Dublin City University

School of Electronic Engineering

Exploring Synthetic Image Generation for Training Computer Vision Models under Data Scarcity

Enric Moreu, B.E., M.E.

A Dissertation submitted in fulfilment of the requirements for the award of Doctor of Philosophy (PhD)

Supervised by Prof. Noel E. O'Connor

Co-supervised by Dr Kevin McGuinness

January 2024

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Doctor of Philosophy is entirely my own work, and that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Enric Moreu

Student number: 18215178

3rd of January 2024

Acknowledgements

I would like to express my sincere gratitude to my supervisors, Prof. Noel O'Connor and Dr Kevin McGuinness, for their guidance throughout my PhD journey. Their invaluable advice, encouragement, and critical feedback have been instrumental in shaping my research and helping me grow as a scholar.

I would also like to thank Dr Eric Arazo for his invaluable support. Eric's expertise and guidance were crucial in helping me navigate the challenges of my research. His availability and willingness to provide assistance and advice were greatly appreciated, and I am deeply grateful for his contributions to my academic development.

Finally, I am profoundly thankful to my friends and family for their love and support throughout my studies. Their belief in me has been a constant source of motivation and strength.

Moltes gràcies a tots.

List of publications

- **Enric Moreu**, Eric Arazo, Kevin McGuinness, Noel E. O'Connor. "Self-Supervised and Semi-Supervised Polyp Segmentation using Synthetic Data". In: *International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2023, pp. 1–9.

- **Enric Moreu**, Alex Martinelli, Martina Naughton, Philip Kelly, Noel E. O'Connor. "Fashion CUT: Unsupervised domain adaptation for visual pattern classification in clothes using synthetic data and pseudo-labels". In: *Scandinavian Conference on Image Analysis (SCIA)*. Springer. 2023, pp. 314–324.

- **Enric Moreu**, Eric Arazo, Kevin McGuinness, Noel E. O'Connor. "Joint one-sided synthetic unpaired image translation and segmentation for colorectal cancer prevention". In: *Expert Systems 40.6*. 2022, e13137.

- **Enric Moreu**, Diego Ortego, Kevin McGuinness, Noel E. O'Connor. "Domain Randomization for Object Counting". In: *Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*. 2021.

- **Enric Moreu**, Kevin McGuinness, Noel E. O'Connor. "Synthetic data for unsupervised polyp segmentation". In: *Irish Conference on Artificial Intelligence and Cognitive Science (AICS)*. 2021.

Contents

List of Tables

List of Figures

Acronyms and Abbreviations

- BSP: Batch Spectral Penalisation
- CAL: Collaborative Adversarial Learning
- CNN: Convolutional Neural Network
- CUT: Contrastive Unpaired Translation
- CVC-ClinicDB: Computer Vision Centre Clincic Database
- CVC-ColonDB: Computer Vision Centre Colonoscopy Database
- CVC-T: Computer Vision Centre Texture
- DA: Domain Adaptation
- DR: Domain Randomisation
- FDA: Fourier Domain Adaptation
- FOV: Field Of View
- GAN: Generative Adversarial Network
- GTA V: Grand Theft Auto V
- IoU: Intersection over Union
- MAE: Mean Absolute Error
- MDD: Maximum Density Divergence
- MSE: Mean Squared Error
- MinneApple: Minnesota Apple
- PGCNet: Perspective-Guided Convolution Network
- PL-CUT-Seg: Pseudo-label Contrastive Unpaired Translation for Segmentation
- PraNet: Parallel Reverse Attention Network
- R-CNN: Region-based Convolutional Neural Network
- SANet: Slice-Aware Network
- SDG: Synthetic Data Generation

- SFA: Selective Feature Aggregation
- SHT A: ShanghaiTech Part A
- SHT B: ShanghaiTech Part B
- TRANCOS: Traffic Congestion Estimation
- UACANet: Uncertainty-Aware Cascaded Attention Network
- YOLO: You Only Look Once

Abstract

"Exploring Synthetic Image Generation for Training Computer Vision Models under Data Scarcity"

Enric Moreu

This thesis presents research conducted in the area of synthetic data generation for computer vision tasks. The research aims to address the challenge of data-hungry deep learning models by generating synthetic images that can effectively train computer vision models to solve tasks such as object counting, polyp segmentation, and pattern classification. The work carried out explores the use of various techniques to ensure effective use of synthetic data, including domain randomisation and domain adaptation in both self- and semi-supervised setups. Through the application of these techniques, the research aims to develop a robust and effective approach for generating synthetic data that can improve the performance of computer vision models with a reduced amount of human annotations.

