Targeted Improvement of a Deep Learning Object Detector Using Synthetic Training Data

Katja Palmkvist
ka1347pa-s@student.lu.se

Olivia Mattsson
ol3270ma-s@student.lu.se

Supervisor, Axis
Marcus Thelander Andrén
marcus.thelander.andren@axis.com

Co-supervisor, Axis
Håkan Ardö
hakan.x.ardo@axis.com

Supervisor, LTH
Andreas Jakobsson
andreas.jakobsson@matstat.lu.se

June 8, 2022

LUND UNIVERSITY

Lunds Tekniska Högskola
Department of Mathematical Sciences

Abstract

When working with object detection, the quality and quantity of the training data is often a recurrent bottleneck. This thesis proposes a technique of incrementally improving an object detector using synthetically rendered data. The current training data within the field of focus is limited, and creating new data comes with significant integrity and security risks. By creating synthetic images where rendered people are placed in different adapted environments, the possibility of performing a targeted improvement of an object detector was examined.

First, the detection errors of an object detection model, trained on real data, were explored to identify weaknesses to target. Next, one of the found weaknesses was used to determine the content of the synthetic data. Furthermore, the synthetic data was generated, resulting in multiple datasets. Domain randomization techniques were used to optimize the data quality, and 3D graphics tools were used to generate bounding boxes. New models were then trained on a mix of real and synthetic data.

Finally, the best synthetic model was extracted, and another detailed evaluation was performed. The results showed that the new model, trained on additional synthetic data, did not suffer any unwanted side effects in its predictions. In addition, the ratio between the real and synthetic training data was examined, showing that using twice the amount of real data compared to synthetic provided the best results. The performance of the synthetic model improved compared to the initial model, trained solely on real data, for three different test datasets. The detailed evaluation showed that the number of detection errors for the targeted weakness decreased significantly. In conclusion, the results showed great promise in using synthetically rendered image data combined with real data to improve an object detection model.

Keywords: Machine Learning, Deep Learning, Object Detection, Computer Vision, Synthetic Data, 3D-rendering

Contents

Acknowledgements

First, we would sincerely like to thank Marcus and Håkan, our highly competent supervisors at Axis. The support and discussions have been invaluable, and we could not imagine better support when working on our thesis. We are forever grateful.

Additionally, we would like to thank our other colleagues at Axis for making the time spent at Axis a blast. Well-deserved breaks containing both pickleball and ping-pong helped us stay focused. Thank you for making us feel like we were a part of the team and welcoming us in the best way possible.

Furthermore, we would also like to thank our supervisor at LTH, Andreas, who has been additional support and discussion partner. The encouragement and feedback throughout the semester have been precious to us.

Finally, we would like to thank our friends and family for supporting us during the thesis and taking an interest in our project.

Chapter 1

Introduction

When used correctly, machine learning (ML) models are powerful tools for solving complex problems. One of many applications of ML is computer vision, where images are processed to find objects of interest or retrieve other information. ML models used in computer vision are often based on deep learning (DL), where deep neural networks are trained on image data to perform tasks such as classification and object detection. However, one of the biggest bottlenecks for developing any ML model for computer vision is the limited availability of suitable training data. Furthermore, visual data depicting humans challenges integrity and strict regulations, especially with the General Data Protection Regulation (GDPR) laws recently introduced within the European Union. As integrity and confidentiality become more critical, providing ML models with sufficient training data becomes more challenging.

An emerging research topic that could solve the issues concerning integrity and the lack of data is the introduction of synthetic training data, where the data is not taken from the real world but instead artificially generated. By generating photorealistic synthetic data imagery that looks similar to real-world examples, there is potential to replace the real images used today. This would solve the integrity issues with using images from the real world, and it could also automate the annotation process, thus reducing the resources spent on doing it manually.

This thesis explores the possibility of improving the performance of an object detector by introducing synthetic training data that targets identified weaknesses. The data is generated and modified to compensate for the current real-world training data. By evaluating the model's performance on specific test datasets, the weaknesses in the model are identified. Synthetic data is generated to target a particular type of detection error that the model is doing and modified to generalize to the target domain. The data is then introduced in the model's training, and the new performance is evaluated against the original to see if any improvement can be seen.

1.1 Project Description

As the thesis aims to see if synthetic data can be used to complement real data to improve its performance, a decision was made to find a use case where the available training data is limited. This means that the detector needs to be working with a very narrow classification problem.

The conclusion was to focus on an object detector that determines whether or not a person is wearing a hard hat. The publicly available datasets containing annotations of hard hats are limited, meaning that there are not much data available to train the model. In addition to the limited amount of training data, this is an area where producing synthetic data would be very favorable. Not only for the GDPR laws in the European Union but also the risks that it would mean taking pictures of people without hard hats in an environment where they are obligated to wear a hard hat

due to safety risks.

This thesis is structured into three steps. The first step is a model evaluation. A DL model built with a common backbone is trained on a publicly available dataset. The performance is then evaluated, and a potential targeted improvement is identified. The potential targeted improvement stems from an identified weakness in the model where the intent is to enhance the model's performance on this weakness. This step uses fundamental evaluation metrics and a more detailed evaluation to determine the model's weaknesses and strengths.

The second step is generating the synthetic data. Images featuring synthetic elements are generated here in the form of computer-generated characters. Characters with similar traits as the targeted weakness found are generated and placed in images by a set of tools. The images are modified to resemble real images and adapt to the targeted domain. Furthermore, the images are annotated automatically to avoid the time-consuming task of manual annotation.

The third step includes training a new model with the same architecture and parameters as the initial, but on both real and synthetic data. Multiple different synthetic datasets are used to generate different models and find the best one. The findings are then compared to the initial model to see if any improvements could be seen and if the synthetic data had counteracted the identified weakness. As a final detail, it was investigated how the amount of real versus synthetic data in the training dataset and how training with a frozen backbone affected the model's performance.

1.1.1 Synthetic Data

As synthetic data is a central concept of this thesis, the following sections will comment on synthetic data, both its definition and the potential advantages of its usage.

1.1.1.1 Definition

This thesis explores the use of synthetic data to improve an already existing DL model. However, the use of synthetic data raises the question of how we distinguish "real" data from synthetic. On one end of the spectrum, synthetic data can be entirely synthetic, where both objects and backgrounds are created artificially. On the other end, there is the completely "real" data, where the data comes from a photo or video representing the real world. It is essential to notice that synthetic data is also applicable to other types of ML, where visual data is not present. However, since this thesis focuses on visual data, that is what the prominent examples will include.

Furthermore, there is modified real data in the gray zone between synthetic and real data that has been altered with filters and rotation. There are also synthetic data in real environments, as used in this thesis. The use of real environments, captured with cameras, combined with synthetic characters, makes for the possibility of generating large amounts of training data. This thesis defines synthetic data as an image where a real background is combined with animated characters.

1.1.1.2 Benefits

As mentioned in Section 1.1, there are multiple potential advantages of introducing synthetic data as a complement to the real data when training a deep learning model. Below, a list of possible advantages of using synthetic data is presented, followed by a further description of each advantage.

- Avoiding security risks
- Protecting integrity
- Automated processes
- Creating targeted data

One of the possibilities of using synthetic data is diminishing the need for risk-taking that comes with creating real video samples without hard hats in potentially dangerous environments. This applies to the hard hat detection use case considered in this thesis. The creation and use of synthetic data imply that no test data with real people without hard hats must be collected in environments where a hard hat is required to stay safe.

Furthermore, the reduced time spent on administrative issues regarding integrity laws with filming and photographing people can be seen as an advantage of using synthetic data. Using data containing humans raises questions concerning integrity and privacy. However, by using data with rendered characters instead of real people, these issues can be disregarded.

Generating training data from a 3D rendered world also comes with the possibility of generating a lot of data in a relatively short period. In addition to the automatic data rendering, there is also the possibility to automate the annotation process when using synthetic data.

Finally, the synthetic data can be customized. This opens up the possibility of finding and creating specific cases where the current model does not behave as desired, complementing the existing training data with these images, and improving its performance. Targeted improvement is the used notation for this process in this thesis. By identifying a model's weaknesses, data can potentially be created to complement these weaknesses. For example, suppose a model is significantly more confident in detecting an object in class A than class B. In that case, it may be caused by an in-balance between the two classes and hopefully altered by adding synthetic data of class B when training the model.

1.2 Research Questions

The purpose of this thesis can be broken down into three research questions:

- How can synthetic data be generated to complement real training data?
- Does introducing synthetic training data result in an improvement of the model?
- Does introducing synthetic training data result in any unwanted side effects in the object detection?

1.3 Delimitations

Some delimitations were defined in the thesis to fit the time frame given and focus the project. One delimitation set in the project was the model used. The initial model, solely trained on real data, used in this thesis was created in collaboration with Axis. The model parameters were never changed between training sessions. The decision to use the same parameters when training was taken in order to isolate the effect of introducing synthetic data.

Another delimitation is the side effects that the thesis will study. The evaluation will focus solely on the model's performance, meaning no deep dive into analyzing the different layers and what features the model is focused on. Instead, the model will be treated as a black box, trying to find side effects in terms of odd predictions and output behavior.

1.4 Work Distribution

Most of the work was done together, including the analysis of the results, previous work research, and thesis writing. However, Katja Palmkvist took more responsibility for setting up the model training codebase, and Olivia Mattsson focused on the automatic annotation process. Both were involved in the generation process of synthetic data, creating characters and backgrounds, and setting up the scene.

1.5 Related Work

Training neural networks on synthetic data have had some exciting results in the last decade. This section presents some of the most recent and relevant research outcomes. The work is split into three categories; training on synthetic data and its result, domain generalization, and some techniques for generating synthetic data.

1.5.1 Training on Synthetic Data

This section presents some essential research findings on the subject of training on synthetic data relevant to this thesis.

Liu and Mildner investigated the effects of introducing synthetic data in an object detector trained on vehicles[1]. First, a synthetic dataset sampled from the video game *Grand Theft Auto V* was introduced. The model was trained on the dataset with a frozen backbone on different vehicles, and the results were compared to training on real data. The model trained on synthetic data was later fine-tuned by adding additional real data. The conclusion was that introducing synthetic data in cases where real data is limited showed great promise.

Synthetic data is not restricted to still images. For example, Ballout et al. used synthetic video streams to train an action categorization model [2]. The research in the report showed that training with synthetic data improved model performance. Good results were presented for models trained with a combination of real and synthetic data and models trained with only synthetic data.

Wood et al. conducted a project using exclusively synthetic training data [3]. The article focused on creating photorealistic faces to replace real data when creating a face detection model. The faces used were rendered by combining a face model with an extensive library of artist-created assets. This resulted in photorealistic images and a synthetic model as good as non-synthetic models with the same use case. The experiments performed in this project showed both benefits and disadvantages of using synthetic data, one of the benefits being the automated annotations with pixel precision. However, the project showed that one of the disadvantages of using photorealistic data was the high rendering cost.

Harrysson used superimposition, where objects were placed randomly in a background without context, to improve a license plate (LP) object detector [4]. The results showed that the synthetic data gave promising results, with the best performance accomplished with a combination of synthetic and non-synthetic data.

There have been some disputes regarding the most efficient way of training with synthetic data. Some studies have shown better results from freezing the backbone weights [1], while others show the contrary. Tremblay et al. found that freezing the backbone weights reduced the performance of their model pre-trained on ImageNet [5]. They trained an object detector on synthetic data, and similar to [1], they fine-tuned the model on real images. They speculated that their results could be motivated by the large variety in their generated synthetic dataset.

Instead of looking at vehicles or detailed faces, this thesis will examine if similar results can be found using synthetic data depicting humans. Furthermore, this thesis will train on a combination of synthetic and real data, no fine-tuning will happen, and the backbone will remain unfrozen.

1.5.2 Domain Generalization and Randomization

How well a DL model generalizes to the real world is important to consider when creating training data for the model. This section touches on the subject of domain generalization and randomization.

When discussing the usability of synthetic data when training neural networks, an important aspect is the cross-domain generalization, i.e., how well does the generated synthetic data generalize to the real-world data? Tobin et al. used a form of domain randomization when they generated their images. The lighting, poses, and textures were randomized in a way that is not photorealistic

[6]. By providing diverse training data produced under these different settings, they were able to train solely on synthetic data and gave promising results when performing object detection on a benchmark dataset.

Ballout et al. also discuss the impact of domain randomization [2]. Characters performing different actions were placed on different backgrounds. In order to satisfy domain randomization, the camera was shaken, different characters were used, and different lighting was selected, all randomized. The actions were also randomized together with different characters. One of the aspects examined in the project was the importance of the background in the synthetic video clips. According to the article, the background is one of the main challenges when generating synthetic data. The research showed that the simplified synthetic data, without the background, performed well. Therefore, the conclusion that background has a minimized impact on action categorization could be made.

Tremblay et al. also performed domain randomization in their paper, where cars were randomized from a set of models, colors, and positions and used as training data for three state-of-the-art object detection models [5]. They found that domain randomization is a good tool for improving the performance of object detectors trained on synthetic data.

While similar techniques as described above will be used to achieve domain generalization, this thesis will use relevant backgrounds in the synthetic data and a mix of real and synthetic training data.

1.5.3 Data Generation

This section will present some research results from creating synthetic visual data, relevant techniques used in earlier articles, and how they correlate to this thesis.

Dwibedi et al. used a technique that they refer to as "cut-and-paste," where images were put on backgrounds to mimic real images [7]. When these images were combined with real data as training input, they performed better than the model that was only trained on real data.

As mentioned earlier, Harrysson also generated synthetic data by superimposing objects in an environment where they normally would not be found [4]. The project resulted in an automated technique for creating synthetic Swedish LPs. After the basic LP was generated randomly, its brightness was adjusted. Furthermore, noise, motion blur, and JPEG compression were added randomly to the LP. The LP was also rotated first around the XY-axis and then the Z-axis. Finally, CycleGAN [8] was used to improve the "realness" of the LP by introducing realistic features, such as shadows and dirt.

In this report, synthetic characters were put in a real environment, which can loosely be related to the cut-and-paste technique. However, the characters in this thesis were placed in regard to context to mimic real scenarios.

Chapter 2

Background

This section covers the relevant theory needed to follow the report, starting with essential information about ML, computer vision, and object detection. Fundamental evaluation metrics in ML classification and object detection tasks are then introduced, followed by a short description of techniques used when creating the synthetic images.

2.1 Machine Learning in Computer Vision

Computer vision is one of many fields of artificial intelligence (AI), where systems process visual input such as images, and make decisions based on the information available [9]. In the early years, features deemed necessary for decision-making were manually extracted and fed into hand-crafted algorithms. These algorithms then produced an output, solving the task that the algorithm was given [10, p.191].

When the ML field grew, it became an essential part of computer vision. Instead of writing long and complex algorithms by hand, ML were introduced to process the manually extracted features and produce an output [10, p.191].

ML algorithms can be divided into two categories—supervised and unsupervised learning. Supervised means that the algorithm uses input-output label pairs when training, whereas unsupervised learning only uses the input. The labels in supervised learning are used as a ground truth, i.e., what the algorithm's output should be when faced with that input. The ML output could either be continuous or discrete depending on the task the algorithm is supposed to solve. Continuous output is used when trying to solve regression tasks where a trend should be found in the input data, whereas discrete output is used for data classification. In classification tasks, the input to the algorithm should be given a class that it belongs to. Classification is the most common task in computer vision, with applications ranging from image classification to object detection [10, p.191]. Since neither regression nor unsupervised training is used in this thesis, they will not be described further.

In supervised classification, the input-output label pairs $(\bar{x}_n, d_n)$ consist of an input vector $\bar{x}_n$ and the label d_n that shows what class $\bar{x}_n$ belongs to. The input vector $\bar{x}_n$ is the n:th input vector in the set of input-output label pairs. The dimensions of d_n vary depending on how many classes the model has. In the simplest case with two classes, called a binary classifier, d_n is a scalar. For more classes than two, d_n is a vector with the same length as the predicted number of classes.

ML needs a training phase to adapt the algorithm parameters to its task. The input-label pairs that is used during this phase is commonly known as the training dataset. In addition to the training dataset, a test and a validation set is often used. The validation also plays an important role during training, whereas the test set is used to see how well the model generalise to new unseen data.

The basics of training a neural network will be described below, before introducing the type of network that is most commonly used for processing visual input and, finally, the object detector.

2.1.1 Deep Neural Networks

Deep Neural Networks (DNN) have replaced the traditional ML algorithms' in computer vision. Instead of manually extracting the features from a given image, the DNN is directly given the input image, extracting features and, for example, producing a classification result. One of the essential components of a DNN is the neuron node.

