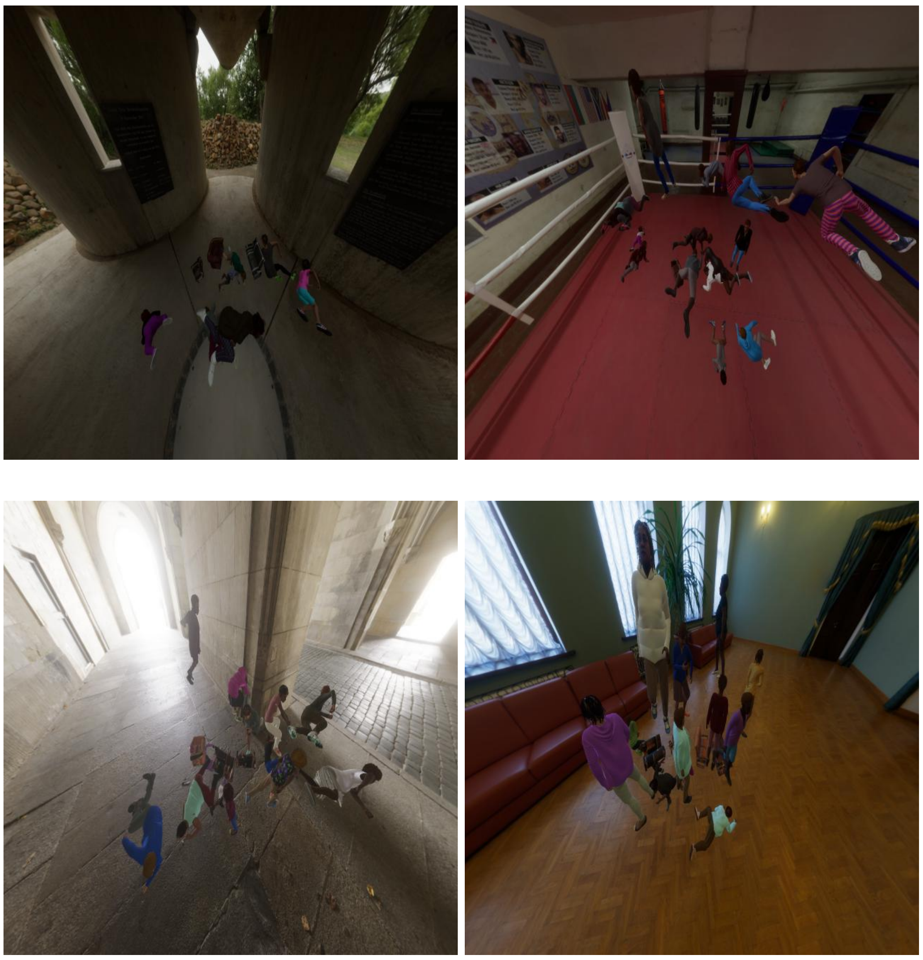


Figure 4.8: Examples from the synthetic crawling datasets: crawl100(top left), crawl75(top right), crawl50(bottom left) & crawl25(bottom right)

The testing data for these experiments are the two Verisure datasets: crawling and standing. The standing dataset is utilized in combination with the crawling dataset to examine any potential performance loss on this set as the model is trained on generated crawling humans.

We will evaluate the PT YOLOv9-c model under four different experiment-sets to understand the impact of synthetic data on crawling person detection:

1. **Baseline Performance**: Testing the PT YOLOv9-c model in its original state on the Verisure crawling test set.
2. **Fine-Tuning with Verisure standing Data**: Fine-tuning the model using the Verisure standing dataset.
3. **Fine-Tuning with Synthetic Data**: Fine-tuning the model using synthetic data designed to include crawling scenarios.
4. **Fine-Tuning with Synthetic Data and Verisure standing Data**: Fine-tuning the model first with synthetic data and then with Verisure standing data.

The performance comparison aims to demonstrate the potential benefits of incorporating synthetic data for improving detection capabilities. The results of experiments using experiment-set 3 are summarized in Table 4.6 and experiment-set 4 in Table 4.7.