Chapter 1

Introduction

1.1 Computer Vision Applications

Computer vision is a well-established but rapidly growing field that enables computers to interpret and understand visual data from the real-world. This includes tasks such as classification, detection, and segmentation, among others (see Figure 1.1). The rise of computer vision has been driven by factors, including the availability of large amounts of data, the development of powerful machine learning algorithms, and the increasing ability of computers to process data efficiently. The applications of computer vision are diverse and range from consumer applications, such as those used in image and video search engines, to more specialised uses in industries such as healthcare, manufacturing, and security. As the field of computer vision continues to grow and evolve, its applications are likely to become even more widespread and impactful.

In this section, a formal introduction to some relevant tasks within computer vision [3], including classification, detection, segmentation, and counting is provided.

1.1.1 Classification

Given a classification dataset $\mathcal{D}$ comprising pairs (x_i, c_i) where x_i denotes an image and c_i represents the class label, a model determines the category or class of the

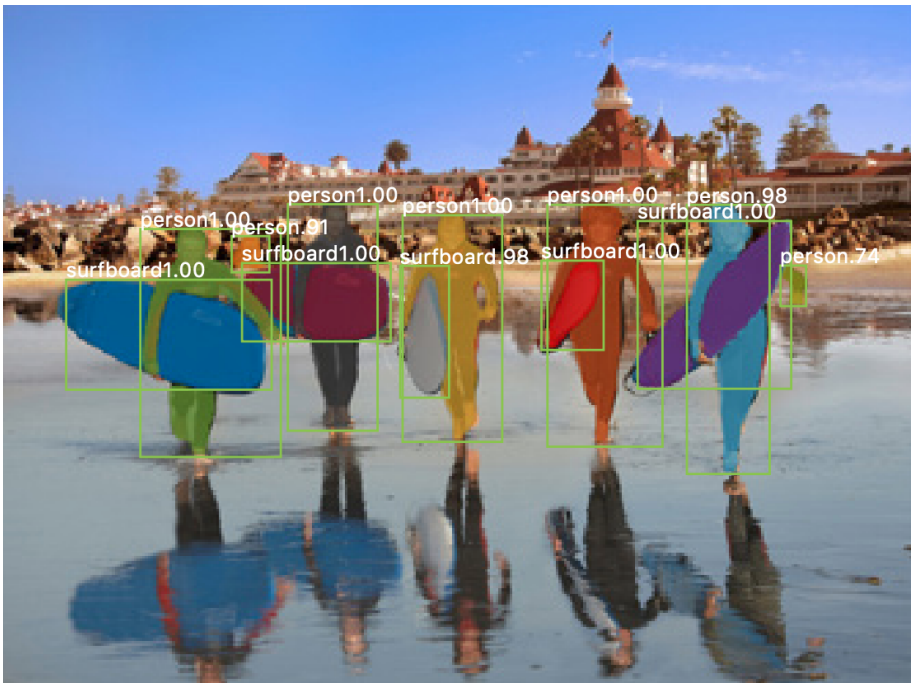


Figure 1.1: Object instance segmentation sample from He et al. [1]. The algorithm us applied on an image from the COCO dataset [2].

entire image. For instance, if we're classifying images like 'cat' and 'dog', the output for an image x_i would be either 'cat' or 'dog':

$$c_i \in \{\text{'cat'}, \text{'dog'}\}$$

Chapter 6 is focused on the classification task for fashion pattern classification using synthetic data. Synthetic images and labels (x_i, c_i) are used to train the model.

1.1.2 Detection

In the context of object detection, $\mathcal{D}$ consists of pairs (x_i, B_i), where x_i is an image and B_i is a set of bounding boxes with corresponding class labels. The aim is to identify and localize multiple objects within the image. Given an image x_i, the model predicts bounding boxes b_j and their associated class labels c_j:

$B_i = \{(b_j, c_j) | b_j$ is the bounding box and c_j is the class label for the j^{th} object in $x_i\}$

This allows the model to detect and classify multiple entities within a single image, providing spatial information about where each entity is located.

1.1.3 Semantic segmentation

In semantic segmentation, $\mathcal{D}$ comprises pairs (x_i, y_i) of images and segmentation masks. The objective is to predict a label for each pixel $x_i^{(n,m)}$, its corresponding label $y_i^{(n,m)}$ would belong to one of the predefined categories, such as 'car', 'tree', 'road', etc.:

$$y_i^{(n,m)} \in \{\text{'car', 'tree', 'road', } \dots\}$$

The primary goal is to segment the image based on semantic meaning, differentiating between distinct object classes and the background. Chapters 3, 5, and 5 are focused on the semantic segmentation task for colorectal polyps.