Each node takes its inputs and performs a weighted sum of them. Each input, x_i, is multiplied by the node's weight for that input, w_i. The node's bias, b, is added to the sum s before a non-linear activation function $\varphi(s)$ is performed on the result and produces output y [10, p.191]

$$y = \varphi(s), \qquad s = \sum_{i=1}^{n} w_i x_i + b.$$

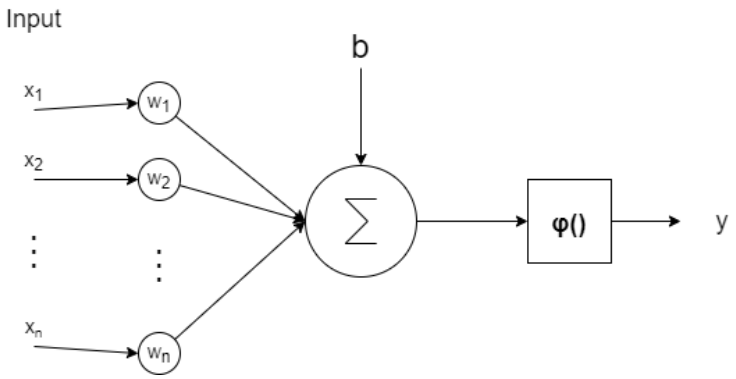


Figure 2.1: A visualization of the neuron. The neuron performs a weighted summary of the inputs $\{x_i\}_{i=1}^{n}$ and their corresponding weight w_i, adds the bias b, and then performs the activation function φ. The result is then passed on in the network, either as input to the next layer or as the network's output.

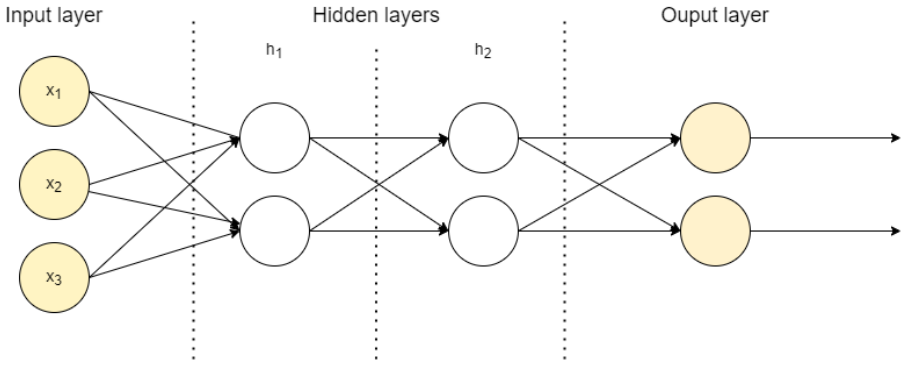


Figure 2.2: Architecture of a DNN with three input features, two hidden layers with two neurons each, and two output neurons.

A visual representation of the neuron can be seen in Figure 2.1.

A DNN can contain various amounts of nodes, and they can be split into different hidden layers. The neuron produces an output that gets "fed forward" in the network, acting as input to other nodes in the next layer. Networks using this process are called *feedforward* networks. Eventually, the nodes in the output layer receive the processed input and produce a result [11, p.659]. A DNN also uses *backpropagation* to learn and adapt to its tasks, meaning that the network will adapt its weights and biases depending on the error of the prediction [11, p.718]. An overview of a simple DNN with two hidden layers can be seen in Figure 2.2.

2.1.2 Activation Function

The activation function, $\varphi(s)$, is a vital step in a neural network. If no activation function would be applied to the sum of the weighted values of the input, an entire network could be reduced to a single neuron [11, p.674]. The lack of activation function reduces the complexity of the problems that the network would be able to solve. To avoid this reduction, a non-linear activation function is introduced in the hidden nodes. There are a lot of different activation functions that can be used. One famous example is the rectified linear unit (ReLU) activation function, where if the weighted sum is negative, the output simply returns 0 [11], i.e.,

$$\varphi(s)_{\text{ReLU}} = \max\{0, s\}.$$

There have recently been introduced modifications to the ReLU functions, for example ReLU6, where its max value has been limited to 6. This increased robustness when using computations with low precision [12].

As a final step in the network, an additional activation function is applied in the output neurons. This output activation funciton can be both non-linear and linear. One example of an output activation function used for binary classification tasks is the non-linear activation function *sigmoid* [10, p.195], defined as

$$\varphi(s)_{\text{sigmoid}} = \frac{1}{1 + e^{-l}}.$$

Sigmoid is used to, given a binary classification task, map the model output to $[0, 1]$. *Softmax* is a generalization of sigmoid and based on Baye's rule, producing the log-likelihood of the class when given input vector x. Softmax can then work for multiclass classification, where each class gets mapped to $[0, 1]$ and the sum of all outputs always is equal to 1. A classification determining the likelihood in this manner is called a Bayesian classification. [10, pp.194-195] For further information about Baye's rule and log-likelihood, see [10, Ch.5.1.2].

2.1.3 Optimization

The focus of neural networks is to fine-tune their weights during training to match the algorithm output to the label, optimizing its performance on the inputs using backpropagation. To allow this fine-tuning, an evaluation is performed, to determine how well the model performed its prediction. There are a lot of optimization methods today, with one of the most important ones for DL being based on the Gradient Descent algorithm. However, before the details of Gradient Descent are described, one needs to know what to optimize [13].

Loss functions $E(\bar{w})$ are the most common way of evaluating the performance in a neural network. The loss function compares how the output y of the model compares to the target d. By differentiating $E(\bar{w})$, the weights can be updated, and small steps towards the function minima can be taken using the gradient. The gradient is the derivative of $E(\bar{w})$, and by taking steps in the direction where the gradient is negative to idea is to find the weights for where the output is as close to the targeted label [11, p.716]. There are loss functions developed to perform on different types of tasks, with the two most dominant being the mean-squared error (MSE) and the cross-entropy error (CEE) functions. While MSE is used for regression tasks, CEE is used for classification. For

both cases, the model prediction is compared to the input label. If the prediction is correct, the error will be set to 0, thus not updating the weights. If the prediction is incorrect, the model uses the implemented loss function to calculate how far off it was, and adjust its weights.

When performing multi-class classification for c classes with N training input-output label pairs, the loss function is expressed as [11, p.149]

$$E(\bar{w})_{multi} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{i=1}^{c} d_{ni}\ln(y_{ni}).$$

The gradient descent method is initialized with a learning rate η. η is a hyperparameter that is manually set before training. It then iterates over all of the training input-label pairs, computes y_n and gradient. Finally, these are used to update the weights as

$$\bar{w} = \bar{w} - \frac{\eta}{n}\sum_{i=1}^{n}\nabla E_i(\bar{w}).$$

Instead of using the entire training dataset to update the weights, making the algorithm slow for large datasets, Stochastic Gradient Descent (SGD) picks a subset of input-label pairs at random. It then computes the gradient for the subset. Due to its stochastic characteristics, it might reach good weights but perhaps not optimal. It has a higher chance of getting out of local minimas in the loss function as well, due to the same characteristics [14, Ch.4, p.124].

There are multiple modifications that can be done to the weight update formula to increase efficiency. One example is the introduction of a momentum term. One can visualize a ball rolling down a hill - it starts slow and builds up momentum while rolling down the slope. As soon as the hill starts to point upwards, the ball will lose speed, stop and eventually change direction towards the lowest point. This idea was in mind when the momentum term m was added [14, Ch.11,p.352], such that

$$m \longleftarrow \beta m - \eta\nabla E_i(\bar{w}),$$
$$\bar{w} = \bar{w} + m.$$

As long as the gradient stays the same, the weight update will take bigger steps until the gradient changes, implying that a minima has been missed. A common value for the momentum term is 0.9 [14], which is also used throughout this thesis.

2.1.4 Regularization

Optimizing the model based on training data is not enough - it should be able to perform well on data that it has not seen before as well. When the model performs well on the training data but poorly on new data, the model does not generalize well and has been overfitted to the training data. By introducing a test set that includes input-label pairs that the model has not seen before, the model's ability to generalize can be estimated and compared to the performance on the training data [11].

There are a lot of regularization techniques used to prevent overfitting, for example, weight regularization, early stopping, and node dropout. Dropout introduces a probability of which a node is dropped during each training step to avoid too much collaboration between nodes [14, Ch.11, p.365]. L1 and L2 regularization are two common weight regularizers, where L2 introduces a regularization term that punishes weights that are too big. This constraint on the weights can prevent the model to overfit [14, Ch.11, p.364]. Both of these methods will be used in this thesis, but will not described further in greater detail. Readers are referred to [11] and [14] to find out more.

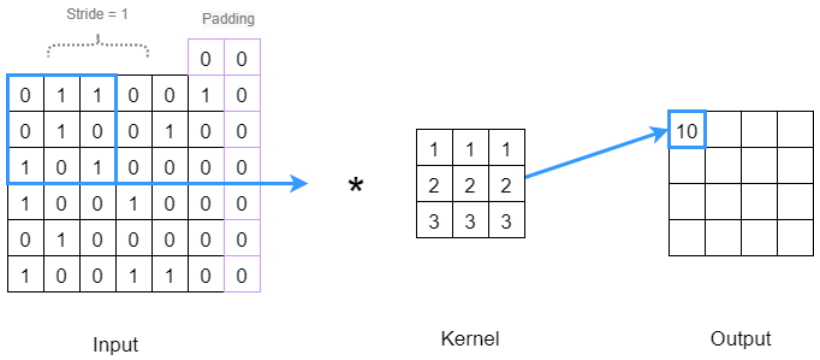


Figure 2.3: An example of the convolutional operation in a convolutional layer. The first 3×3 cells from the input matrix get convoluted with the kernel and produce the output 10 in the first cell of the output matrix. If the stride were set to 1, the kernel would move 1 step across the matrix and produce the subsequent output. The figure also shows the concept of padding, represented by zero-valued cells around the input matrix.

2.1.5 Convolutional Neural Network

Convolutional Neural Networks (CNN) are a type of DNN commonly used for analyzing images. CNNs makes use of convolutional filters to increase or reduce dimensions and retrieve important information. The essential part of a CNN is, as the name implies, the convolutional layer. This layer is a necessity in a CNN, but it can also contain other types of layers [14, Ch.14, p.448]. Convolution has multiple usages across a lot of mathematical and physical fields. Convolutional layers make use of a kernel K, a matrix containing the weights that are convoluted with an image matrix I, producing a result H [13, Ch.9] (See Figure 2.3), such that

$$H(i,j) = (I * K)(i,j) = \sum_{m,n} I(i+m, j+n)K(m,n).$$

The kernels can have varying shapes and sizes depending on the tasks given, such as finding edges or blurring the image. Since multiple convolutions can be reduced to a single convolution with both kernels, in the same ways that a neural network can be reduced, a non-linear activation function φ is used to avoid this. A combination of a kernel and activation function is referred to as a filter [13, Ch.9], i.e.,

$$F(i,j) = \varphi(b + H(i,j)).$$

These filters slide over the input data and perform a weighted, element-wise multiplication of the filter and the input, producing a sum. The filter dimensions determine how many of the surrounding pixels that are used in the multiplication. Furthermore, the stride parameter determines how many pixels the kernel should move between each multiplication, and padding can add rows and columns to the edge of the image to determine the dimensions of the output matrix [14, Ch.14, p.453].

During training, the convolutional layer will learn what filter to use by adapting its weights. A convolutional layer where all of the neurons use the same filter will produce a *feature map* that emphasizes the parts of the image that trigger the filter the most [14, Ch.14, p.450].

The architecture of the nodes in a convolutional layer differentiates from a traditional layer, where a convolutional layer is not *fully connected*. This means that only some of the nodes in a previous layer are used as input to the next. Another property of a convolutional layer is the usage of *shared weights*, where the nodes do not have a unique weight but instead share them within the channel. These properties result in fewer weights to update than with a fully connected network

with individual weights. This advantage is significant when using images as input, which often are large in size [14, Ch.14, p.447].

2.1.6 Object Detection

Object detection is a specific task in computer vision, where the image might contain multiple objects that need to be classified. Instead of splitting the image into smaller images and performing standard classifications on them, a detector is used. The detector searches the image after regions that might contain the specific objects [10, p.302]. Object detection can be performed with both ML and deep learning (DL). CNN architectures are ubiquitous in object detection DL models and are used in the most popular ones today.[14, Ch.14, p.492] Supervised object detection models require annotations of the objects in an image. The annotation typically consists of a file with image coordinates for the objects' bounding boxes and their corresponding classes. These annotations are typically called ground truths.

The object detectors produce, together with a predicted bounding box and corresponding class, a confidence score. The confidence score of each prediction shows the confidence of the model in that particular prediction, i.e., the probability that the object belongs to a certain class [14, Ch.14, p.492].

As the architecture of CNN object detectors often is complex, it is common to base the model on a pre-trained network. They often have an architecture based on two subnetworks - the backbone and the head. The backbone extracts features from an image for the head to perform classification on. It is possible to freeze the weights in different layers, such as the entire backbone, and let the other weights be trainable. As mentioned in Section 1.5, the opinion on whether the backbone should be frozen while training on synthetic data is disputed.

Many different backbone architectures exist today for use on object detection, each with their own benefits. One of them is EfficientNet, which was chosen as the backbone in this work.

2.1.6.1 EfficientNet

As mentioned, a common backbone is the EfficientNet [15]. EfficientNet balances the network's width and depth with the image resolution using a *compound coefficient*. It is developed with the idea that if an image is of a higher resolution, the network needs more layers and channels to perform well. By balancing the number of layers and channels depending on the image sizes, the EfficientNet model scales easily and performs very well on the ImageNet dataset compared to state-of-the-arts models, but with fewer parameters [15]. EfficientNet uses ReLU6 as its activation function, limiting the maximum output to 6. EfficientNet is a computationally cheap alternative to other backbones [15].

2.1.6.2 Single-Stage Networks

The two types of networks that can be used as a head of the model are the two-stage and single-stage networks. While two-stage networks can be exact and accurate in their predictions, they are slower than single-stage networks. Single-stage, as mentioned, are typically faster but might not be as accurate, smaller objects, for example, might be a problem [16].

Instead of using two separate networks for proposing regions and classifying the object inside those regions, the single-stage network uses one that does both. Today's popular single-stage networks are the YOLO (You Only Look Once) models and Single Shot Multibox Detector (SSD) [17] [10, p.306]. The SSD uses default bounding boxes of each feature map, allowing the bounding boxes to be of different shape [17]. SSD then predicts the class and offset from these default bounding boxes by applying convolutional filters on feature maps. It was found to be faster than, at the time, the state-of-the-art single-stage network YOLO and perform as well as the two-stage networks in terms of accuracy [17]. SSD is the head architecture that will be used in this thesis work.

2.2 Evaluation Metrics

When comparing ML models, there is a need to evaluate the performance. For example, the most common way to evaluate the model in ML classification tasks is to see how well it can classify the different inputs. Object detectors also introduce the aspect of finding the object in an image. This section presents some of the most common ways of evaluating object detectors.

Four measurements are commonly used in classification performance tasks. The measurements represent how the model predicts the different objects compared to their actual class. If we apply these measurements to object detection, they can be defined as:

- **True Positive** (TP) - The model predicted an object of a certain class within a certain location, and it was correct.
- **False Positive** (FP) - The model predicted an object of a certain class within a certain location, and it was incorrect.
- **True Negatives** (TN) - The model did not predict an object of a certain class within a certain location, and it was correct.
- **False Negatives** (FN) - The model did not predict an object of a certain class within a certain location, and it was incorrect.

When applying these measurements to object detection, the amount of FN is determined by how many ground truth boxes exist where the predictions did not produce one in the exact location and/or misclassified the object. TN is where the ground truth did not contain a box, and the object detector did not produce a prediction. The TNs of an image are all possible predictions in the background that are not predicted by the object detector. What is ideal is a high amount of TP and TN - the model should predict the input data as well as possible.

There is another essential part of performance measurements called the decision threshold. The output of the network will return a probability that the object is a part of the given class, i.e., the confidence score described in Section 2.1.6. The decision threshold can be manually adjusted after training the model. The set threshold value will affect the number of correct guesses the model has. A low threshold, meaning that the model does not have to be that confident, will allow the model to predict more often. This might result in more misclassifications, but a very high threshold might miss many detections instead.

2.2.1 Accuracy

Dividing the number of correct observations by the number of total observations gives the accuracy, $\mathrm{A}_{\mathrm{accuracy}}$, of the model, as in

$$\mathrm{A}_{\mathrm{accuracy}} = \frac{TP + TN}{TP + FP + TN + FN} \qquad 0 \leq \mathrm{A} \leq 1$$

The accuracy can be used as a measurement of how well the model performs. When using this metric, one must remember that it might not reflect the overall performance when using an unbalanced dataset. To maximize the accuracy, the model might favor predictions of the overrepresented classes. In object detection tasks, it could be favorable for the model not to make any predictions. Since each image has a lot of negative bounding boxes, TNs, training not to do any detections will give the model good accuracy. Accuracy is therefore not a good measurement in these tasks.