	Crawling test data		Standing test data	
Model	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
PT	0.675	**0.41**	**0.956**	**0.567**
PT-ft Verisure standing	0.372	0.126	0.719	0.387
PT-ft crawl100	**0.684**	0.401	0.647	0.337
PT-ft crawl75	0.633	0.389	0.647	0.357
PT-ft crawl50	0.618	0.374	0.689	0.375
PT-ft crawl25	0.520	0.301	0.673	0.376

Table 4.6: Results from experiment-set 3 including benchmark experiment-set 1(PT) & 2(PT-ft Verisure)

	Crawling test data		Standing test data	
Model	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
PT	0.675	**0.41**	**0.956**	**0.567**
PT-ft Verisure standing	0.372	0.126	0.719	0.387
PT-ft crawl100 & Verisure standing	0.635	0.375	0.833	0.515
PT-ft crawl75 & Verisure standing	**0.707**	0.401	0.878	0.548
PT-ft crawl50 & Verisure standing	0.622	0.38	0.852	0.546
PT-ft crawl25 & Verisure standing	0.556	0.298	0.844	0.528

Table 4.7: Results from condition 4 including benchmark conditions 1(PT) & 2(PT-ft Verisure)

From table 4.7 we can see that the best performance on the crawling dataset was achieved using a synthetic dataset consisting of 75% crawling and 25% standing humans, which was then fine-tuned on the Verisure standing dataset. This model outperformed the pretrained YOLOv9-c model on the crawling test set which illustrates the applicability of synthetic data. This rather smal performance increase for the mAP@0.5 did however also result in a drop-off in performance when it came to the standing test set. This drop-off was expected as the model is retrained using synthetic data and adjust the weights so that it performs better on a set from a different domain. If the ultimate goal of the model is to perform well on both test sets, we would argue that the pretrained model still outperforms all other models.

4.4 Freezing the backbone

Continuing from our previous experiment, where we retrained the model with synthetic and Verisure datasets, this section explores freezing the backbone of the pretrained model. This approach, highlighted by Hinterstoisser et al.[13] can be an effective method for leveraging pretrained features while focusing the training on the task-specific layers. This technique helps maintain the robust feature extraction capabilities of the pretrained network, potentially improving detection accuracy for our specific application.

In Section 4.3, we utilized the pretrained YOLOv9-c model and fine-tuned it. In this experiment, we employed the best-performing synthetic dataset, crawl75, while keeping the backbone of the YOLOv9-c model frozen throughout the training. We applied the same procedure to the benchmark model, which involved fine-tuning the YOLOv9-c model on the Verisure target dataset. The results of these experiments are presented in Figure 4.8.

	Crawling test data		Standing test data	
Model	mAP@0.5	mAP@0.5:0.95	mAP@0.5	mAP@0.5:0.95
PT	0.675	0.41	**0.956**	0.567
PT-ft Verisure standing	0.704	0.341	0.883	0.543
PT-ft crawl75	0.629	0.368	0.803	0.453
PT-ft crawl75 & Verisure standing	**0.794**	**0.493**	0.919	**0.596**

Table 4.8: Pretrained YOLOv9-c model fine-tuned with frozen backbone, tested on normal and crawling data

The results presented in Table 4.8 demonstrate a notable performance increase when this approach is applied. The model fine-tuned on the crawl75 and Verisure standing datasets now outperforms the pretrained model by a notable amount in crawling detection. Importantly, this improvement is achieved with a much smaller decrease in standing performance compared to the PT model, indicating that freezing the backbone helps maintain a balance between detecting both standing and crawling individuals.

Chapter 5

Discussion

This discussion chapter synthesizes the key findings from our experiments to analyze the strengths and weaknesses of synthetic data in enhancing person detection capabilities. We critically assess the methodologies employed, acknowledging their strengths and limitations. Additionally, we explore potential avenues for future research aimed at optimizing the use of synthetic data to improve model robustness and generalizability across diverse surveillance environments. By proposing these directions, we contribute to ongoing efforts in leveraging synthetic data for training object detection models.