1.1.4 Instance segmentation

Instance segmentation can be considered an extension of semantic segmentation. While semantic segmentation predicts a class label for each pixel, instance segmentation also differentiates between distinct instances of the same class. Given $\mathcal{D}$ with pairs (x_i, z_i), a model predicts not just the class label but also an instance identifier for each pixel. For instance, two cars in an image would have the same class label 'car' but different instance identifiers:

$$z_i^{(n,m)} = (y_i^{(n,m)}, i_{\text{id}})$$

Where $y_i^{(n,m)}$ is the class label and i_{id} is the instance identifier. This distinction allows for understanding both the kind of object and its unique occurrence in an image.

1.1.5 Object counting

Object counting is the task of quantifying the number of specific objects present within an image or sequence of images. Given $\mathcal{D}$ with image and object pairs (x_i, O_i), where x_i represents an image and O_i is a set of object instances, a model aims to determine the cardinality of O_i, denoted as $|O_i|$ for each image:

$$\forall x_i \in \mathcal{D} : \text{Count}(x_i) = |O_i|$$

Unlike detection, where the spatial positioning of each object is of primary concern, or segmentation, which focuses on delineating the precise boundaries of objects, counting emphasizes the quantification of instances. It's particularly challenging in scenarios where objects overlap or are densely packed, making individual distinction difficult. Chapter 2 is focused on the object counting task for people, vehicles, fruit, and animals.

1.2 Convolutional Neural Networks

Building upon the foundations of computer vision, it's important to note that its recent advancements have been largely fuelled by the emergence of Convolutional Neural Networks (CNNs) [4]. These are a specific type of neural network designed to process data with a grid-shaped structure, such as images.

CNNs contain multiple grid-shaped layers of interconnected nodes or neurons. These networks can learn representations of data in a layered hierarchy, with higher layers capturing more abstract and composite features. The depth of these networks, meaning the number of layers, has played a crucial role in their ability to perform complex tasks. The operation of a neuron in a layer can be described as:

$$y = f\left(\sum_i w_i x_i + b\right)$$

where x_i are the input values, w_i are the weights, b is a bias term, and f is an

activation function such as the rectified linear unit [5]. The combination of these simple mathematical operations across numerous neurons and layers enables the network to model complex non-linear relationships.

In the context of a two-dimensional image, a convolution involves the use of a filter or kernel to slide over the image. Given an image matrix I and a filter matrix F, the convolution operation for a specific position is computed as:

$$(C * F)(x, y) = \sum_{i=-\infty}^{\infty} \sum_{j=-\infty}^{\infty} I(i, j) \cdot F(x - i, y - j)$$

Where $(C * F)(x, y)$ is the output feature map after the convolution operation. The filter, usually a small matrix, identifies certain features from the input matrix based on how it's trained.

Training a deep neural network involves the use of algorithms like backpropagation [6]. Given a dataset, the network makes a prediction, and the error between this prediction and the true label is computed using a loss function. Backpropagation, combined with optimisation techniques such as stochastic gradient descent [7], adjusts the weights of the network to minimize this error.

One of the main reasons that CNNs are so effective is that they can automatically learn the features that are relevant for a given task, rather than requiring the features to be manually designed by a human. This is useful for tasks where abstract features that are important for the application are hard for a human to define. Additionally, CNNs can take advantage of the spatial structure of the data, which is important for tasks such as object recognition, where the relative position of the objects in the image is important.

However, there are some challenges associated with CNNs. They require a large amount of data to be trained effectively. This can be a problem because it can be difficult and time-consuming to collect and annotate large amounts of data, especially for tasks that require specialised knowledge or expertise, such as medical imaging. In addition, using large amounts of data raises privacy concerns, as it may be difficult to ensure that personal data is protected when it is used for training a

CNN. Furthermore, the performance of a CNN can vary depending on the quality and diversity of the data, so it is important to curate the data used for training carefully. For example, when training a vision model for a self-driving car, it is equally important to use images with cloudy weather and sunny weather.

1.3 Synthetic data

Synthetic data is any sample that is not obtained by direct measurement in the physical world. In the context of computer vision, synthetic data can be generated using software that produces images that bear a resemblance to real-world images (see Figure 1.2). The use of synthetic data can help to solve some of the above-mentioned challenges associated with training CNNs.

Figure 1.2: Sample from the GCC synthetic dataset [8] generated using a game engine.