2.2.2 Precision and Recall

Instead of solely looking at the accuracy of the model, precision and recall are two complementary measurements used to see how well the model correctly classifies the objects. Precision is a measurement of how many of the predicted positives actually were positive by dividing the TPs by the sum of TP and FP according to

$$\mathrm{P}_{\mathrm{precision}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} \qquad 0 \leq \mathrm{P}_{\mathrm{precision}} \leq 1$$

A high precision in a model is equivalent to a model that does not produce a lot of false predictions.

Recall shows how many of the objects present in the data were predicted correctly. A high recall means that the model does not miss many objects. Recall is defined as

$$\mathrm{R}_{\mathrm{recall}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} \qquad 0 \leq \mathrm{R}_{\mathrm{recall}} \leq 1$$

Both precision and recall can be produced for a single class. They can also be produced for the average of all existing classes in the object detector.

2.2.3 F1-score

High precision and recall are desirable, but they can sometimes be hard to balance. By improving precision and forcing the classifier to be more precise, the recall score might decline, thus, missing some of the objects. The F_1-score takes the harmonic mean of precision and recall. Given that precision and recall are upper-bounded by 1, the F_1-score has the same upper bound. A high F_1-score comes from both precision and recall being high. F_1-score is defined as

$$\mathrm{F}_1 = 2 * \frac{\mathrm{P} * \mathrm{R}}{\mathrm{P} + \mathrm{R}} \qquad 0 \leq \mathrm{F}_1 \leq 1$$

2.2.4 Average over Precision-Recall Curve

This section will describe the average over Precision-Recall curve, and average precision (AP), as a measurement.

2.2.4.1 Intersection over Union

A critical aspect of the detector is how well the model can spatially identify an object. Where is the bounding box for the object prediction, and where is the ground truth bounding box? The so-called Intersection over Union (IoU) is the area of the overlapping boxes divided by the union of the areas, as in Figure 2.4. IoU is the value for a threshold when we say that a prediction is correct, i.e., how much overlap between the two bounding boxes is necessary to be classified correct? As the measurements above, IoU is bounded between 0 and 1. It determines when something is classified as a TP or FP. With a threshold set to 0.5 an detection with an IoU of 0.55 would be classified as a TP and a detection with 0.45 would be classified as a FP. An IoU threshold of 0.5 is the most common used today when evaluating performance.

2.2.4.2 The Precision-Recall Curve

Recall and precision will vary depending on the set threshold value. With a high threshold, precision is increased. Less of our TP will be classified correctly, but the ones classified as positive will, with a high probability, not include any FP. However, the higher threshold also decreases the recall. Misclassifying a lot of the positives will lead to a lower recall. This precision and recall relationship can be combined and visualized in a curve - the Precision-Recall curve. An example of the curve is shown in Figure 2.5.

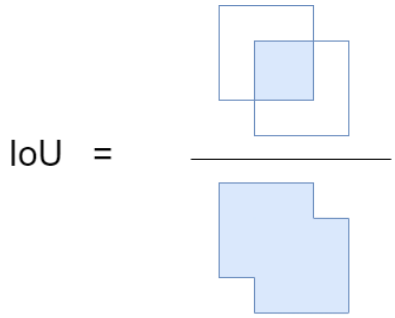


Figure 2.4: A visualization of how IoU is calculated. The area of the overlapping ground truth and the predicted bounding box is divided by the union of the boxes.

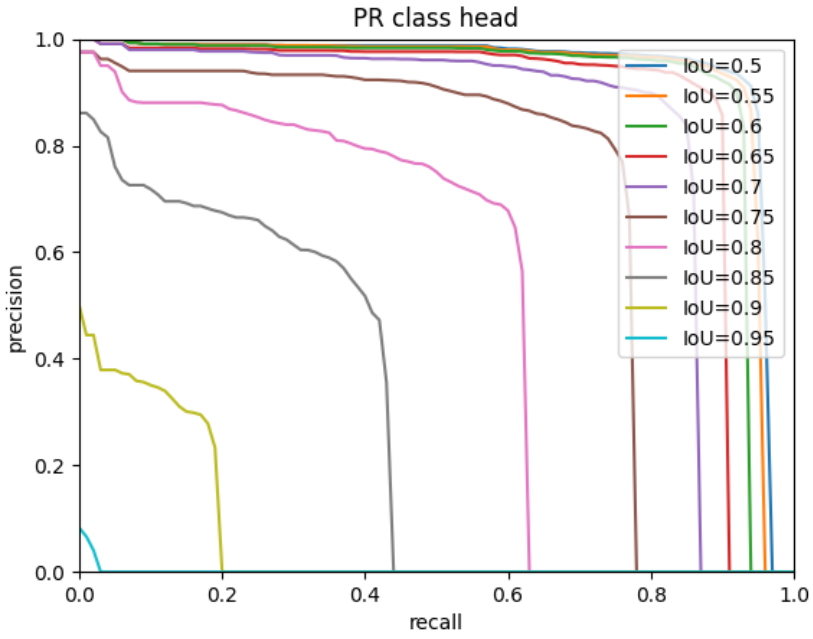


Figure 2.5: An example of a Precision-Recall curve. The lines show the different Precision-Recall curves depending on the IoU threshold, from 0.5 to 0.95. An optimal curve is similar to the one shown in this figure for IoU 0.5, where precision is close to 1 for recall values until recall reaches 1.

2.2.4.3 Average Precision

The F_1-score gives a single estimate of the Precision-Recall curve at one threshold and produces the F_1 score. On the other hand, AP is the integral of the Precision-Recall curve. This integral is estimated with several points located on the curve in practice. The formula for calculating the AP is as follows:

$$\mathrm{AP} = \int_0^1 \mathrm{P(R)}dR$$

Combining the AP of all of the classes gives the mean Average Precision (mAP)

$$\text{mAP} = \frac{1}{N} \sum_{n=1}^{N} \text{AP}_n$$

where N is the number of classes. With mAP, evaluations between different iterations and models can be done considering all predicted classes and locations. The mAP is a typical measurement when evaluating object detectors, such as in the COCO evaluation metrics described in Section 2.2.5.

2.2.5 COCO Evaluation Metrics

The COCO (Common Objects in Context) dataset is a commonly used object detection dataset, sponsored by CVDF, Microsoft, Facebook, and Mighty AI [18]. Together with the dataset, an API that enables evaluation is provided. COCO evaluation is used in this report to evaluate the different object detection models. The same evaluation metrics used in the benchmarking of the COCO dataset is used in this thesis.

The results from the COCO evaluation contain different AP, and Average Recall (AR) metrics [19]. The AP and AR metrics are sorted by bounding box size, IoU, and maximum detections. The three bounding box sizes used are small, medium, and large, where small ranges from 0 to 32^2, medium ranges from 32^2 to 96^2, and large implies bounding boxes are of size 96^2 and larger. The sizes are in pixels.

Additionally, the AP and AR scores are calculated separately for each class, including a mean AP and AR for all classes. An example of the results from COCO evaluate can be seen in Figure 2.6 below.

```
RESULTS ALL CLASSES:
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.608
 Average Precision  (AP) @[ IoU=0.50      | area=   all | maxDets=100 ] = 0.949
 Average Precision  (AP) @[ IoU=0.75      | area=   all | maxDets=100 ] = 0.704
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.362
 Average Precision  (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.615
 Average Precision  (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.675
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=  1 ] = 0.187
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets= 10 ] = 0.647
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=   all | maxDets=100 ] = 0.670
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = 0.466
 Average Recall     (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.676
 Average Recall     (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.739
```

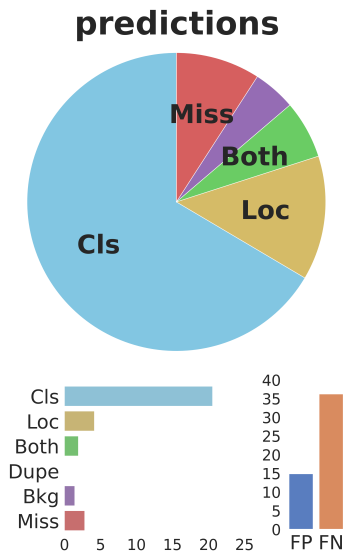
Figure 2.6: COCO evaluation results example. It returns AP and AR for different IoU thresholds, as well as for different area sizes.

2.2.6 TIDE

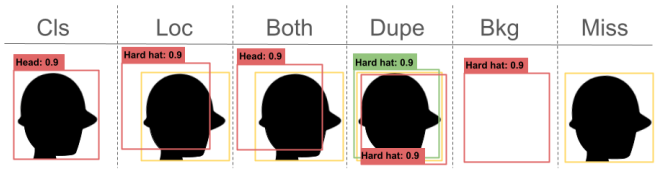
The TIDE (A General Toolbox for Identifying Object Detection Errors) measurements were created to identify object detection errors [20]. In TIDE, the detection errors (FP and FN) from the evaluation are divided into multiple categories for a more detailed evaluation of the model results. The results are concluded in a figure like the one in Figure 2.7a. The different categories, with their associated explanation, are presented below, and a graphical explanation can be seen in Figure 2.7b.

- Cls (Class) - The class is incorrect, but the prediction bounding box is placed correctly, framing an object of another class than the one predicted.

- Loc (Location) - The location of the prediction is incorrect, but the class is correct. Meaning that the prediction bounding box is not placed correctly, but contains parts of the object.
- Both (Class + Location) - The class and the location is incorrect. A combination of Cls and Loc, the prediction bounding box is placed incorrectly, while including part of an object of another class than predicted.
- Dupe (Duplicate) - The object was detected more than once, there are multiple prediction bounding boxes associated with the same ground truth bounding box.
- Bkg (Background) - The prediction bounding box marks a part of the background, containing no object, or part of an object.
- Miss (Missed) - The object is not found during prediction.



(a) TIDE example diagram. The pie chart shows the numerical proportion between the prediction errors, and the amount of each error is shown in the table at the bottom left. The amount of FP and FN is shown to the right.



(b) A figurative explanation of the TIDE errors that is explained in the list above.

Figure 2.7: TIDE results and explanation.

2.3 Rendering Techniques

The sections below describe standard techniques used when rendering synthetic images. The techniques include ray tracing, projective geometry, and image morphology.

2.3.1 Ray Tracing

Ray tracing is a technique used when rendering digital images. The technique is used to simulate how the lighting interacts with a scene. The tracing considers interactions between objects and light and results in each pixel's final color value [21]. In Figure 2.8, an example of the reflection and refraction of the rays can be seen. The primary application of ray tracing is a wide range of computer graphics, architecture, engineering, and lighting design. In computer graphics, ray tracing is used both in non-real-time and real-time applications. For example, dome lighting uses a simple ray tracing technique, using photos of skies to model incoming reflections and lighting. This technique uses High Dynamic Range Imaging (HDRI). While the ray tracing technique is an implementation that improves the realistic look of digital images, it is also a very computationally intensive technique. However, using a Graphics Processing Unit (GPU) reduces the rendering time significantly.

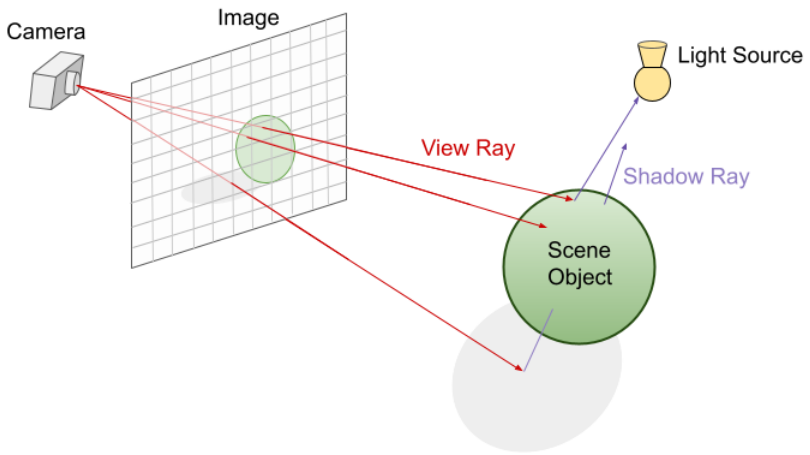


Figure 2.8: Explanation of ray tracing. The rays go out from the camera, pass the image's pixel grids, and reach the scene. The rays hit the objects and are then cast towards the light source, deciding whether the ray will result in illumination or shadow. (Image inspired by Nvidia blog post [22])

2.3.2 Projective Geometry

Since this report covers creating images in a artificially generated world, some equations and definitions used in projective geometry are necessary to understand the steps taken to generate the synthetic data. In addition, the section will cover equations used to translate three-dimensional (3D) coordinates to two-dimensional (2D) coordinates and vice versa.

The perspective projection of an optimal pinhole camera can be modelled using the camera's 3×4 camera matrix P, expressed as [10, p.48]

$$P = K[R \quad t] \tag{2.1}$$

where P is the product of the intrinsic camera parameters in K and the extrinsic camera parameters in $[R \quad t]$, with

$$K = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \qquad [R \quad t] = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{2.2}$$

The intrinsic camera parameters in K include the camera-specific settings such as focal length f, pixel size, skew, and the principal point (c_x, c_y). It is a common convention to have the principal point of the camera at the origin $(0, 0, 0)$. [23, Ch.16, p.502] The intrinsic camera parameters are used to project the 3D camera coordinates on the two dimensional image. A simplified form, without skew, can be expressed as a 3×3 matrix as seen in (2.2) [10, p.46].

The extrinsic parameters, on the other hand, describe how and where the camera is located in the 3D world by a rotation R followed by a translation t. The extrinsic parameters map 3D world points to 3D camera coordinates. The rotation R is a 3×3 matrix describing the rotation around the 3 cardinal axes (x, y, z), and t describe 3D translation by a 3×1 vector.

To understand how the rotational matrix is formed, it is preferable to start with how transformations in 2D Euclidean space happen and then introduce 3D transformations.

In 2D space, the rotation matrix rotates 2D coordinates (x, y) expressed in a column vector by an angle θ to a new point $(\hat{x}, \hat{y})$ [23, Ch.19, p.586]

$$\begin{bmatrix} \hat{x} \\ \hat{y} \end{bmatrix} = R(\theta) \begin{bmatrix} x \\ y \end{bmatrix} \qquad R(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix}$$

Now, in 3D space there are three axes where the vector can rotate around, resulting in three basic rotation matrices for each axis. With a 3D point in space (x, y, z), a rotation around each axis can be computed using the rotational matrices [23, Ch.19, pp.586-587]

$$R_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix}$$

$$R_y(\theta) = \begin{bmatrix} \cos\theta & 0 & \sin\theta \\ 0 & 1 & 0 \\ -\sin\theta & 0 & \cos\theta \end{bmatrix}$$

$$R_z(\theta) = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

The rotation matrix R can then be modelled as an extrinsic rotation around all three axes with the Euler angles α, β, and γ, such that [23, Ch.19, pp.586-587]

$$R = R_z(\gamma) R_y(\beta) R_x(\alpha) \tag{2.3}$$

It is important to note that 3D rotations are not commutative. Performing a xyz-rotation does not produce the same rotation matrix as a zyx-rotation. R as presented in (2.3) is equivalent to a xyz-rotation. With P found, the 3D world coordinates $p_w = (x_w, y_w, z_w, 1)$ can be projected to pixel 2D coordinates $p_s = (x_s, y_s, 1)$

$$p_s \sim P \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \tag{2.4}$$

where $\sim$ is equality up to scale. Projecting points on the ground, where $z = 0$, is then

$$p_s \sim P \begin{bmatrix} x_w \\ y_w \\ 0 \\ 1 \end{bmatrix} \tag{2.5}$$

We can then, due to zero multiplication, turn P into a 3×3 matrix H:

$$p_s \sim H \begin{bmatrix} x_w \\ y_w \\ 1 \end{bmatrix} \tag{2.6}$$

H is called the *homography* matrix. Transforming using the homography matrix preserves straight lines and is how the perspective remains in the transformation. Important to note is that the the resulting coordinates needs to be normalized, since H is homogenous [10, pp.33-34].