5.1 Interpretations of results

In section 4.1 and 4.3 different synthetic datasets are generated and tested. The results from the different models in both of these sections are highly varied depending on synthetic dataset used. This suggests that the quality and characteristics of the synthetic data plays a crucial role in training outcomes. Small changes in placements, poses, or the number of humans generated are shown to affect the results of the models significantly.
The results from section 4.2 show that models trained on synthetic data, when fine-tuned on the target datasets, performed comparably to those pre-trained on the COCO dataset and outperformed models trained solely on target datasets. This indicates that synthetic data can be an effective alternative to real data for initial model training. However, as indicated in Table 4.2, the model trained exclusively on synthetic data does not generalize to real-world test sets as effectively as a model trained on real data. These results highlight the domain gap between synthetic and real data and demonstrate that fine-tuning with even a small portion of the target dataset shows potential to effectively bridge this gap.
Finally, in section 4.3 and 4.4 the results showcase the potential use of synthetic data to enhance the performance of a model in an out-of-distribution scenario. The results from section 4.4 illustrates that with a good synthetic dataset and good choice of training parameters, the performance on out-of-distribution scenarios can be significantly enhanced without a large

impact on in-distribution scenario performance.

5.2 Methodology

The methodology employed in this study ensures that the differences in model performance are solely attributable to the data used. By using the YOLOv9-c model with consistent hyperparameters across all experiments, we eliminate variability due to training configurations. However, this approach does not account for the possibility that different datasets might have different optimal hyperparameters, which could yield better results if adjusted individually.

As stated in section 5.1, the synthetic datasets generated using the Unity perception framework showed significant variability in model performance. The potential exists for an optimal synthetic dataset to substantially improve model performance, yet identifying and creating this ideal dataset remains challenging.

A notable advantage of synthetic data, which this study did not fully exploit, is the ability to generate virtually unlimited amounts of data. This contrasts with the constraints of real data collection, where dataset size is often limited. This ability could greatly enhance model robustness and generalizability if leveraged properly.

5.3 Future research

In this thesis, we have examined the use of synthetic data and found it to be close to on par with real data only if we have a target dataset to fine-tune the model on. Closing this domain gap is currently a hot topic, and the benefits of achieving this could enable the use of synthetic data completely instead of real data. This would have significant implications for fields that rely heavily on large datasets for training, as synthetic data can be generated in a controlled and scalable manner, unlike real-world data which can be limited and costly to obtain.

Future research should particularly focus on transforming synthetic humans to look more like real humans, potentially using GANs (Generative Adversarial Networks) to transform the images into the real domain. GANs have shown great promise in generating highly realistic images, and their application in enhancing the realism of synthetic data could help bridge the gap between synthetic and real data domains. This transformation process could involve refining textures, pixel consistency, and ensuring the anatomical accuracy of synthetic humans.

Moreover, further research into training techniques is essential to fully unlock the potential of synthetic data. The effects of freezing the backbone in my experiments were significant, indicating that the training technique used plays a crucial role in the results. Investigating other advanced training methodologies, could provide additional insights into optimizing the use of synthetic data.

Chapter 6
Conclusions

This thesis set out to investigate the efficacy of synthetic data in enhancing person detection in surveillance scenarios, particularly where real-world data is scarce or difficult to acquire. By comparing models trained on synthetic data against those trained on real-world data from the COCO dataset, the study aimed to evaluate their effectiveness across different surveillance datasets, such as CCTV and Verisure's standing and crawling datasets.

We found that a higher degree of randomization produced favorable results compared to focusing on making the image look more realistic for our purpose. The results showed that our generated synthetic data did not perform on par with real data without fine-tuning the model to the target domain. However, when fine-tuning the model with a very small dataset from the target domain, we found that the synthetic data performed equivalently to the real data. These results show that synthetic data could potentially be used to enhance the performance of object detection models, as synthetic data is much easier to generate in large quantities.