One of the key benefits of using synthetic data is that it can be generated in large quantities. This allows CNNs to be trained on a much larger amount of data than would be possible using real-world data alone. In addition, synthetic data does not require any sensitive or private information during its generation. Hence, synthetic datasets can ensure that no personal information is used to train the model. Furthermore, because synthetic data is generated using algorithms, it can be generated with a wide range of random variations, allowing CNNs to be trained on

diverse and varied data sets. This can help to improve the performance of the CNN, as it will be better able to handle a wider range of inputs.

Last but not least, synthetic data can be automatically annotated during the synthetic data generation process. This is important because one of the challenges of training CNNs is the need for large amounts of annotated data, which can be time-consuming and labour-intensive to produce. By using synthetic data, the annotations can be generated automatically, which can save time and effort. For example, a human annotator can spend up to 7 minutes [9] to annotate all the pixels in an image while a rendering engine can do this in less than a second and in an error-free manner.

1.3.1 Domain shift

While synthetic data can be useful for some purposes, it is not always useful for computer vision tasks. One of the reasons for this is the "domain shift", also referred to as "domain gap" or "reality gap".

Domain shift is a problem that arises when the distribution of data in the training set differs from the distribution of data in the test set. This can reduce performance because a model that is trained on one distribution of data may not perform well on a different data distribution. In the context of synthetic data, domain shift can be a problem because synthetic data is often generated under controlled conditions, which can result in a distribution of synthetic data that differs from the distribution of real-world data. As a consequence, the computer vision model won't be able to generalise properly because it will be biased by the synthetic data. For example, if a model is trained on synthetic data that has been generated to have perfect lighting and clear shapes, it may not perform well on real-world images that have low lighting, noise, or blur – see Figure 1.3 for a comparison of a real and a synthetic image of an urban environment.

Domain shift is a well-known problem in computer vision, and there are a variety of techniques to overcome it, such as transfer learning or data augmentation. This

Figure 1.3: Comparison of real-world image from the Cityscapes dataset [9] and synthetic image from the Synthia dataset [10].

thesis studies how domain randomisation and domain adaptation techniques can reduce the effects of domain shift when training on synthetic data.

1.4 Literature review

1.4.1 Deep Convolutional Networks

The concept of a neural network can be traced back to the perceptron model introduced in the late 1950s by Rosenblatt et al. [11]. However, it wasn't until the advent of powerful computational hardware and large datasets that deep learning truly began to flourish in the 2010s with AlexNet [12].

An important advancement in deep learning was the introduction of regularisation techniques, like dropout [13]. Dropout involves randomly setting a fraction of input units to 0 at each update during training, preventing overfitting and leading to a more robust network. This form of regularisation was essential in ensuring that deep networks generalize well to unseen data.

Another cornerstone in the evolution of deep learning is transfer learning [14], where a pre-trained model on a large dataset is fine-tuned for a specific task with a smaller dataset. This technique leverages the generic features learned by the model,

reducing the need for large labelled datasets and computational resources in diverse applications.

Deep learning, underpinned by these concepts and advancements, has become the bedrock of numerous applications, from computer vision and natural language processing to medical diagnosis and autonomous driving. Its capacity to extract intricate patterns and representations from vast amounts of data has continually pushed the boundaries of what machines can perceive and achieve.

One significant advancement in CNNs has been the inception of skip connections [15]. In deep networks, training becomes challenging due to the vanishing and exploding gradient problems, which can stagnate learning or even make it counter-productive. Skip connections tackle this by providing an identity shortcut connection that bypasses one or more layers. This approach not only alleviates the gradient issues but also allows for the training of much deeper networks.

Another major enhancement has been the utilisation of dilated convolutions [16], which can increase the receptive field of the network without increasing the number of parameters or computational costs. Dilated convolutions expand the kernel by introducing gaps, enabling the network to capture a broader context without the need for pooling or increasing the convolutional kernel size. This has been particularly useful for segmentation tasks where fine-grained spatial details are imperative.

In addition to these improvements, the application of normalisation techniques, like batch normalisation [17], has revolutionised training dynamics. By normalising the activations of each layer to maintain a mean of zero and a variance of one, batch normalisation not only accelerates training but also mitigates the internal covariate shift, which refers to the change in the distribution of network activations due to the changing parameters during training.

1.4.2 Synthetic datasets for computer vision applications

Since the early days of computer vision, synthetic datasets have been used to avoid gathering and annotating images from the real-world. This section reviews different

approaches to generating synthetic datasets, from primitive 2D shapes to photo-realistic images from a videogame.

Data augmentation is often used as a replacement for synthetic data because it can significantly expand the size of the dataset. Data augmentation is the process of performing simple operations to produce additional training data by rotating the image, stretching it or altering the colours (see Figure 1.4). It is widely used in deep learning because it can considerably improve generalisation. For example, applying a simple horizontal flip to a dataset will double the amount of training data available. Modern data augmentation techniques can increase the dataset size by a magnitude of 2,048 as in AlexNet [12]. These operations can dramatically improve the performance of the model because they increase the data available for training.