Since the homography matrix is invertible it is possible to rotate and project the 2D image coordinates back to the 3D world and on the plane $z = 0$ [10, pp.33-34]

$$p_w = H^{-1} p_s \tag{2.7}$$

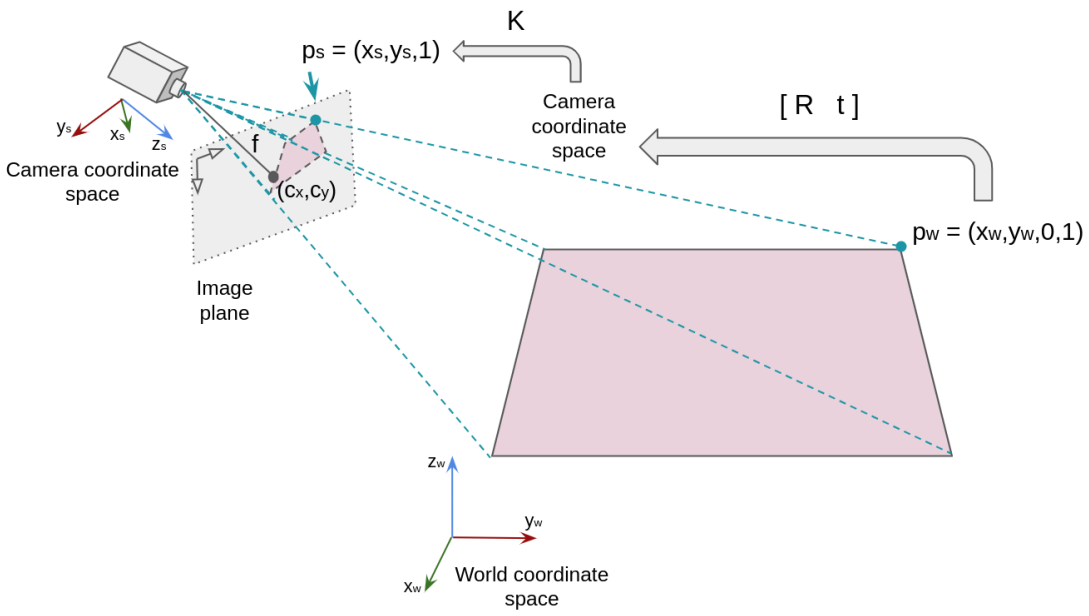


Figure 2.9: 3D world coordinate translation to camera 2D coordinates. The camera extrinsic matrix $[R \quad t]$ maps the world coordinate point p_w to the camera coordinate space. The point is projected on the image plane using the camera intrinsic matrix K containing the focal length f and center of projection (c_x, c_y).

2.3.3 Image Morphology

Morphology is a collective word for a series of shape-based operations performed on images and was used in this thesis in the automatic annotation process. Erosion and dilation are two of the

fundamental operations in morphology most commonly used when processing binary images (it has later been extended to work on grayscale images). Both erosion and dilation use a structuring element, or a kernel, to modify the boundaries of objects in a binary image. While erosion *erodes* away contouring image boundary pixels, dilation does the opposite and adds pixels to the boundaries. Erosion can help remove floating pixels and areas, whereas dilation makes the shapes bigger and close holes [23, Ch.3, p.69]. They are dual operations, meaning they are equal to the complement of each other with the same structuring element [24, p.27].

The kernel (in the binary case) can be represented by a 2D binary matrix with its dimensions set to the kernel size. The shapes can vary from a square or a circle to a line. The amount of pixels eroded or dilated depends on the size and shape of the kernel.

The kernel iterates over the image and performs the respective operation, applying a rule over the neighborhood pixels determined by the kernel shape and size. While looping over the pixels in the image, the pixel of interest gets a pixel value determined by the rule. In the dilation rule, the pixel of interest gets the maximum pixel value of the neighborhood pixels. For erosion, the pixel of interest gets the lowest value [23, Ch.2, p.31]. An erosion followed by a dilation is referred to as a *morphological opening*. Figure 2.10 shows the process, with a visualization of both erosion and dilation [23, Ch.3, p.77].

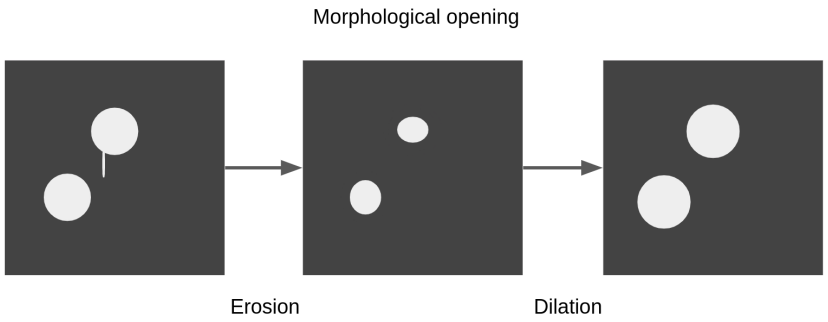


Figure 2.10: Overview of the morphological opening process. The original image (right) to the right is first applied with an erosion kernel, which reduces the circles' boundaries and removes the floating area below the upper circle. Then, dilation is applied to add pixels to the circles' boundaries, making them bigger and similar to their original size.

Chapter 3

Approach

This chapter begins with a description of the tools, datasets, and the model architecture used during the thesis. It continues to describe the evaluation methods used and training methodology. Finally, the chapter ends with a description of how the synthetic data was created and annotated.

3.1 Tools

Several tools were used to analyze the object detector model and render synthetic data. A brief description of each tool used is presented in the sections below.

3.1.1 Fiftyone

Fiftyone is an application created to visualize machine learning datasets [25]. The tool makes it possible to examine the model by, for example, comparing the ground truth annotations with the predictions made by the model. It also has several built-in standard evaluation methods as the COCO evaluation metrics described in Section 2.2.5. Fiftyone can also be used to extract metrics, such as confusion matrices, false positives, and F1-scores. This thesis used the tool to visualize and retrieve information about the datasets and as an evaluation tool. Fiftyone worked as a complement to the COCO evaluation metrics. Since the analysis only returned the metrics and not information about distributions of FP and TP inside each class, Fiftyone was used to extract these metrics manually. The manual process made it possible to identify how the performance varied between different class traits.

3.1.2 Blender

Blender is an open-source tool used to create 3D graphics [26]. The tool can create animated films, visual effects, virtual reality, and computer games. A new version was released during the thesis writing; hence, versions 3.0 and 3.1 were used. Blender has an API that was used to automate the image generation process and served as the data generation tool. Scripts were written in Python, calling the Blender API, and the real-world backgrounds were combined with the generated characters to create the camera scenarios.

3.1.3 Character Creator

Character Creator 3 (CC3) is an application made for creating game characters [27]. In the Reallusion Software Store, different character features, clothes, and accessories can be bought to complement the standard range in Character Creator. The newest version released is Character Creator 4;

however, the application version used in this report is CC3. CC3 was used to create the character library and modify the characters to match the chosen targeted weakness of the object detector.

3.2 Datasets

This section describes the different datasets used during the project. Three datasets were used when training and evaluating the model. As mentioned in the introduction, finding public datasets with the object classes head and hard hat is difficult. Therefore, in addition to the public dataset used for training, there was a need to find additional test data.

3.2.1 Roboflow

The initial model was trained on the public hard hat workers dataset from Roboflow [28] shared by Northeastern University in China. The dataset consists of 7035 images with 27 039 annotations of three different classes - helmet, head, and person. The person class was ignored, and the helmet class was re-labeled as hard hat in this thesis. The images are from photographs showing construction site workers and visitors. The images cover different scenarios, from posing in front of the camera to pictures of them working. The average size of the annotation bounding boxes is 500×333 pixels. [28] The annotation quality varies. Some of the bounding boxes are big compared to their objects, and some are very tight around them.

The dataset is split into three parts where 10% is used to test, 10% to validate, and the remaining 80% for training. Table 3.1 shows the class distribution of the dataset.

Class name	**Number of annotations**	**Percentage of annotations**
Hard hat	19 747	75%
Head	6 677	25%

Table 3.1: Roboflow class distribution with the amount of annotations of each class represented in the dataset.

3.2.2 Summerwork

The Summerwork dataset consists of images from video recordings that summer workers at Axis recorded. The annotated material consists of a few people moving in different ways, with and without hard hats, around the premises of Axis' offices in Lund. It consists of 401 images with 855 annotations of heads and hard hats. The entire dataset was used for testing and evaluating the model. The head class includes people wearing hats other than a hard hat. The class distribution for the Summerwork dataset is presented in Table 3.2.

Class name	**Number of annotations**	**Percentage of annotations**
Hard hat	493	56%
Head	381	44%

Table 3.2: Summerwork class distribution with the amount of annotations of each class represented in the dataset.

3.2.3 Dubai

The Dubai dataset consists of frames from multiple surveillance video recordings from a construction site in Dubai. The recordings differ from close-up to overview views, including different angles. The images were taken from 5 different cameras placed on different parts of the construction site. The Dubai dataset consists of 411 images with 900 annotations. Similar to Summerwork, there are some recurring individuals in the images since they are taken from the same video recordings. The class distribution of the Dubai dataset can be seen in Table 3.3.

Class name	Number of annotations	Percentage of annotations
Hard hat	659	73%
Head	241	27%

Table 3.3: Dubai class distribution with the amount of annotations of each class represented in the dataset.

3.3 Model Evaluation

This section will introduce the initial model, its parameters, and how it was evaluated.

3.3.1 Initial Model

The initial model used in this thesis was based on EfficientNet Lite as a backbone with no frozen layers during training. An overview of the EfficientNet layer architecture can be found in Figure 3.1. The backbone was pre-trained on ImageNet, receiving initial weights before the training. The model was then trained on the Roboflow dataset, using the split described in Section 3.2.1, leading to a training set of 5628 images.

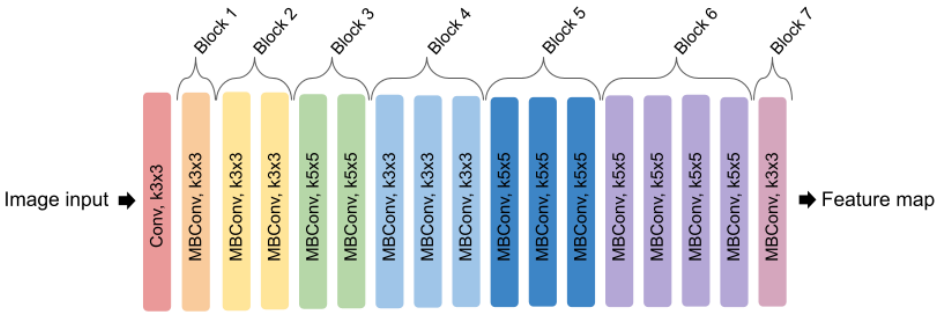


Figure 3.1: Layers in the EfficientNet Lite backbone. Conv, k3x3 refers to a convolutional layer with a kernel size of 3x3. MBConv is a so called mobile inverted bottleneck layer [15].

SSD was used as a head for the model. An architectural overview of the SSD can be seen in Figure 3.2. Each convolutional layer uses the activation function ReLU6. The specifications of the initial model are presented in Table 3.4.

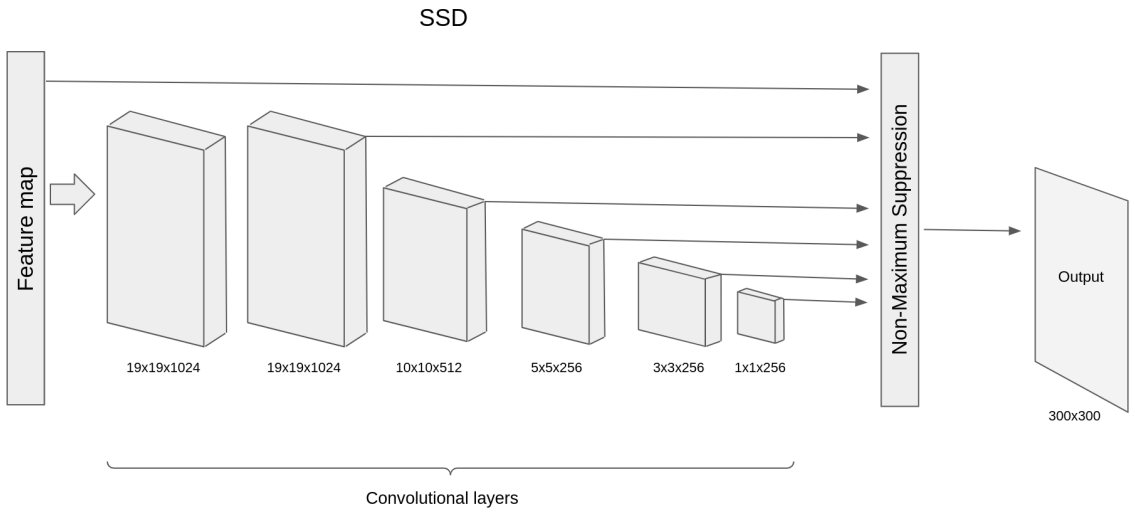


Figure 3.2: An architectural overview of the SSD network. When SSD is used as a head network, the feature map in the image will come from the output of the backbone. SSD consists of six convolutional layers, producing predictions of different default bounding boxes. The prediction includes the offset to the default bounding boxes and the confidence score. (Image inspired by Liu et al. [17])

	Initial Model
Backbone	
Backbone architecture	EfficientNet Lite-0
Backbone activation function	ReLU6
Pre-trained	Yes, on ImageNet
Regularization	Dropout
Dropout probability	0.2
Head	
Head architecture	SSD
Head output activation function	Sigmoid
Optimizer	SGD with momentum
Momentum term	0.9
Regularization	L2
L2 term	0.00001
Training hyperparameters	
Epochs	25
Batch size	8
Learning rate	0.001

Table 3.4: The specifications of the initial model.

3.3.2 Evaluation Method

A set of evaluation metrics and methods were chosen to evaluate the performance of each model. In the sections below, the different evaluation approaches are described.

3.3.2.1 Performance Metrics

As mentioned in Section 2.2.5, COCO evaluate is an API that provides support to evaluate an object detection model. COCO evaluation was used in this thesis to calculate the AP for each class and the mAP.

Furthermore, the precision, recall, and F_1-scores were calculated, corresponding with the best threshold for each model and class. In the Sections 2.2.2 and 2.2.3 it can be found how these metrics are calculated.

3.3.2.2 Evaluation Using FiftyOne

A further evaluation was performed to complement the AP, R, P, and F_1-scores. FiftyOne, described in section 3.1.1, was used in order to perform a detailed evaluation of the performance of the model. First, the tool examined the FPs and FNs by visualizing the model predictions. Then, weaknesses of the model were found from the visualizations by seeing correlations between the wrongly predicted instances. Compared to the performance metrics received from the COCO evaluation, the confidence threshold is always set to 0.4. This is important to remember since the results might not be comparable to the results from the COCO evaluation since the precision, recall, and F_1-score are calculated using the best confidence threshold. This means that the detailed evaluations using Fiftyone can only be evaluated against other FiftyOne evaluations.

FiftyOne was also used to evaluate Confident FPs. A confident FP means that the FP had a confidence score over 0.9 and an IoU higher than 0.5. This metric was used to see if the model became more or less confident on FPs when introducing the synthetic training data. The intuition was that even if the other evaluation metrics remained similar, the amount of confident FPs could decrease and indicate that the model adapted itself to the new training data.

3.3.2.3 TIDE Evaluation

The TIDE evaluation of FPs and FNs was also used to classify the detection errors. More information about TIDE and the different categories of detection errors can be found in section 2.2.6.

3.3.3 Training Method

As mentioned earlier in the report, when training a new model, the same settings were used as for the initial model, see Section 3.3.1.

The only parameters changing between training sessions were:

1. The synthetic dataset used
2. The number of images in the datasets

Later, additional investigations were conducted, where some of the training parameters were updated. This includes changing the ratio between the datasets and freezing blocks of the backbone, as described below.

3.3.3.1 Varying Ratio

When training on both real and synthetic datasets, the training data was kept as two separate datasets. Therefore, a probability was set for each dataset. The probability determined with what probability an image was to be picked for each iteration during training. Each iteration includes an SGD-step as described in Section 2.1.3. Images from each dataset were picked with the probability that is given. The batch containing the selected images will be used to update the weights for each iteration. In this thesis, these probabilities are mentioned as a ratio. For example, given the ratio of 1:0.5, the probability that a frame from the first dataset is picked is twice that of a frame from the second dataset. This thesis will present the synthetic dataset as the second dataset in the ratio.

3.3.3.2 Frozen backbone

The initial and best-performed model was trained again but with a frozen backbone as an additional investigation. The intent was to see if the results would have improved compared to the unfrozen backbone used in this thesis. Since the opinions on how whether the backbone should be frozen or not when training on synthetic data is highly disputed, see Section 1.5, the intent was to see if an improvement could be seen. The bottom four blocks of feature extraction layers were frozen during the training, see Figure 3.1.

3.3.3.3 Dubai Evaluation

The Dubai dataset can be seen as a validation set for the changes introduced to the datasets. Since the intent was to optimize the model's behavior on the Summerwork dataset, it is essential to note that good performance on the targeted set does not always generalize well. Therefore, the idea was to see if the improvement still could be seen on the Dubai dataset that is non-biased of our performance-enhancing targets. The Dubai dataset was also used to get a bigger testset since the number of images in the Summerwork set is low.

3.4 Generation of Synthetic Data

The synthetic data was created by randomizing characters and placing them in a real environment. The characters where combined with the backgrounds in Blender, mentioned in section 3.1.2.

Earlier research has shown, see Section 1.5.2, that domain randomization of synthetic data has provided good results, achieving domain generalization. Several variables were randomized throughout the rendering process to reduce the gap between synthetic and test data, including characters, poses, backgrounds, camera angles, and light. An overview of the rendering process can be seen in Figure 3.3. The following sections will describe each step of the process in depth.