We also found that synthetic data shows a lot of promise in specific situations where access to real data is very sparse. Our best model trained to detect crawling people only exhibited a small drop-off when detecting standing people, illustrating the potential of utilizing synthetic data.

Finally, the results of this study are limited to the test sets used, which impacts the generalizability of the findings. Broader testing across diverse scenarios would be necessary to draw more comprehensive conclusions about the effectiveness of synthetic data in object detection tasks.

Appendices

Appendix A
Hyperparameters

Hyperparameters used for training of models using datasets of size 9000 training images:

- learning_rate: 0.002
- Optimizer: AdamW
- batch_size: 16
- momentum: 0.9
- weight_decay: 0.0005
- epochs: 50
- image size: 640x640

Hyperparameters used for fine-tuning of models to target datasets:

- learning_rate: 0.002
- Optimizer: AdamW
- batch_size: 16
- momentum: 0.9
- weight_decay: 0.0005
- epochs: 300
- image size: 640x640

References

[1] S. Borkman, A. Crespi, S. Dhakad, Sujoy Ganguly, Jonathan Hogins, Y. Jhang, Mohsen Kamalzadeh, Bowen Li, Steven Leal, Pete Parisi, Cesar Romero, Wesley Smith, Alex Thaman, Samuel Warren, and Nupur Yadav. Unity perception: Generate synthetic data for computer vision. *ArXiv*, abs/2107.04259, 2021.

[2] Matthew Browne and Saeed Shiry Ghidary. Convolutional neural networks for image processing: An application in robot vision. In *Australian Conference on Artificial Intelligence*, 2003.

[3] Jon Hogins Cameron Sun, Salehe Erfanian Ebadi. Synthetic humans package in unity. `https://github.com/Unity-Technologies/com.unity.cv.synthetichumans`, 2022. Accessed: 2024-04-15.

[4] Manuel Carranza-García, Jesús Torres-Mateo, Pedro Lara-Benítez, and Jorge García-Gutiérrez. On the performance of one-stage and two-stage object detectors in autonomous vehicles using camera data. *Remote Sensing*, 13(1), 2021.

[5] Alexandru Damian, Claudiu Filip, A. Nistor, Irina Petrariu, Cătălin Mariuc, and Valentin Stratan. Experimental results on synthetic data generation in unreal engine 5 for real-world object detection. *2023 17th International Conference on Engineering of Modern Electric Systems (EMES)*, pages 1–4, 2023.

[6] dataset. cctv dataset. `https://universe.roboflow.com/dataset-uutxr/cctv-naxyo`, dec 2023. visited on 2024-04-18.

[7] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. *arXiv e-prints*, page arXiv:1603.07285, March 2016.

[8] Salehe Erfanian Ebadi, Saurav Dhakad, Sanjay Vishwakarma, Chunpu Wang, You-Cyuan Jhang, Maciek Chociej, Adam Crespi, Alex Thaman, and Sujoy Ganguly. Psp-hdri+: A synthetic dataset generator for pre-training of human-centric computer vision models, 2022.

[9] Salehe Erfanian Ebadi, You-Cyuan Jhang, Alex Zook, Saurav Dhakad, Adam Crespi, Pete Parisi, Steve Borkman, Jonathan Hogins, and Sujoy Ganguly. Peoplesanspeople: A synthetic data generator for human-centric computer vision. 2021.

[10] I. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron C. Courville, and Yoshua Bengio. Generative adversarial networks. *Communications of the ACM*, 63:139 – 144, 2014.

[11] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. *CoRR*, abs/1703.06870, 2017.

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. *CoRR*, abs/1512.03385, 2015.

[13] Stefan Hinterstoisser, Vincent Lepetit, Paul Wohlhart, and Kurt Konolige. On pre-trained image features and synthetic images for deep learning. *CoRR*, abs/1710.10710, 2017.

[14] Judy Hoffman, Dequan Wang, Fisher Yu, and Trevor Darrell. Fcns in the wild: Pixel-level adversarial and constraint-based adaptation. *CoRR*, abs/1612.02649, 2016.