Figure 1.4: Data augmentation operations using the Albumentations package [18]. Sample from the Albumentations documentation.

Data augmentation can improve the performance of a model with very low overhead since it only requires simple CPU operations to rotate or crop an image.

However, data augmentation should be used with caution because it may alter the nature of the image, e.g. stretching a healthy chest X-ray image will modify the shape of the heart, potentially causing it to be classified as an anomaly. Last but not least, data augmentation requires an existing dataset from the real world to generate variations of the data.

It is worth noting that data augmentation is not exactly the same as synthetic data. While both methods are useful for creating new training data, synthetic data is generated from scratch, while data augmentation uses the existing data to create new samples. Synthetic data also provides additional benefits, such as privacy and security protection.

One of the early studies that exclusively used synthetic images was conducted by Pomerleau [19] in 1991, who used a computer graphics program to generate 1,200 synthetic images of roads and trained a neural network to identify road features like signs and lanes. Pomerleau found that the neural network trained on synthetic data performed well on real-world data, demonstrating the potential of synthetic data for training and evaluating machine learning algorithms.

Rahnemoonfar et al. [20] generated a synthetic dataset for fruit counting using simple 2D primitive shapes (see Figure 1.5). The synthetic dataset is composed of 24,000 images representing tomatoes on a green background. For each image, an annotation is generated based on the number of red circles used (the number of tomatoes).

Figure 1.5: Generation of a synthetic image for counting tomatoes using simple 2D circles by Rahnemoonfar et al. [20].

With the ongoing rapid evolution of modern rendering engines [21], 3D-based synthetic datasets provide additional features like realistic lighting and camera effects

such as perspective or blur. MPI Sintel [22] is an animation film generated with 3D software that the authors released as a synthetic dataset, including the flows, depth, occlusions, and other useful data alongside the film (see Figure 1.6). It is one of the main benchmarks for the task of optical flow prediction. Optical flow is the task of estimating the movement between two images, typically consecutive frames of a video. An optical flow model produces a map of the direction and magnitude of movement for each pixel between the two frames.

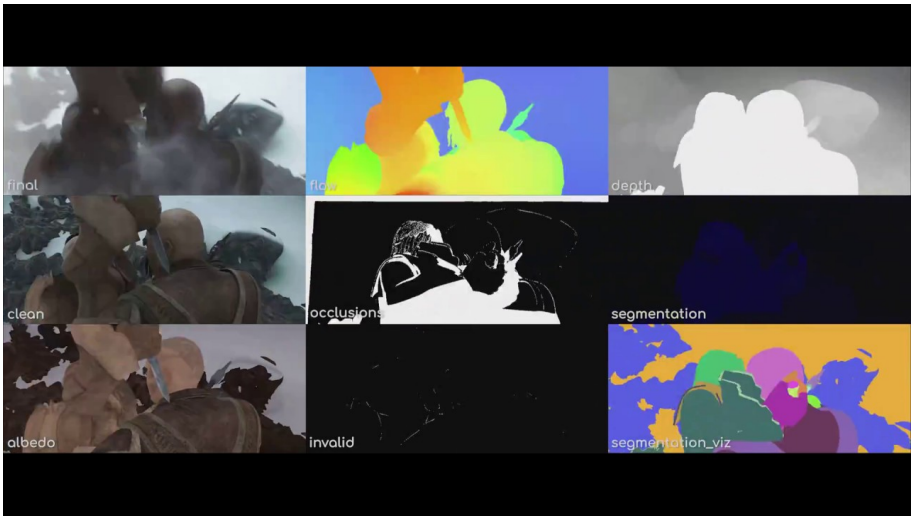


Figure 1.6: Sample from the Sintel [22] dataset

The Flying Chairs dataset [23] is also used for optical flow estimation. It is composed of 22,872 image pairs generated with backgrounds from Flickr and 3D models of chairs from ShapeNet [24] with random displacements and rotations. More complex versions of The Flying Chairs dataset were subsequently generated, including other objects, realistic lighting, and improved background. Figure 1.7 displays a sample from the FlyingThings3D dataset [25], which is an extended version of the Flying Chairs dataset that contains a wider variety of 3D models. In addition, the background is composed of large textured primitive shapes, rather than plain images.