3.4.0.1 Characters

The characters used in the synthetic data were created using Character Creator (CC), described in Section 3.1.3. Access to a library of 200 characters was granted when starting the thesis work. The character library included various people with different hair colors, skin colors, ages, hairstyles, clothes, and accessories. A selection of the characters from the character library can be seen in Figure 3.4.

After evaluating the model and deciding what model weakness to target, the character library was examined to evaluate how many characters already existed with that specific trait. The other characters were then modified to fit the particular trait.

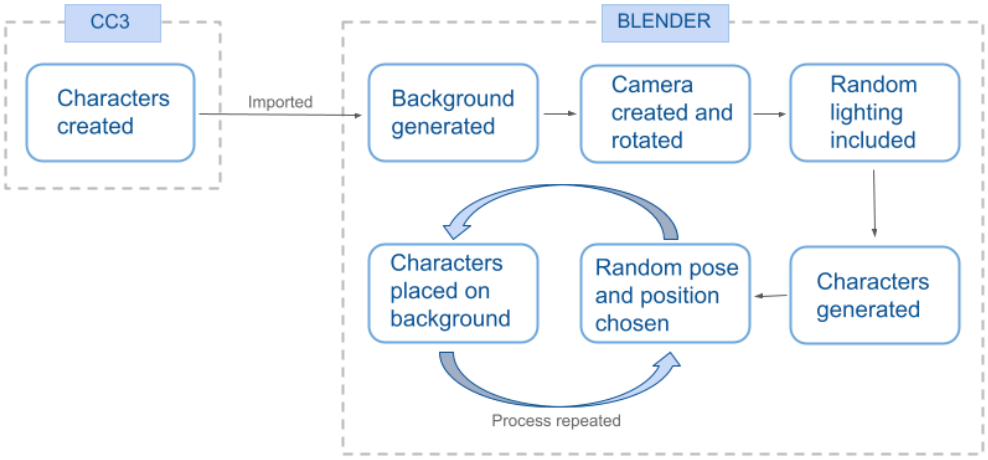


Figure 3.3: The different steps of the rendering process. The characters are created in CC3, while the rendering of the images uses Blender. The last two actions are repeated a set number of times in each rendering process.

Figure 3.4: A selection of characters from the character library. The characters have a wide variety of clothes, skin color, ages, and hair styles.

Pose

A pose was randomly picked from a library for each character when the characters were placed in the background. The library consisted of 116 different animations; 86 were movements in an upright position and were used in this thesis work. The average length of the animations in the upright position was about 473 frames. This resulted in a variation of about 40 700 poses the characters could take.

3.4.0.2 Background

A set of seven different background environments were collected using a fisheye camera mounted on top of a tripod. The frames were collected at a construction site to match some of the most relevant target environments. Three of the seven background frames were taken outdoors, while four were taken indoors. The camera's height was noted to be able to set the same height for the virtual scene

camera, allowing the characters to be placed on the background mesh.

To restrict the area where the imported characters should stand, the floor on the images was marked in a custom-built tool. The floor was marked by pixel coordinates and inserted in a metafile containing all the backgrounds and their corresponding Region of Interest (ROI). The ROI was noted as the corner pixel coordinates of the polygon and then retrieved when spawning the characters. The characters were only allowed to spawn inside the ROI. The ROI ensured the characters did not stand on a wall, in the air, or in other areas that were not allowed. An example of an ROI in one of the used backgrounds can be seen in Figure 3.5.

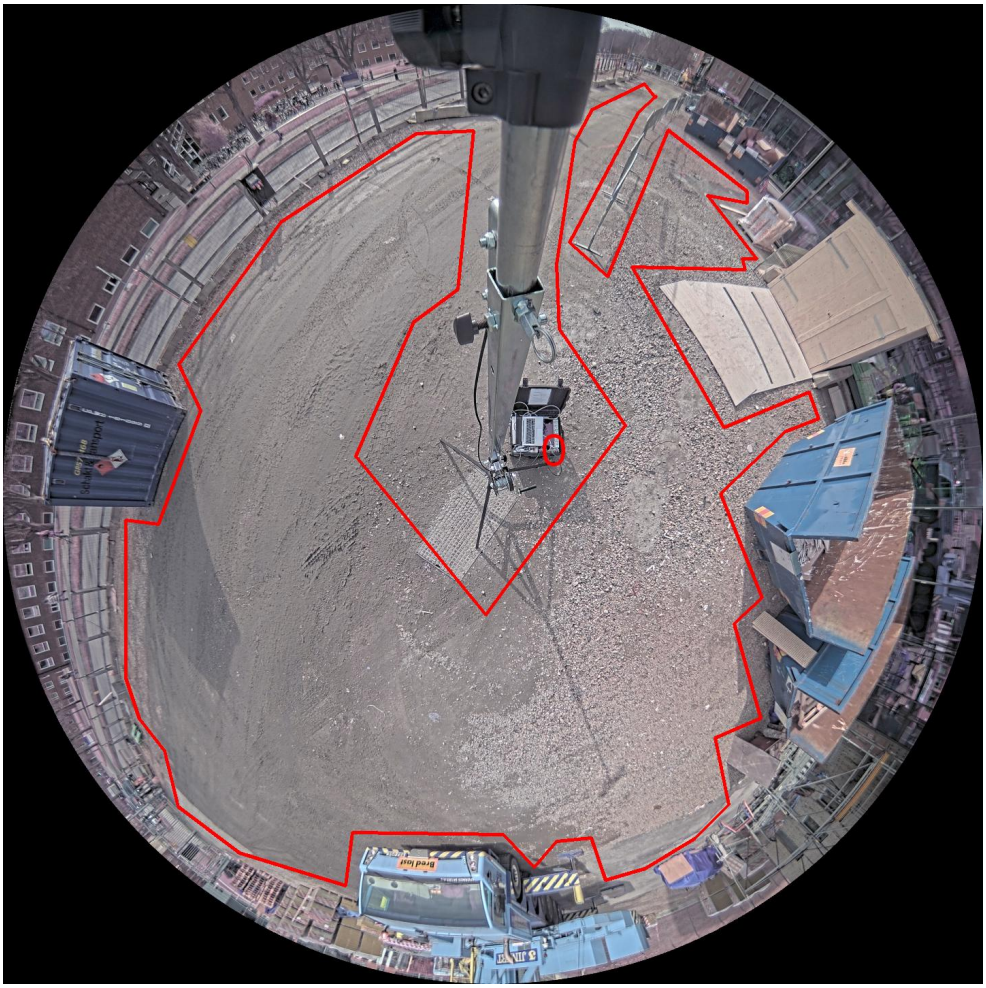


Figure 3.5: Visual example of ROI in a background frame. Each corner coordinate of the drawn figure is saved in a file and later used when placing the characters on the ground, ensuring they do not stand on the fence or up in the air.

3.4.0.3 Lighting

A library of different dome lights was used to vary the rendered frames further and make the data more realistic. Ray tracing was used, as described in the background section 3.4.0.2, to handle the angles at which each light ray entered each pixel. This resulted in frames with different light settings

and provided the characters with appropriate shadows. The dome light was selected randomly to ensure the variety of the data, and the light used varied from dusk to dawn.

3.4.0.4 Camera

The sections below will comment on the different camera settings and adaptations to create the wanted synthetic data.

Camera settings

The camera lens was set to the desired image size, and the focal length was fixed to $f = 1.8$ (compare Section 2.3.2). The focal length was found by experimenting with the settings in Blender. The desired focal length should allow multiple characters to spawn inside the camera view, therefore not being too high, resulting in a very narrow view. On the other hand, it should not be set to low, giving less space for the camera to randomize its view.

Even though the background images were taken with a fisheye camera, the Blender API allows changes to the virtual camera in the scenario to change the settings to behave as a perspective camera. Since the real data is taken with perspective cameras, the decision was to match them using the same view in the synthetic imagery. Since the camera object is responsible for where the background is placed, modifying the original camera object resulted in distorted scenes and the ROI of the entire background image not mapped correctly. Instead, a new camera lens was used only for retrieving the camera view and was created on top of the already existing one, updated with the correct camera matrix, separating their responsibilities. These will be referred to as background camera and perspective camera further in the thesis.

While the focal length of the camera lens was fixed, the tilt and pan were randomized to generate images with different perspectives of the characters. The pan range, corresponding to a rotation around the z-axis (γ in (2.3)), was randomized between 0 and 360 degrees. This allowed scenes to vary between each image. The tilt range, corresponding to a rotation around the x-axis (α in (2.3)), was set from 35 to 45 degrees to get different views around the image. The range restrictions were set to avoid parts where the tripod took up much of the view. As it can be seen in Figure 3.5, the tripod occupies a large part of the ROI if the camera is tilted down too much. On the other hand, if the tilt was too high, the camera got parts of the background mesh not covered with the image, pointing the camera up in the sky.

Then, the intrinsic parameters, the focal length, and center coordinates were put into the virtual camera. The translation was set to $t = [0001]$. The rotational matrix was retrieved from the new tilt α and pan γ and calculated using (2.3). Since the camera did not rotate around the y-axis, β was set to 0. Since the matrix multiplication is non-commutative, the order of the multiplications is important and needs to be matched to how Blender performs its camera operations. The rotations were done in XYZ-order, as in Section 2.3.2:

$$
\begin{aligned}
R &= R_z(\gamma)R_y(0)R_x(\alpha) = R_z(\gamma)\mathrm{I}R_x(\alpha) \\
&= \begin{bmatrix} \cos\gamma & -\sin\gamma & 0 \\ \sin\gamma & \cos\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha \\ 0 & \sin\alpha & \cos\alpha \end{bmatrix} \\
&= \begin{bmatrix} \cos\gamma & -\sin\gamma\cos\alpha & \sin\gamma\sin\alpha \\ \sin\gamma & \cos\alpha\cos\gamma & -\sin\alpha\cos\gamma \\ 0 & \sin\alpha & \cos\gamma \end{bmatrix}
\end{aligned}
$$

The height of the camera was given from the background metadata. As explained in the next section, the height was later used when converting the 2D coordinates to 3D coordinates and vice versa.

ROI Adaptation

Initially, the characters could be placed out of frame since the perspective camera's view was not covering the entire background. This led to empty images in some iterations. The background ROI:s expressed in pixel coordinates (note that the coordinates are for the entire image of the background, see Figure 2.4) were translated in the background camera to 3D coordinates using (2.7). This is how the characters could appear on the ground visually, but they were not restricted to the camera's view. The ROI needed to be limited to the view of the perspective camera. Using H, found in the same manner as in Section 2.3.2 with the found R, leading to the transformed 3D coordinates p_w. Since the center of projection followed the common convention of residing at the origin $(0, 0, 0)$, the camera height z is expressed as a negative distance to the plane.

The 4 pixel coordinates of the perspective camera view $[(0, 0), (1024, 0), (0, 576), (1024, 576)]$ were converted to 3D world coordinates by (2.7) in the same manner. This gave the full view of the camera, including positions not covered by the ROI.

Furthermore, two arrays of each ROI were translated into 3D polygons. These two polygons were then intersected, returning the ROI for where the characters can stand inside the camera view but still on the ground. The intersection of the polygons resulted in a new polygon where the characters appear in the rendered image. The polygon corner coordinates are expressed in 3D coordinates and could be used directly to place the characters, i.e., representing the updated ROI. A visualization of the process can be seen in Figure 3.6.

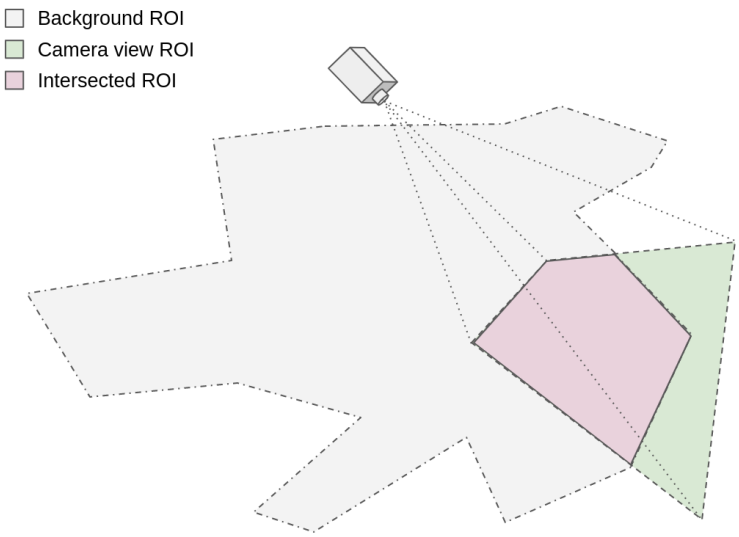


Figure 3.6: A visualization of how the ROI was retrieved where the characters were in the image and still on the ground. Both the background ROI and the camera ROI were retrieved. Then, an intersection was performed to calculate the new ROI represented by a pink polygon in the image.

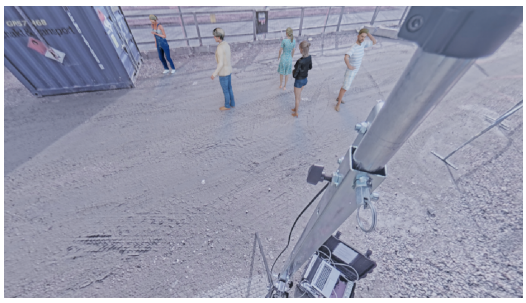
The updated ROI did not account for the character's height, resulting in scenarios where the characters were placed on the far edge of the view and therefore only had parts of their bodies inside the image. The methods for getting the ROI were later on updated. A smaller ROI could be returned by extracting another ROI from the camera where the height was set to two meters higher than the original placement, i.e., merely project the camera view on a plane two meters higher than the ground using (2.4). Another intersection was then performed between the camera view and the modified height ROI. This resulted in a limited ROI in the center of the image. The characters of a height below two meters would be placed entirely within the image.

3.4.1 Annotation

One of the benefits of using synthetic data, as mentioned before, is the ability to automate the annotation process. Since thousands of images may be required for training and manual annotation can be time-consuming, there was a need for an automatic generation of bounding boxes for the heads of the placed characters.

3.4.1.1 Automatic Annotation Process

The initial rendering process returned the image, segmentation information in a `.npy`-file, as well as a `.json`-file with annotations about the characters showing in the image. The annotations included the name, segmentation id, and bounding box for each character. Unfortunately, the bounding boxes were for the entire character using the information in the segmentation file, thus requiring some manual work to return bounding boxes for the heads alone.



(a) The rendered image with five characters.

(b) The segmentation of the image in Figure 3.7a. One of the character's segmentation is hard to see since it is a similar shade of gray as the background.

Figure 3.7: The rendered image and the segmentation view of a the same image. The segmentation is used to retrieve the head masks for the characters.

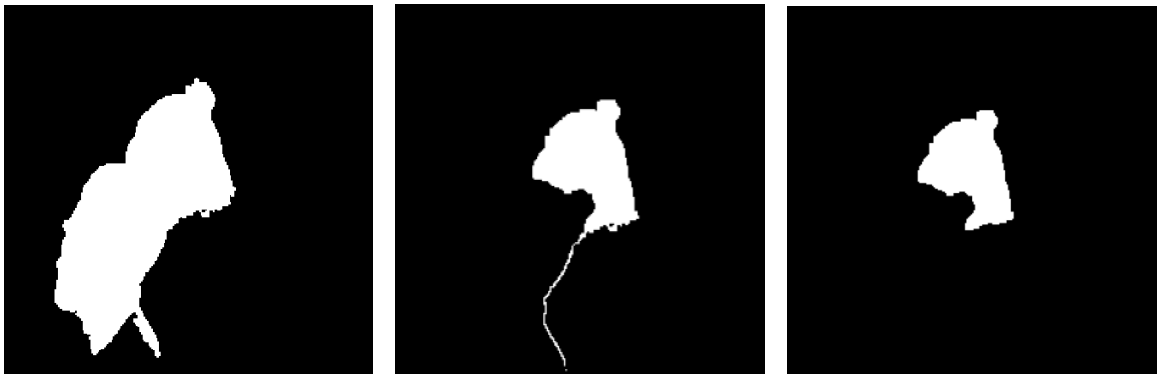
An additional file was generated for each image, a so-called head mask image. Since each vertex, intersections of points (edges) on the 3D model surface, in the 3D model belongs to a vertex group, the vertices in the character that belonged to the head group were retrieved. The vertices contain information about their location in the 3D space and could be transformed into pixel coordinates in the same manner as the ROI in the previous section. Then, the vertices were painted white, and a binary image was created representing the head mask (Figure 3.8a).

The segmentation file and the head mask were element-wise multiplied to retrieve a separate mask for each character. The segmentation file is a grayscale image. The head mask is a binary image. The element-wise multiplication resulted in pixels with a value separated from 0 for every pixel that belongs to the individual head mask. The head mask was later used to retrieve the bounding boxes for each character at the end of the rendering process. The unique segmentation ids could be mapped to each character, adding a bounding box for each head in the `.json`-file.