[15] J. Huang, V. Rathod, C. Sun, M. Zhu, A. Korattikara, A. Fathi, I. Fischer, Z. Wojna, Y. Song, S. Guadarrama, and K. Murphy. Speed/accuracy trade-offs for modern convolutional object detectors. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3296–3297. IEEE, 2017.

[16] Håkon Hukkelås, Morten Smebye, Rudolf Mester, and Frank Lindseth. Realistic full-body anonymization with surface-guided gans. *CoRR*, abs/2201.02193, 2022.

[17] Sergey Ioffe and Christian Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, page 448–456. JMLR.org, 2015.

[18] J. Jabez, Maria Anu, and Dr Jabez. The power of deep learning models: Applications. *International Journal of Recent Technology and Engineering*, 8, 11 2019.

[19] L. Jiao et al. A survey of deep learning-based object detection. *IEEE Access*, 7:128837–128868, 2019.

[20] Heejae Lee, Jongmoo Jeon, Doyeop Lee, Chansik Park, Jinwoo Kim, and Dongmin Lee. Game engine-driven synthetic data generation for computer vision-based safety monitoring of construction workers. *Automation in Construction*, 155:105060, 2023.

[21] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, Lubomir D. Bourdev, Ross B. Girshick, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312, 2014.

[22] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. *SSD: Single Shot MultiBox Detector. Lecture Notes in Computer Science*, page 21–37. Springer International Publishing, 2016.

[23] Mohamed Loey, Gunasekaran Manogaran, Mohamed Hamed N. Taha, and Nour Eldeen M. Khalifa. Fighting against covid-19: A novel deep learning model based on yolo-v2 with resnet-50 for medical face mask detection. *Sustainable Cities and Society*, 65:102600, 2021.

[24] Spandan Madan, Timothy Henry, Jamell Dozier, Helen Ho, Nishchal Bhandari, Tomotake Sasaki, Frédo Durand, Hanspeter Pfister, and Xavier Boix. On the capability of neural networks to generalize to unseen category-pose combinations. *CoRR*, abs/2007.08032, 2020.

[25] Maximilian Menke, Thomas Wenzel, and Andreas Schwung. Improving gan-based domain adaptation for object detection. In *2022 IEEE 25th International Conference on Intelligent Transportation Systems (ITSC)*, pages 3880–3885, 2022.

[26] Jouveer Naidoo, Nicholas Bates, Trevor Gee, and Mahla Nejati. Pallet detection from synthetic data using game engines. *ArXiv*, abs/2304.03602, 2023.

[27] Villanustre Flavio Khoshgoftaar Taghi M Seliya Naeem Wald Randall Muharemagic Edin Najafabadi, Maryam M. Deep learning applications and challenges in big data analytic. *Journal of Big Data*, abs/2196-1115, 2015.

[28] Marcel Neuhausen, Patrick Herbers, and Markus König. *Synthetic Data for Evaluating the Visual Tracking of Construction Workers*, pages 354–361. Construction Research Congress 2020.

[29] Sergey I. Nikolenko. *Synthetic-to-Real Domain Adaptation and Refinement*, pages 235–268. Springer International Publishing, Synthetic Data for Deep Learning, 2021.

[30] Daniel Pototzky, Azhar Sultan, and Lars Schmidt-Thieme. Parting with illusions about synthetic data. In *2022 IEEE 14th Image, Video, and Multidimensional Signal Processing Workshop (IVMSP)*, pages 1–4, 2022.

[31] Ingeborg Rasmussen, Sigurd Kvalsvik, Per-Arne Andersen, T. N. Aune, and Daniel Hagen. Development of a novel object detection system based on synthetic data generated from unreal game engine. *Applied Sciences*, 2022.

[32] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 779–788, Los Alamitos, CA, USA, jun 2016. IEEE Computer Society.

[33] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015.

[34] S. Shen, Z. Liu, B. Zhao, L. Chen, and C. Zhang. Improving real-world object detection using balanced loss. In *2020 IEEE International Symposium on Broadband Multimedia Systems and Broadcasting (BMSB)*, pages 1–5. IEEE, 2020.