However, optical flow models are extremely sensitive to the distribution of movements in the training dataset. Mayer et al. [26] found that if the distribution of movements in the training dataset is significantly different from the distribution of movements in the test dataset, the model may not perform well on the test data

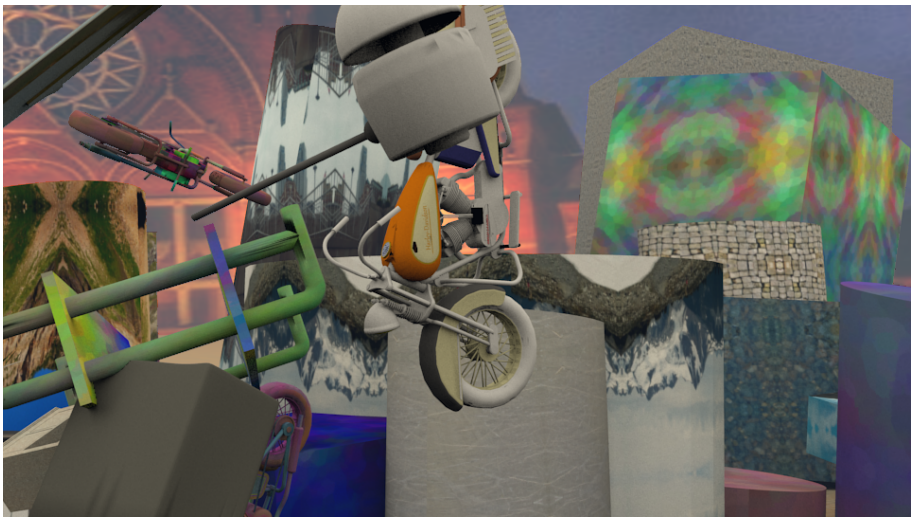


Figure 1.7: Sample from the FlyingThings3D dataset [25] for optical flow estimation.

and may produce poor results. This is because the model has not been exposed to a sufficient variety of movements during training and is not able to generalise well to new situations. For example, an optical flow prediction model trained on a video of a tennis match will perform well on similar datasets with limited movements (e.g. players and ball), but may not generalise to other types of sports videos with diverse movements.

Richter et al. [27] produced 25,000 synthetic images using the game engine from the Grand Theft Auto V (GTA V)[27] videogame (see Figure 1.8). GTA V uses realistic lighting and reflections combined with effects such as motion blur or lens distortion. Although the GTA V dataset may appear photorealistic, the performance of a computer vision model trained on it may not be as expected. One might assume that investing more effort in generating more realistic data (e.g., using a team of 3D artists and a custom render engine) would result in a model that performs better. However, the performance of the model depends more on using the appropriate machine learning techniques to address the domain shift effect, as demonstrated by Wang et al. in their study [8]. The authors found that, without domain adaptation, the performance of the model was poor even when the images were highly realistic. If the right techniques are not used, the model may simply memorise the synthetic traits that look realistic to humans (e.g., dynamic lighting, high-quality textures, or

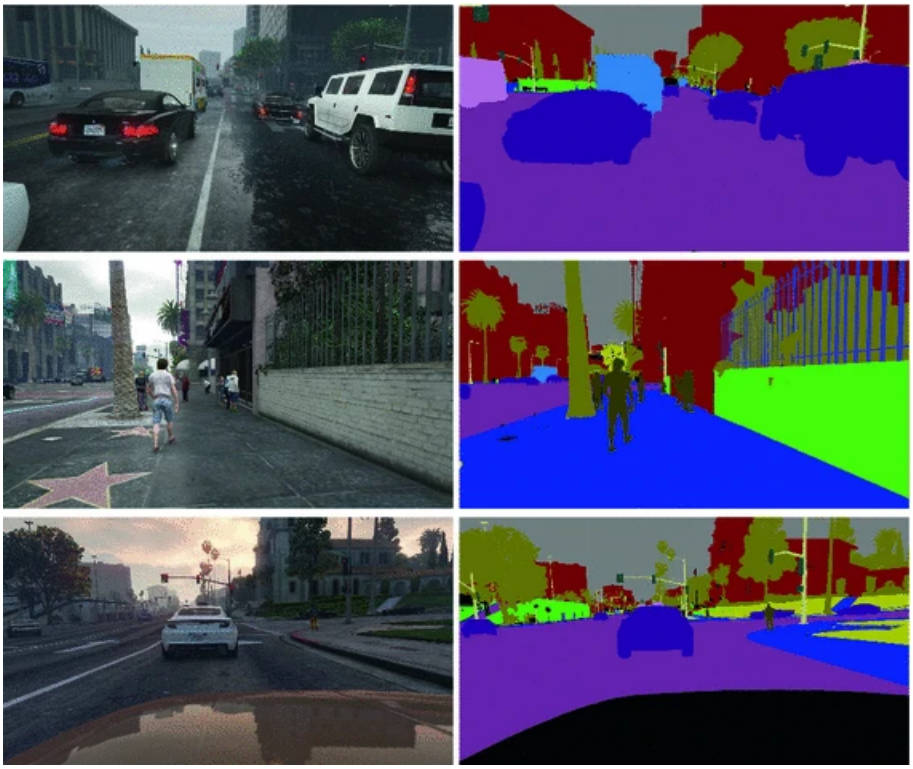


Figure 1.8: Samples extracted from the GTA V videogame for image segmentation[27].

high-poly models), but are not necessarily relevant for the target dataset.