3.4.1.2 Morphological Opening of Masks

There were some issues when the head masks of two characters overlapped. Figure 3.8a presents the binary head mask for the generated image in Figure 3.8a where the head masks contained some overlapping. When generating each separate head mask, the boundary regions got irregularities as in Figure 3.8b. The idea was to use the morphological opening on the intersected binary masks to flatten them out, leaving fewer surfaces to falsely correspond to the mask and removing single pixels

located outside of the mask. Using erosion to shrink the image regions followed by dilation with the same kernel size, the idea was to flatten the regions out and result in more correct bounding boxes. An example of the resulting head mask from where the bounding box is extracted is presented in Figure 3.8c. The kernel used was square-shaped, while the size of the kernel was experimented with to find the best result. The kernel sizes used were 2×2 and 4×4. The used kernel is provided in the description for each of the created datasets.



(a) Original head mask image, where all heads are in the same file.

(b) Head mask for the character in the back that has been filtered out using the segmentation image. There are some floating pixels around the contour of the head mask from the character in the front.

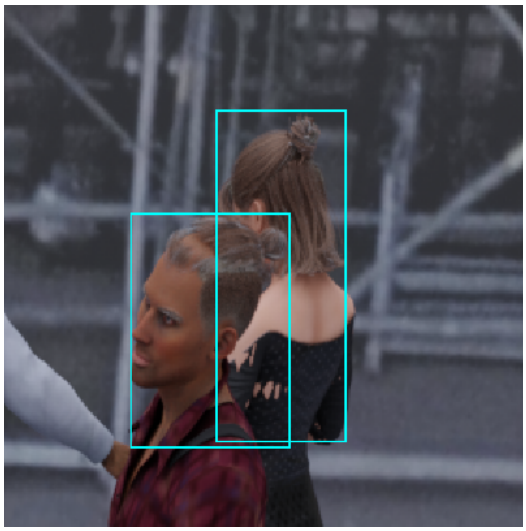
(c) Head mask after the morphological opening of the head mask, resulting in a head mask without the floating pixels.

Figure 3.8: Steps of eroding and dilating the head masks. The head mask for the character is filtered out using the segmentation information, and then a morphological opening is performed to receive a head mask without floating pixels.

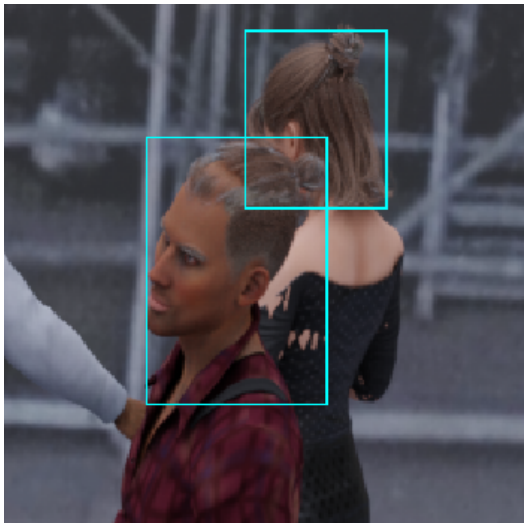
3.4.2 Rendering Process

For each iteration of the rendering process, a random number of characters within a given interval were generated. They were spawned in a random pose within the resulting ROI, as described in Section 3.4.0.2 above and Figure 3.3. The same generated characters were then placed in different positions within the same background a fixed number of times with new poses for the characters. The new pose is from the same animation but at another timestamp. This sped up the process significantly since a noticeable amount of time was spent generating the characters and the background. An example of the described process can be seen in Figure 3.10.

Two separate machines were used to render the images. One of them used its own GPU, whereas the other was connected to a cluster of GPUs. The average images per hour on these machines combined were around 174 images per hour, whereas the computer not connected to the cluster had an average of 42 images per hour.



(a) Bounding boxes before the morphological opening. The floating pixels resulted in a bounding box that was too big for the character in the back.



(b) Bounding boxes after the morphological opening, resulting in a more accurate bounding box for the character in the back.

Figure 3.9: Bounding boxes before and after the morphological opening.

Figure 3.10: Example of the same characters randomly placed within the acceptable ROI of the certain background. The selected characters is placed in the image with a pose. New images in the same iteration are generated by giving the characters new positions and a new pose, before a new iteration with another selection of characters, background, and camera angle is randomized.

Chapter 4

Results

This chapter presents the results of the initial model evaluation. Then, after presenting the targeted improvement, the generated synthetic datasets are described, and the models trained on the datasets are evaluated. The best-performing model is then selected for a detailed evaluation with prediction distribution and confident FPs to see the impact of the dataset on the targeted improvement. Then, further investigations are done by comparing the ratios between the datasets and training with a frozen backbone to see what effects it has on the result.

4.1 Initial Model Evaluation

The initial model was evaluated using the COCO evaluation metrics, which produced the AP. The F_1 score, recall, and precision were also returned for the optimal threshold. The evaluation was done on the Roboflow test set and the Summerwork dataset. The results are presented in Table 4.1. The model performed significantly better on the test set from Roboflow with an AP and F_1-score, evaluated for the best threshold, over 0.9 for both classes. The gap between the AP of the classes was larger when looking at the Summerwork dataset. The Head class dropped by almost 50% and Hard hat by 14% compared to the Roboflow dataset. The Precision-Recall curve for each class on the Summerwork dataset in Figure 4.1 shows the low recall value for the Head class.

The size of the ground truth annotations was also examined to find potential correlations. The amount of TP and FN for each ground truth pixel size is presented in Table 4.3.

	Initial model							
	Roboflow				**Summerwork**			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.958	0.947	0.955	0.940	0.828	0.825	0.829	0.820
Head	0.936	0.928	0.936	0.920	0.502	0.608	0.859	0.470

Table 4.1: Initial model evaluation on Roboflow and Summerwork with an IoU threshold of 0.5.

4.1.1 TIDE Results

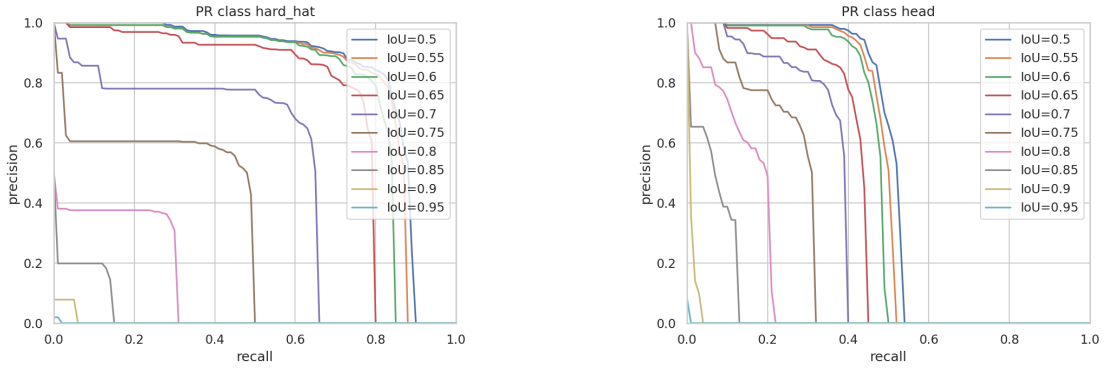
It can be seen in the TIDE results, in Figure 4.2a, that the prediction of the Roboflow dataset using the intial model resulted in few detection errors. The few errors lie within the following categories; cls, loc, bkg and miss. In Figure 4.2b, it can be seen that predicting on the Summerwork dataset resulted in a higher number of FPs and FNs, where FNs dominated. The largest categories here were misclassified objects and missed objects.

Class	Trait	Ground truths	Predictions	TP	FN	FP
Head	Light hair	215	158	91	124	66
	Dark hair	80	54	47	33	8
	Other hat	78	80	29	49	51
	Total	373	292	167	206	125
Hard hat	Total	482	600	415	67	185

Table 4.2: Model prediction distribution on the Summerwork dataset based on class traits using Fiftyone. The confidence threshold is set to 0.4.

Size	Ground truths	TP	FN
Small	100	16	84
Medium	371	225	146
Large	384	341	43

Table 4.3: Model prediction distribution on the **Summerwork** dataset based on ground truth pixel size using Fiftyone. The confidence threshold is set to 0.4.



(a) Precision-Recall curve for class Hard hat

(b) Precision-Recall curve for class head

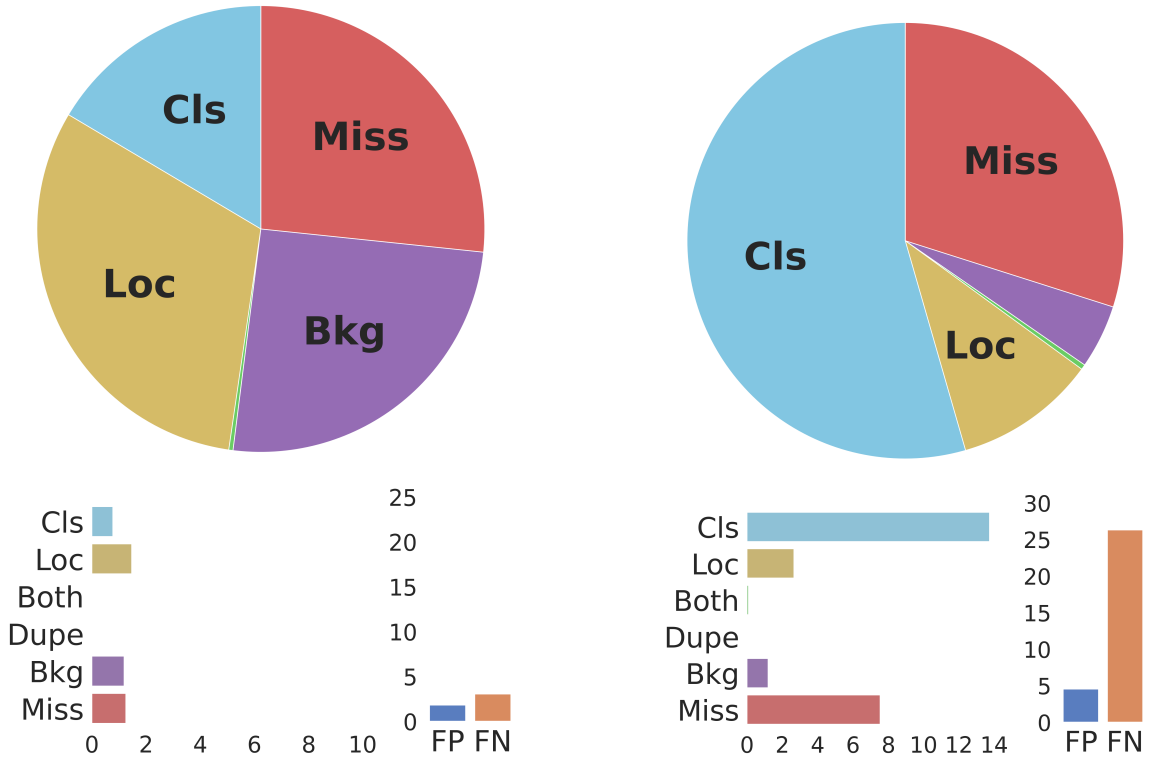
Figure 4.1: Precision-Recall curves for different IoUs for the two respective classes and the Summerwork dataset.

4.1.2 Detailed Evaluation

A more detailed evaluation was done against the Summerwork dataset due to its weak performance in comparison to the Roboflow dataset. When analyzing the model's weaknesses, FiftyOne was used to visualize the predictions done against GT, as described in Section 3.3.2.2. The summarized numbers of TP, FN, and FP are presented in Table 4.2 against each class and trait explored, and in Table 4.3 against the ground truth annotation size and an IoU threshold 0.5. It can be seen that the model correctly classified 86% of the Hard hat class and 41% of the Head class. The Head class had a lot more FN than the Hard hat class, with corresponding high FP in the Hard hat class.

4.1.2.1 Confident FP

The FPs were evaluated further against the confidence scores. Out of the 401 images 63 contained confident FP, i.e., the confident score was higher than 0.9 and it was a FP. It resulted in a total of 69 confident FPs. The respective average confidence score for each class is presented in Table 4.4.



(a) TIDE results on the Roboflow dataset.

(b) TIDE results on the Summerwork dataset.

Figure 4.2: TIDE results for the initial model on Roboflow and Summerwork datasets.

The annotations were analyzed to see what type of traits the ground truth had to see where it had weaknesses in its predictions.

Class	**Trait**	**No. of confident FP**	**Avg. confidence score**
Head	Light hair	20	0.961
	Dark hair	2	0.935
	Other hat	37	0.972
	Total	22	0.958
Background	Objects	5	0.942

Table 4.4: Number of confident False Positives sorted on category. Total confidence score is the sum of all Confidence score for each FP with each trait.

The confident FPs were situations where class head was classified as Hard hat as well as background objects being picked up by the model. The FPs could be split into three sub-categories: Head without a hat, Head with a hat, and background detections that do not have a corresponding class. The average confidence score for the predictions for all three sub-categories was high. Heads without hats could be split into two sub-categories - light or dark hair.

4.1.3 Identifying Model Weaknesses

To be able to proceed with a weakness to target during training the results of the initial model needed to be analysed in this stage. There were several areas for which the targeted improvement could be chosen. First, the AP on the Summerwork dataset indicated a worse result for the Head class than Hard hat. Specifically, the recall was very low. This was also seen in the TIDE results, where the FN dominated.

Second, the detailed evaluation found that two traits in the Head class had many FN: People with light hair and people with other hats. In addition, the Hard hat class had a lot of FP. This would imply that the model was more likely to misclassify a Head as a Hard hat than the other way around. Another interesting aspect was the ground truth annotation size and its effect on predictions. The amount of FN in proportion to the number of ground truths decreased as the pixel size got bigger.

When analyzing the confident FP, the same pattern followed. Again, light hair and other hats were the traits of the Head class where the model was most confident in its FPs.

The targeted improvement was set to the Head class with the trait of light hair. Since hair was easy to vary and modify the existing characters by changing their hair color, it was found a plausible improvement that could be made in the time frame without demanding too much manual work.

4.2 Generation of Synthetic Data

The results of the image rendering process are presented in this section. The total number of rendered images is just below 50 000. More detailed information about the rendered data is presented in the sections below.

4.2.1 Synthetic Datasets

The synthetic data rendered during this thesis work was divided into different datasets. The datasets are presented below, and the rendering process for each dataset is explained. Each dataset was given a capital letter to simplify the differentiation. In Table 4.5, the number of frames and annotations in each dataset are summarized.

Dataset	No. of images	No. of head annotations	No. of Unique Characters
Synth-A	9 820	21 098	80
Synth-B	9 820	42 875	100
Synth-C	19 640	64 970	100
Synth-D	9 820	42 649	250
Synth-E	19 640	86 263	250
Synth-F	39 280	151 233	250
Synth-G	47 370	186 444	250

Table 4.5: The generated synthetic datasets with the amount of images and annotations that they contain.

4.2.1.1 Synthetic A

Synthetic dataset A (Synth-A) contains characters with light hair. Each frame contains between 3 to 10 randomly selected characters placed randomly in the scene. This implies that a character risks being occluded, with only parts of its body within the camera's field of view. The characters

were selected from a library containing approximately 80 different characters with light hair. The morphological opening was done with a kernel size of 2x2 pixels within this dataset.

4.2.1.2 Synthetic B

Synthetic dataset B (Synth-B) consists of characters with light hair, similar to the Synth-A dataset. However, the characters in this dataset were restricted to be placed entirely within the camera's field of view. Each frame in this dataset contains 3 to 6 characters, randomly selected from a library of about 100 characters. The morphological opening was done with a kernel of size 4×4 pixels in the Synth-B dataset.

4.2.1.3 Synthetic C

Synthetic dataset C (Synth-C) is a dataset where all data from Synth-A and Synth-B were combined.

4.2.1.4 Synthetic D

Synthetic dataset D (Synth-D) contains characters with all hair colors, including light, brown, and dark hair. The dataset is created similarly to the Synth-B dataset. The only difference between the two datasets is that this dataset uses a more extensive character library of approximately 250 characters.

4.2.1.5 Synthetic E

Synthetic dataset E (Synth-E) is a dataset that, just like Synth-C, contains characters with all hair colors. In addition, the dataset is generated like Synth-D but contains double the amount of images.

4.2.1.6 Synthetic F

The Synthetic dataset F (Synth-F) is a collection of all images in the datasets Synth-B and Synth-E.

4.2.1.7 Synthetic G

The Synthetic dataset G (Synth-G) was created by combining all rendered data. This means that Synth-G combines Synth-C, Synth-E, and additional data, rendered as Synth-D.

4.3 Retraining and Evaluation

Seven models were trained on the Roboflow dataset, together with one of the generated synthetic datasets. The ratio between the real and the synthetic dataset for these models was set to (1:0.5), see Section 3.3.3.1 for furter description. The models and what datasets they were trained on are presented in Table 4.6.