[35] Petru Soviany and Radu Tudor Ionescu. Optimizing the trade-off between single-stage and two-stage object detectors using image difficulty prediction. *CoRR*, abs/1803.08707, 2018.

[36] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott E. Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. *CoRR*, abs/1409.4842, 2014.

[37] N. Tilkin. Is the use of synthetic datasets a solution to improve object detection models on real data? Master's thesis, Faculté des Sciences appliquées, 2023.

[38] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. Training deep networks with synthetic data: Bridging the reality gap by domain randomization. *CoRR*, abs/1804.06516, 2018.

[39] Unity Technologies. Unity Perception package. `https://github.com/Unity-Technologies/com.unity.perception`, 2020.

[40] Chien-Yao Wang, I-Hau Yeh, and Hongpeng Liao. Yolov9: Learning what you want to learn using programmable gradient information. *ArXiv*, abs/2402.13616, 2024.

[41] Hui Zhang, Yonglin Tian, Kunfeng Wang, Haibo He, and Fei-Yue Wang. Synthetic-to-real domain adaptation for object instance segmentation. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2024-08-21

EXAMENSARBETE Evaluating synthetic data for enhancement of object detection models
STUDENT Carl Wikström
HANDLEDARE Volker Krueger (LTH), Patric Fröjd (Verisure Innovation)
EXAMINATOR Jacek Malec (LTH)

Exploring the Power of Synthetic Data in Detecting Humans

POPULÄRVETENSKAPLIG SAMMANFATTNING **Carl Wikström**

Synthetic data is being explored as a powerful tool to enhance object detection models, offering a promising alternative to traditional real-world datasets. But how powerful is synthetic data when training an object detection model for more specific tasks?

To delve deeper into the capabilities of synthetic data in object detection, we decided to examine its impact on the YOLOv9c model, a widely-used tool known for its efficiency in detecting objects within images. We began by training the model on synthetic data, which allowed us to generate extensive and diverse training scenarios not readily available in real-world datasets. Following this, we fine-tuned the model on specific real-world datasets to evaluate how well it could adapt and perform in practical situations. This two-step process aimed to explore the potential of synthetic data to improve detection accuracy and robustness in real-world applications.

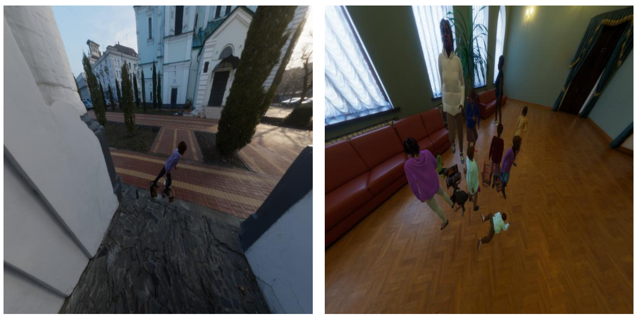


Figure 1: Synthetic data images

Our study specifically targeted human detection in security surveillance contexts, a domain where real-world data can be both limited and challenging to acquire. By introducing variations in the synthetic data—such as changes in human placements, poses, and quantities—we sought to determine how these factors influence model performance. This approach provided insights into how well synthetic data can prepare a model for the complexities of real-world scenarios and enhance its overall effectiveness.

In our evaluation, we compared the performance of models trained solely on synthetic data with those trained exclusively on real data. The goal was to understand the benefits of using synthetic data as a preliminary training resource and its impact on model accuracy when subsequently fine-tuned with real-world data.

Our results revealed that the performance of synthetic data varies significantly depending on its composition, highlighting the importance of selecting the right data mix. Notably, synthetic data can match real data performance when the model is fine-tuned with a small portion of the target dataset. Additionally, synthetic data proved valuable in improving detection capabilities for previously unseen scenarios, such as crawling people, demonstrating its ability to enhance model robustness in diverse situations.