The synthetic representation in Figure 1.9 illustrates how this problem may arise. The argument made in the previous paragraph is that despite the complex features of the simulated setting, which includes 3D models, lighting, shadows, and textures, the model might incorrectly learn to identify anything with an asphalt texture as a road. This could result in misidentification of real-world buildings with similar textures as roads. Note that the asphalt texture in the synthetic image is uniform, unlike real asphalt which varies in colour and may have different markings or debris. The study by Geirhos et al. [28] supports the observation that CNNs often make such errors by showing how an image of a cat is incorrectly classified as an elephant because the model favours the use of texture recognition over other key features like shape.

The model may learn details that are not important for the target domain but only serve to impress human viewers.

Figure 1.9: Sample from the Synthia dataset [10] simulating a urban environment.

1.4.3 Overcoming the Domain Shift Problem

Transfer learning is the most common approach to reduce the negative impact of domain shift. First, a large model is trained on a source dataset from a source domain where data and annotations are abundant. Then, the model is fine-tuned on the target domain using a smaller dataset. The concept was first introduced in 1976 by Bozinovski et al. [29]. Transfer learning is especially effective when the target domain is small or lacks annotated data. It also has the added benefit of speeding up the training process. This is because the model is initialised with weights that have already been trained on a large dataset, which allows it to converge faster towards the target domain. One major limitation of transfer learning, however, is that it requires a large, annotated dataset in the source domain, which may not always be available.

An alternative to reducing the impact of domain shift is Domain Randomisation (DR). It was first proposed by Tobin et al. [30] as a technique for reducing the domain gap in computer vision tasks. They demonstrated the effect of varying the texture of objects, background images, and lighting in a semantic segmentation task. The goal of DR is to create sufficient variations in the synthetic data such that the model views real data as just another variation, even if the variations used for training may appear unrealistic to humans. If the model is trained on a sufficient number of environments, it will perform well on novel ones. Domain randomisation can be

seen as an advanced form of data augmentation and is especially effective when the target domain is significantly different from the source domain. Figure 1.10 illustrates how Tremblay et al. [31] use DR to produce a synthetic dataset for urban image segmentation by combining 3D models of vehicles with random 3D shapes and textures.

Figure 1.10: Sample from a domain randomisation dataset [31]. Random 3D shapes and textures are introduced in the scene to force the model to generalise better.

While domain randomisation is a useful technique for reducing the domain gap and improving performance on the target domain, it has its limitations. For example, it may not be practical to generate enough variations in synthetic data to cover the full range of variations in the target domain. Additionally, the model may not generalise well to the target domain if the variations used for training are significantly different from those in the target domain.

To address these limitations, domain adaptation techniques [32] have been proposed as an alternative approach for reducing the domain gap. Unlike domain randomisation, domain adaptation aims to make the data more realistic. Domain adaptation involves adjusting the source domain to the target domain using unsupervised or semi-supervised learning methods. This approach is effective in reducing the domain gap, especially when there is limited labelled data in the target domain. Figure 1.11 illustrates the distinctions between domain adaptation and domain

randomisation.