A summarized evaluation of all of the new models can be seen in Tables 4.7-4.11. The tables include the results from evaluating on the test set of the real-data training dataset, Roboflow, and the dataset Summerwork. Both datasets are further described above, in Sections 3.2.1 and 3.2.2. Each table presents the AP for each class and dataset, as well as mAP, F_1-score, precision, and recall.

The tables show that for almost all the models, the addition of synthetic data improved the AP on the Summerwork dataset. The increase is overall higher for the Head class. However, when evaluating Model-B, Model-D, and Model-G against the Summerwork dataset, the AP for the Hard hat decreased by an average of 19.67 compared to the initial model. The results of the Roboflow dataset remained somewhat stable for all the models. The model with the highest AP and F_1-score

Model	Training Dataset
Model-A	Roboflow and Synth-A
Model-B	Roboflow and Synth-B
Model-C	Roboflow and Synth-C
Model-D	Roboflow and Synth-D
Model-E	Roboflow and Synth-E
Model-F	Roboflow and Synth-F
Model-G	Roboflow and Synth-G

Table 4.6: Models and the training sets used to create them. Every model was trained with a ratio of (1:0.5).

when evaluating on the Summerwork dataset was Model-C. Regarding the evaluation against the Roboflow test set, Model-C provides the highest APs, but not F_1-scores.

	Model-A							
	Roboflow				**Summerwork**			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.961 (+0.003)	0.945	0.951	0.940	0.844 (+0.015)	0.842	0.876	0.810
Head	0.939 (+0.003)	0.923	0.926	0.920	0.717 (+0.109)	0.762	0.850	0.690

Table 4.7: Model-A evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

	Model-B							
	Roboflow				**Summerwork**			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.958 (+0.000)	0.942	0.955	0.930	0.810 (−0.018)	0.815	0.866	0.770
Head	0.942 (+0.006)	0.922	0.935	0.910	0.658 (+0.156)	0.713	0.820	0.630

Table 4.8: Model-B evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

	Model-C							
	Roboflow				**Summerwork**			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.959 (+0.001)	0.942	0.944	0.940	0.853 (+0.025)	0.834	0.849	0.820
Head	0.940 (−0.004)	0.924	0.927	0.920	0.781 (+0.279)	0.811	0.897	0.740

Table 4.9: Model-C evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

4.3.1 Evaluating the Best Model

As mentioned above, Model-C overall performed best when evaluating on the Summerwork dataset. This section will present a more detailed evaluation of Model-C in the same manner as has been done with the initial model.

	Model-D							
	Roboflow				Summerwork			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.956 (−0.002)	0.945	0.960	0.930	0.806 (−0.022)	0.807	0.849	0.770
Head	0.934 (−0.002)	0.924	0.938	0.910	0.712 (+0.210)	0.777	0.844	0.720

Table 4.10: Model-D evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

	Model-E							
	Roboflow				Summerwork			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.960 (+0.002)	0.946	0.962	0.930	0.846 (+0.018)	0.821	0.892	0.760
Head	0.939 (+0.003)	0.922	0.945	0.900	0.741 (+0.239)	0.782	0.886	0.700

Table 4.11: Model-E evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

	Model-F							
	Roboflow				Summerwork			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.951 (−0.007)	0.942	0.954	0.930	0.840 (+0.012)	0.836	0.864	0.810
Head	0.938 (+0.002)	0.924	0.928	0.920	0.669 (+0.167)	0.758	0.908	0.650

Table 4.12: Model-F evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

	Model-G							
	Roboflow				Summerwork			
Class	**AP**	**F_1**	**P**	**R**	**AP**	**F_1**	**P**	**R**
Hard hat	0.958 (+0.000)	0.945	0.959	0.930	0.809 (−0.019)	0.807	0.861	0.750
Head	0.936 (+0.000)	0.924	0.927	0.920	0.759 (+0.257)	0.805	0.868	0.760

Table 4.13: Model-G evaluation on Roboflow and Summerwork with an IoU threshold of 0.5. AP in parenthesis is the change from the initial model.

In Figure 4.3, the PR-curves for the two respective classes for Model-C on the Summerwork dataset are presented. There has been a significant increase for the IoU thresholds compared to the initial model in Figure 4.1.

The TIDE-results in Figure 4.4 shows a stable decrease in misclassifications and misses of Model-C in comparison to the initial model when studying the results on the Summerwork dataset.

4.3.1.1 Detailed Evaluation

The detailed evaluation of Model-C can be seen in Table 4.14 and 4.15. The model's predictions, TP, FP, and FN are presented against class traits and bounding box size. Model-C showed an increase of 51 TP on class Head with light hair, and a total of 87 more FP in the Head class. The number of predictions for Hard hat decreased significantly, whereas the predictions for Head class increased. When studying the predictions grouped on bounding box size, there has been a slight increase for every size, with the most significant improvement in medium-sized bounding boxes.

Class	Trait	Ground truths	Predictions	TP	FN	FP
Head	Light hair	215	173	143	72	30
	Dark hair	80	59	51	29	8
	Other hats	78	83	60	18	23
	Total	373	315	254	119	61
Hard hat	Total	482	424	409	73	15

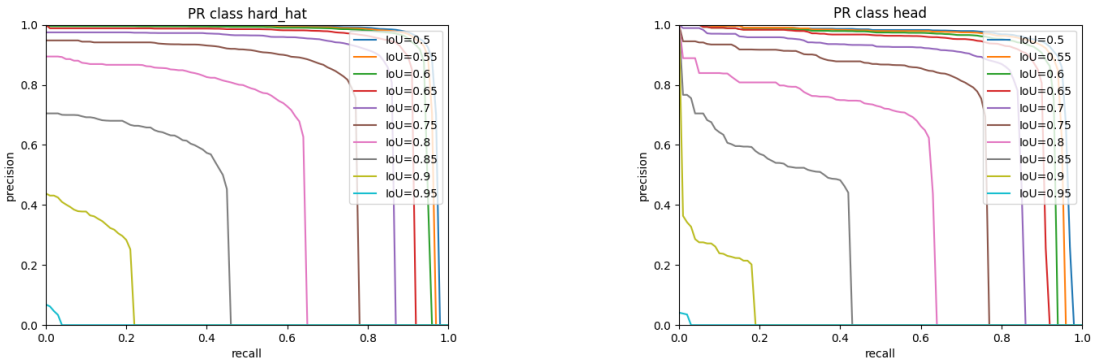
Table 4.14: Model-C prediction distribution on the Summerwork dataset based on class traits using Fiftyone. The confidence threshold is set to 0.4.

Size	Ground truths	TP	FN
Small	100	21	79
Medium	371	289	82
Large	384	353	31

Table 4.15: Model-C prediction distribution on the Summerwork dataset based on ground truth pixel size using Fiftyone. The confidence threshold is set to 0.4.

4.3.1.2 TIDE results

The TIDE evaluation performed on Model-C can be seen in Figure 4.4. Looking into the evaluation on the Roboflow dataset, it can be seen that the results are very similar to the evaluation from the initial model. The only prediction error categories affected are Bkg and Miss, which both decreased slightly with Model-C. However, when it comes to the Summerwork dataset it is clear that the Cls category decreased significantly. Both the Loc and Miss categories decreased by half, while the rest of the categories remained stable.



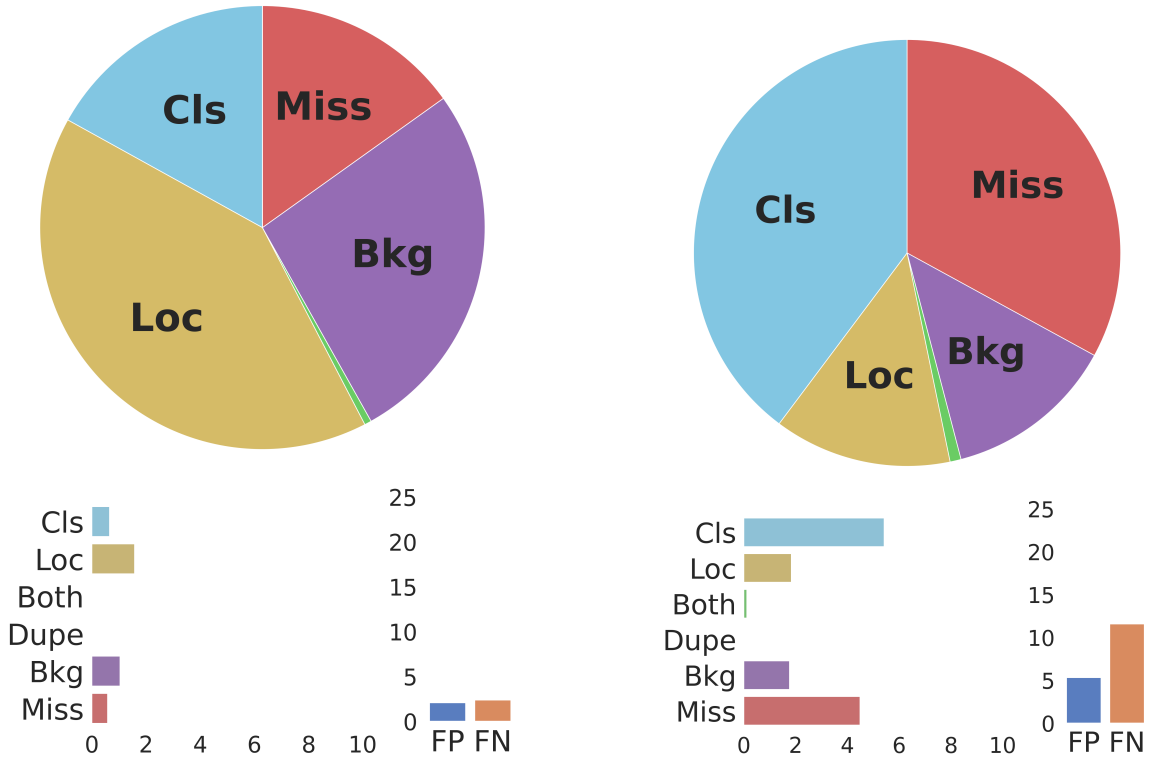
(a) Precision-Recall curve for class Hard hat.

(b) Precision-Recall curve for class head.

Figure 4.3: Precision-Recall curves for the two respective classes for Model-C and the Summerwork dataset.

4.3.1.3 Confident FP

The confident FP was again evaluated and compared against the inital model's. There has been a significant decrease of confident FP in comparison to the initial model. The total confident FP for



(a) TIDE results on the Roboflow dataset.

(b) TIDE results on the Summerwork dataset.

Figure 4.4: TIDE results for Model-C on Roboflow and Summerwork datasets.

class Head without hats decreased by 16, as well as 29 less for class Head with other hats. Even the background confident FP decreased, and no confident FP of class Hard hat was found.

4.3.2 Further Experiments

After evaluating the best performing model, some further investigations were conducted. Different ratios between the datasets were evaluated to see the impact on the resulting model. The model was also re-trained with a frozen backbone to see if the performance changed. Furthermore, an additional dataset was introduced to validate the former results.

4.3.2.1 Training with Different Ratios

The results from training with different ratios can be seen in Table 4.17, where all training sessions are run with Model-C. As seen in the table, the mAP for the Summerwork test set starts improving already at ratio 1:0.25. In contrast, the mAP for the Robolow remains in around the same as the original model, trained entirely on non-synthetic data. The best mAP results were obtained when the ratio was set to 1:0.5, where the prediction results for both the Roboflow and Summerwork test sets improved from the initial model. The improved results can also be seen when the ratio is 0.25:1. However, from a ratio 0.75:1 the mAP for Summerwork decreases significantly.

Class	Trait	No. of confident FP	Avg. confidence score
Head	Light hair	3 (−17)	0.960 (−0.005)
	Dark hair	3 (+1)	0.943 (+0.008)
	Total	6 (−16)	0.952 (−0.006)
Head	Other hat	8 (−29)	0.958 (−0.014)
Background	Objects	1 (−4)	0.940 (−0.002)

Table 4.16: Number of confident False Positives sorted on class for Model-C. The values in parenthesis are the comparison to the initial model.

4.3.2.2 Training with Frozen Layers

An experiment with frozen layers was performed to investigate whether it would improve the model performance. The training resulted in a model with worse performance than the other models, trained with unfrozen layers. However, training with additional synthetic data, compared with only real data from Roboflow, still resulted in an improved model, when evaluated on the Summerwork dataset. When evaluating on the test split of the Roboflow dataset no improvement could be seen.

4.3.2.3 Dubai Evaluations

The Dubai dataset was introduced later in the process to secure former results further and detect potential side effects. The evaluation was done for the initial model and the Model-C, the synthetic model that performed the best on the other test sets. In Table 4.19, the AP results are presented. The evaluation shows that Model-C provides an improved mAP compared to the initial model, also when it comes to the Dubai dataset.

An additional detailed evaluation for the initial and Model-C on Dubai was performed. The results show an increase of 11 TP for light and 9 for dark hair. The amount of FN for class Head with light hair decreased by 8 and FP increased by 1. Dark hair got 18 less FN and 5 less FP. The results also showed an overall higher prediction rate on class Head for Model-C.

The results showed, just like with the former test sets, that the new model, trained with synthetic data, provided better performance.

		Roboflow		Summerwork	
Ratio	**Class**	**AP**	**F1**	**AP**	**F1**
1:0	Hard hat	0.958	0.947	0.828	0.825
	Head	0.936	0.928	0.502	0.608
	Average	0.947	-	0.665	-
1:0.25	Hard hat	0.956	0.943	0.782	0.791
	Head	0.935	0.924	0.761	0.770
	Average	0.946	-	0.771	-
1:0.5	Hard hat	0.959	0.942	0.853	0.834
	Head	0.940	0.924	0.781	0.811
	Average	0.950	-	0.817	-
1:0.75	Hard hat	0.959	0.943	0.851	0.830
	Head	0.939	0.923	0.696	0.760
	Average	0.949	-	0.774	-
1:1	Hard hat	0.958	0.945	0.838	0.829
	Head	0.941	0.925	0.723	0.770
	Average	0.950	-	0.780	-
0.75:1	Hard hat	0.961	0.944	0.791	0.793
	Head	0.939	0.921	0.768	0.780
	Average	0.950	-	0.779	-
0.5:1	Hard hat	0.959	0.944	0.819	0.824
	Head	0.936	0.922	0.659	0.757
	Average	0.948	-	0.739	-
0.25:1	Hard hat	0.958	0.941	0.789	0.790
	Head	0.939	0.923	0.706	0.765
	Average	0.949	-	0.748	-

Table 4.17: Model-C evaluation results for different ratios.

Test dataset	**Model**	**AP (Head)**	**AP (Hard hat)**	**mAP**
Roboflow	Frozen Initial	0.927	0.950	0.939
	Frozen Synthetic	0.927	0.950	0.938
Summerwork	Frozen Initial	0.352	0.791	0.571
	Frozen Synthetic	0.668	0.772	0.720

Table 4.18: Comparison in AP of Model-C and the initial model with a frozen and unfrozen backbone.

Model	**AP (Head)**	**AP (Hard hat)**	**mAP**
Initial model	0.667	0.675	0.671
Model-C	0.722	0.718	0.720

Table 4.19: AP of the Initial model and the best performed model, Model-C, on the Dubai dataset.

Chapter 5

Discussion

This chapter discusses the results and conclusions drawn from the research questions. Subjects discussed are, for example, whether the targeted improvement was successful, what could have affected the results, and if any side effects could be seen. In addition, the results from the investigation with changing the ratio are discussed, and whether training on a frozen or unfrozen backbone is preferable.

5.1 Targeted Improvement

The results showed that the trait chosen to target indeed improved with synthetic training data. It was observed that TP for the best model increased by about 55%, while the amount of FP decreased by about 55% for class Head with light hair (Table 4.14), indicating that the targeted improvement was successful. Furthermore, the TIDE results indicated the same thing; Figure 4.4b shows that there are fewer misclassifications and misses than in the same figure for the initial model (Figure 4.2b). However, it can be discussed whether the improvement was directly correlated to the targeted trait since the model's performance in the Head class improved overall. One of the reasons for questioning the target results is that the Summerwork dataset content is limited to a few individuals. Therefore, an improvement in detecting one individual may affect the result in multiple occurrences.

By observing the results from the best-performed model trained on mixed hair colors, Model-E, one can see that Model-C outperforms Model-E in AP and F_1 for both classes. Therefore, the result indicates that the targeted improvement was reached since the dataset with more light-haired characters gave the best results.

Furthermore, the result also indicated that the performance related to bounding box size improved. This improvement was especially clear for the bounding boxes of size medium. These results indicate that the overall performance increase does not solely depend on the targeted trait of light hair. It is plausible that the better result on people with light hair was an effect of the characters often being given a medium-sized annotation due to where they were placed in the image.

On the other hand, an improvement in detection of the Head class with light hair could also be seen in the results for the Dubai dataset, indicating the target improvement. This improvement could be seen by the increased amount of TP for both light and dark hair, whereas the FP decreased by 8 for light hair. It was also an overall improvement on the mAP, see Table 4.19, by 0.05, and a more significant improvement in AP for class Head than Hard hat.

Another result of the model evaluations was the drastic decrease of confident FPs in Model-C compared to the initial model across all classes for the Summerwork dataset. At the same time, as the amount of total FP for dark hair remained the same as it was in the original model, the

confident FPs of light hair went down significantly. This indicates that training the model on a mix of synthetic and real data leads to better weights and biases.

5.2 Ratio

As mentioned in Section 4.3.2.1, the best results were obtained when the ratio was set to 1:0.5, meaning double the chance to pick an image from the Roboflow dataset than from the synthetic dataset in each iteration. These results indicate that picking fewer synthetic images is preferred, or that using a ratio of 1:0.5 resulted in a balanced dataset. Since Roboflow has 24.7% head annotations, the introduction of the synthetic dataset, with only head annotations, provided a better balance in the class distribution in the training dataset. However, it can also be questioned if introducing examples of the Hard hat class in the synthetic data would impact the results, given that the Roboflow dataset is unbalanced.

5.3 Side effects

The TIDE evaluation shows that the number of misclassified objects decreased significantly with Model-C, while the other detection error categories stayed relatively stable. There were, for example, no multiple new occurrences in the background category, indicating that the model would detect random objects in the image. The results from TIDE can be used to strengthen the fact that there were no unwanted side effects introduced to the model when training with synthetic data. Furthermore, the detailed evaluation using FiftyOne confirmed this belief since the new models did not introduce any side effects seen by the human eye. However, no detailed evaluation of the model's layers has been done to see how introducing synthetic data has affected the feature extraction.

5.4 Optimal Synthetic Data

There were some interesting results regarding the data quality in the respective dataset. Comparing Model-A and Model-B, the first model, Model-A, could contain empty scenes and erroneous bounding boxes gave better AP and F_1. These result were surprising, as the motivation for generating Synth-B and continuing with the same settings for the rest of the datasets that followed was that more annotations of the Head class in the images would yield better results. Synth-B also had a broader variety in appearances of the people used. These facts, combined with improved bounding boxes, seemed plausible for the performance to be better.

It seems like negative examples might have been a reason for the difference in the performance results. Perhaps training on empty images improved the model's understanding of features not to emphasize. The synthetic characters also had a higher chance of being occluded since they could be placed at the edges of the images, only partly showing their heads. This could potentially have resulted in an improvement of occluded occurrences in the test sets.

As the synthetic datasets grew in size, combining the data from previous datasets, the fraction of images from Synth-A decreased. This might be another contributing factor to the promising performance of Model-C, as discussed above. Model-C had a higher probability that the images from Synth-A were used during training than the bigger models. It may be discussed whether the optimal synthetic data was the dataset that introduced more light-haired characters or if occlusion and negative examples were what provided these results.

5.5 Frozen vs. Unfrozen Backbone

The model used a backbone with unfrozen layers, meaning that all pre-trained weights could be adjusted with our training data. As mentioned in the related work, freezing the backbone layers or not is a highly debated subject. Therefore, as a final test, the initial model and the best synthetic model were retrained with frozen backbone layers to see if this could affect the results.

As seen in Table 4.18, training with an unfrozen backbone seemed to give the best results. A comparison can be made between our results and the results from Tremblay et al. [5], where they argued that they had a great variety in their dataset, which could lead to their good results on training on an unfrozen backbone. Interestingly, the models trained in this thesis are done on a combination of real and synthetic data, whereas their models were trained on synthetic and only fine-tuned on real data. Liu and Mildner used the same training process with fine-tuning on real data, which could explain the difference in results between their models and the ones used here [1]. Even though the mix of synthetic and real data was a part of the training dataset in this case, similar results could be seen in Tremblay et al. with a better model when training on an unfrozen backbone [5]. Perhaps the diversity in data that comes from having the mix of real and synthetic could contribute to our results, or the fact that the diversity introduced in the synthetic data was good enough.

Comparing the initial frozen model with the frozen Model-C, an improvement with training on additional synthetic data could still be seen with frozen layers. This proves that synthetic data have the potential to affect the result in a positive manner with both frozen and unfrozen layers, indicating that the synthetic data is highly relevant in multiple training situations.

5.6 Generation of Synthetic Data

Overall, generating and rendering the specific scenes was found effective for creating a large amount of data with a great variety. Using the backgrounds taken with a fisheye-camera, it was possible to create multiple different views. This was done with the help of a randomized tilt and pan in the virtual camera.

The scenes were easy to customize. It was possible to specify what type of characters were supposed to be in the scene. The number of characters in each scene could also be adjusted. Although the implementation process was time-consuming, the generation process did not require much manual work. These facts make the method a good way of adding specific data for unbalanced training datasets.

The morphological opening used to get more precise bounding boxes successfully removed the floating pixels and improved the quality of the bounding boxes. The results showed that different kernel sizes provided different resulting bounding boxes, commented in Section 3.4.1.2. When using a kernel size of 2×2, there were still some cases where the bounding boxes became too big and still left some floating pixels. Using a 4×4 kernel instead eroded the contours of the face, especially on parts such as noses and ears. A different kernel shape might have had a better result than the squared shape, not missing contours of the face when placing the bounding boxes.

The binary head mask images also contain the hair of the characters. For example, this resulted very large bounding boxes for characters with long hair showing. There also existed cases where the character's skin showed through its clothing, making the bounding box go further down the neck than preferable. To end up with correct bounding boxes, the hair and vertices below the neck will need to be removed from the binary head mask.

After studying the results, the impact of the bounding boxes' correctness is still unclear. As seen in Tables 4.7 and 4.8, Model-A had a better performance than Model-B even though Model-B had more precise bounding boxes. This these results indicate that even though tight bounding boxes are generally preferred, it might have had a minor impact on the model's performance due to the

somewhat poor annotation quality of Roboflow.

5.6.1 Optimize Positions of Characters

In this study, the ROI for where the characters should be placed was limited to the camera view. The ROI was then further limited so that the head of the characters were placed within the image (Synth-B in Table 4.6 and below), using a set height that the characters needed to be shorter than two meters. The adaptation that showed the characters' heads was an approach that resulted in a centered, smaller ROI that showed the character's heads every time. This method could be further improved, allowing characters to be somewhat occluded.

5.6.2 Domain Generalization

Since the synthetic data could improve the model's performance, it may be assumed that the synthetic data generated was realistic enough and generalized to the real-world data. The randomized camera angles, scene light, and character poses might have been enough to achieve good domain randomization. Since the same adaptations were done for all models, the importance of the adaptation can not fully be established. However, the results indicate that the domain randomization used in this thesis helped to increase the performance. It would be interesting to do similar evaluations of models trained on solely one of the domain randomizations applied here to see what gives the best performance gain in isolation. Especially if the real background had any impact or if the same results could have been achieved without backgrounds as Lin et al. found when examining action organization [18]. Unfortunately, the time limit did not allow further examinations on this topic.

5.7 Usage of Synthetic Data

When it comes to the usage of Synthetic data, there are, as seen in this thesis, great benefits when creating an object detection model. However, some aspects need to be considered when creating and using synthetic training data. Rendering data is computationally expensive, and while the automatic rendering process avoids the workload of a manual annotation process, the rendering process demands GPU power in order to perform within a reasonable time frame. These aspects need to be considered when deciding to use synthetic data, is the time spent setting up methods for generation worth the outcome? In this case, where the risks involved when filming people without hard hats in an environment that demands safety gear are highly present, the use of a synthetic rendering process is a valuable advantage.

Furthermore, while synthetic data can improve certain weaknesses of a model, the possibility of monitoring the rendering comes with the risk that the resulting data becomes biased. Therefore, it is important to be aware of and work actively to improve the representation in the data. On the other side, if synthetic data can be used to fill the holes in the available real data, there are no excuses for not including certain traits. For example, the excuse that there are not a lot of training footage of light haired-people (or any other trait) does not hold since data can be generated to compensate for this lack in the real data.

Chapter 6

Conclusion

This thesis explores the possibility of target improving an object detector using synthetic data. The proposed approach identifies the weaknesses of the initial model by performing a detailed evaluation and then creating directed synthetic data. When training the model, the synthetic data is then used as training data in combination with the original real data. After training the model, the possible performance improvements are evaluated. The approach showed great potential, remarkably improving the initial model's performance.

First, while some aspects of creating the optimal synthetic training data remain unclear, it is undeniable that the model used in this thesis was improved using synthetic training data. This thesis's best performing training data was somewhat surprising, as this data was considered of worse quality. These results indicate that it is difficult to predict from what data the model profits the most.

Considering the improvement of the model, it is clear from the results that the domain adaptation of the data was sufficient. The usage of character poses, lighting, positioning, and real backgrounds may have contributed to the generalizing of the model. However, whether a synthetic or missing background would have affected the results remain unknown.

No side effects were discovered when evaluating the models trained on synthetic training data. While the model was not examined layer-wise, this indicates that the synthetic data produced in this thesis does not introduce any unwanted side effects, even though most of the training was performed with unfrozen model layers.

In conclusion, the targeted trait, light hair, which was a weakness in the thesis's initial model, was improved by training on the rendered synthetic data. This was true for the three different datasets used. While additional test data would further strengthen these results, it remains clear that synthetic data can be used to improve the performance of an object detection model.

Chapter 7

Future Work

To further understand the results from this thesis and provide additional results within this research area, the following directions for future work are suggested. Below a list of our main recommendations for future work is presented, followed by a more detailed description of each aspect.

- Exploring additional targets
- Alternating training parameters
- Improving the synthetic data

7.1 Exploring Additional Targets

As mentioned, there were multiple weaknesses found within the initial model. However, to fit the scope of this thesis, one of the weaknesses, namely lighter hair detected as hard hats, was chosen for improvement. To further examine the possibility to perform targeted improvement on an object detection model, other weaknesses should be explored. For example, it was found that other hats were often detected as hard hats, especially hats with brighter colours. By applying the approach described in this report, it would be possible to add characters wearing various hats, not being hard hats.

Additionally, the evaluation results showed that small and large bounding boxes were part of the model's weaknesses. Looking into the training data, it can be seen that most of the bounding boxes lie within the medium range. Knowing this, it would be of great interest to explore the possibility of eliminating this weakness using further varied training data. Creating data with the focus of characters being placed especially close to or far away from the camera could further prove the possibility of target improving this weakness.

Additional weaknesses to explore are; hard hats placed randomly in the scene (without connected head) and parts of the background being classified as hard hats, such as reflective surfaces.

7.2 Optimizing Training Parameters

Another aspect that did not fall within the scope of this thesis was optimizing the training parameters when working on improving the object detection model. As mentioned, most of the training parameters were fixed throughout the different training sessions in this project. This was done in order to process the real and synthetic data in the same way and lower the varying parameters between the sessions.

When it comes to freezing layers, for example, studies show different results, see section 1.5.1. Some claim that training with frozen layers provides the best results with synthetic data. In comparison, others show that training without frozen layers provides better results. Additional examining of freezing different model layers and modifying other various model parameters would provide a better overview of how to best benefit from the synthetic data.

7.3 Improving the Synthetic Data

When evaluating the models in this thesis, it was shown that some of the rendered data performed better than others. The reason for this was not fully established, and to further understand what aspects improve the model's performance, it would be necessary to compare different kinds of synthetic data. What improves the quality of the synthetic data can be discussed. However, some of the possible alterations discussed are improving the annotation of the data, including synthetic data with hard hats, and introducing noise in the images. The introduction of noise could, for example, be done using CycleGAN, as in [4].

Bibliography

[1] T. Liu and A. Mildner, "Training deep neural networks on synthetic data," Master's thesis, Dept. Computer Science, Lund University, Lund, 2020.

[2] M. Ballout, M. Tuqan, D. Asmar, E. Shammas, and G. Sakr, "The benefits of synthetic data for action categorization," *International Joint Conference on Neural Networks (IJCNN)*, 2020, [Online]. Available: https://arxiv.org/abs/2001.11091. Accessed: May 16, 2022.

[3] E. Wood, T. Baltrušaitis, C. Hewitt, S. Dziadzio, M. Johnson, V. Estellers, T. J. Cashman, and J. Shotton, "Fake it till you make it: Face analysis in the wild using synthetic data alone," 2021, [Online]. Available: https://arxiv.org/abs/2109.15102. Accessed: May 16, 2022.

[4] O. Harrysson, "License plate detection utilizing synthetic data from superimposition," Master's thesis, Dept. Mathematical Sciences, Lund University, Lund, 2019.

[5] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," in *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, 2018, pp. 1082–10 828.

[6] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," *CoRR*, vol. abs/1703.06907, 2017, [Online]. Available: http://arxiv.org/abs/1703.06907. Accessed: May 16, 2022.

[7] D. Dwibedi, I. Misra, and M. Hebert, "Cut, paste and learn: Surprisingly easy synthesis for instance detection," *CoRR*, vol. abs/1708.01642, 2017, [Online]. Available: http://arxiv.org/abs/1708.01642. Accessed: May 16, 2022.

[8] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[9] IBM, "What is computer vision?" [Online]. Available: https://www.ibm.com/se-en/topics/computer-vision. Accessed: May 16 2022.

[10] R. Szeliski, *Computer Vision: Algorithms and Applications*, ser. Texts in Computer Science. Springer Cham, 2022, [Online]. Available: https://doi-org.ludwig.lub.lu.se/10.1007/978-3-030-34372-9. Accessed: May 16, 2022.

[11] A. Glassner, *Deep learning: From basics to practice*. Seattle, WA, USA: The Imaginary Institute, 2018.

[12] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, "Mobilenets: Efficient convolutional neural networks for mobile vision applications," 2017, [Online]. Available: https://arxiv.org/abs/1704.04861. Accessed: May 16, 2022.

[13] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, [Online]. Available: http://www.deeplearningbook.org. Accessed: May 16, 2022.

[14] A. Géron, *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow : concepts, tools, and techniques to build intelligent systems.*, 2nd ed. O'Reilly Media, 2019.

[15] M. Tan and Q. V. Le, "Efficientnet: Rethinking model scaling for convolutional neural networks," *CoRR*, vol. abs/1905.11946, 2019, [Online]. Available: http://arxiv.org/abs/1905.11946. Accessed: May 16, 2022.

[16] Mathworks, "What is object detection?" 2022, [Online]. Available: https://www.mathworks.com/discovery/object-detection.html. Accessed: May 16, 2022.

[17] W. Liu, D. Anguelov, D. Erhan, C. Szegedy, S. E. Reed, C. Fu, and A. C. Berg, "SSD: single shot multibox detector," *CoRR*, vol. abs/1512.02325, 2015, [Online]. Available: http://arxiv.org/abs/1512.02325. Accessed: May 16, 2022.

[18] T. Lin, M. Maire, S. J. Belongie, L. D. Bourdev, R. B. Girshick, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft COCO: common objects in context," *CoRR*, vol. abs/1405.0312, 2014, [Online]. Available: http://arxiv.org/abs/1405.0312. Accessed: May 16, 2022.

[19] COCO Consortium, "Detection evaluation," [Online]. Available: https://cocodataset.org/#detection-eval. Accessed: March 3, 2022.

[20] D. Bolya, S. Foley, J. Hays, and J. Hoffman, "Tide: A general toolbox for identifying object detection errors." *Computer Vision – ECCV*, 2020, [Online]. Available: http://arxiv.org/abs/2008.08115. Accessed: May 16, 2022.

[21] Nvidia, "Real-time ray tracing," [Online]. Available: https://developer.nvidia.com/rtx/ray-tracing. Accessed: May 2, 2022.

[22] N. Alarcon, "Ray tracing essentials part 1: Basics of ray tracing," [Online]. Available: https://developer.nvidia.com/blog/ray-tracing-essentials-part-1-basics-of-ray-tracing/. Accessed: April 26, 2022.

[23] E. Davies, *Computer Vision*, 5th ed., E. Davies, Ed. Academic Press, 2018, [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9780128092842000034. Accessed: May 16, 2022.

[24] M. H. F. Wilkinson and J. B. Roerdink, "Mathematical morphology and its application to signal and image processing." in *9th International Symposium on Mathematical Morphology*, ser. Image Processing, Computer Vision, Pattern Recognition, and Graphics: 5720. Springer Berlin Heidelberg, August 24-27 2009.

[25] Voxel51, "Fiftyone home page," 2022, [Online]. Available: https://voxel51.com/docs/fiftyone/. Accessed: May 5, 2022.

[26] Blender, "Blender home page," 2022, [Online]. Available: https://www.blender.org/. Accessed: May 5, 2022.

[27] Reallusion, "Character creator home page," 2021, [Online]. Available: https://www.reallusion.com/character-creator/. Accessed: May 5, 2022.

[28] Northeastern University China, "Hard hat workers dataset," April 2020, [Online]. Available: https://public.roboflow.com/object-detection/hard-hat-workers. Accessed: May 16, 2022.