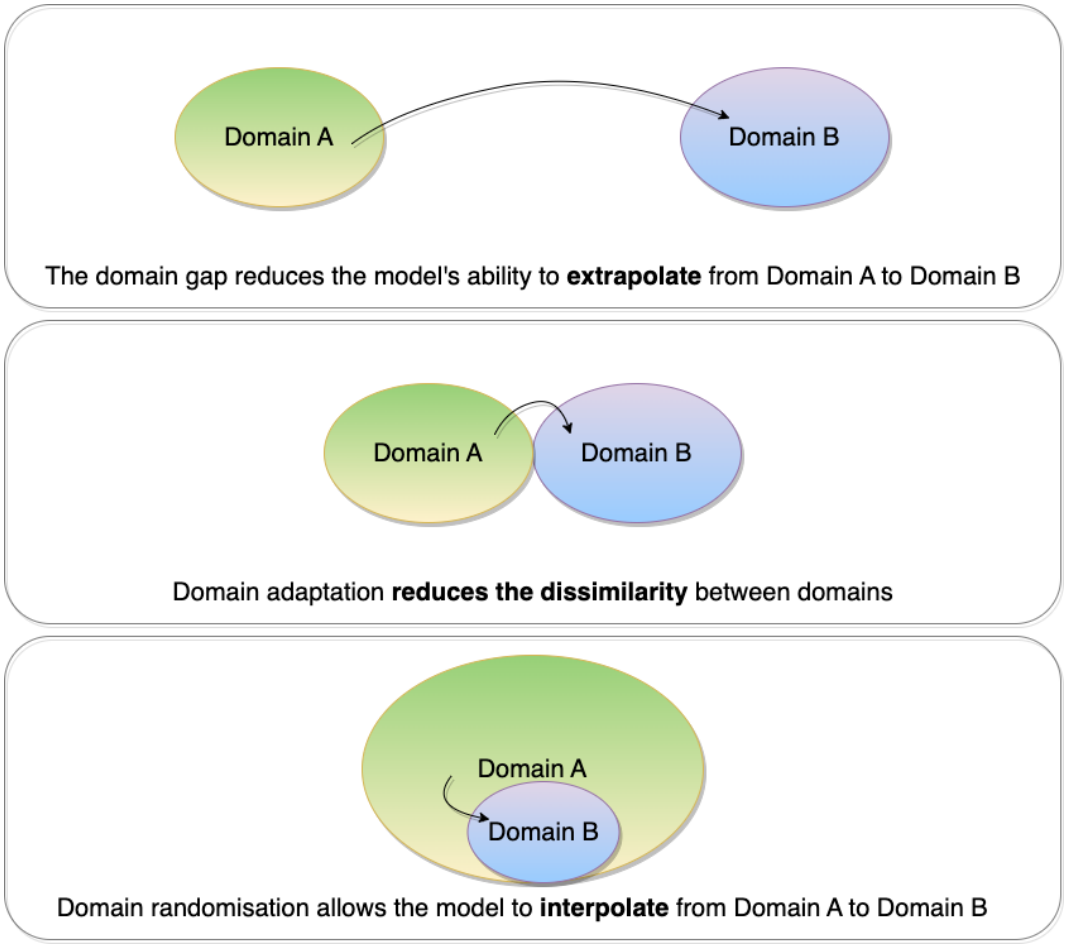


Figure 1.11: Diagram illustrating how domain adaptation and domain randomisation can address the domain gap issue.

Before the development of GANs (Generative Adversarial Networks), Gatys et al. [33] introduced style transfer: a method for manipulating images by combining the style and content of two separate images. This approach, which served as a precursor to GANs, involved using a CNN to extract the style and content representations of each image, and then combining these representations to create a new image with the desired style and content (see Figure 1.12). Style transfer has become a widely used tool in the field of domain adaptation because it allows the target domain to be set as the style.

Style transfer, however, is not particularly efficient, as it requires optimising the combination of the style and content for each image. In contrast, GANs such as CycleGAN [35] can perform image style transfer through inference, rather than

Figure 1.12: Style transfer samples that combine the content of Chicago's skyline (first row) with the style of "The Great Wave off Kanagawa" by Hokusai (second row) and "Mysterious Rain Princess" by Leonid Afremov (third row). Images were generated using the Fast Style Transfer implementation [34].

optimisation, which makes them much faster and more efficient.

GANs [36] are considered to be a highly effective technique for domain adaptation. A GAN is a type of neural network that consists of two parts: a generator and a discriminator. These two parts work together in a min-max game, where the generator tries to produce samples that are indistinguishable from the target domain, while the discriminator tries to distinguish between the generated samples and real samples from the target domain. The generator and discriminator continue to improve and "compete" with each other until the generator can produce samples that are indistinguishable from the real samples in the target domain.

Wang et al. [8] use CycleGAN to translate synthetic images of crowds to the real-world domain. Their translation model learned a mapping between synthetic

and real images in the context of crowded scenes. Unpaired image translation models are effective for domain adaptation in computer vision tasks since they don't require one-to-one correspondence between the images in the two domains. Despite the usefulness of CycleGAN, the lack of label information can make it difficult for unpaired image translation models to accurately translate images with complex labels like segmentation masks. More details about CycleGAN are provided in Chapter 3.

In conclusion, the use of synthetic data for computer vision has gained increasing attention because CNNs require large datasets to achieve good performance. Using expensive 3D techniques to generate synthetic data does not always produce satisfactory results, as the model may simply memorise the synthetic traits rather than learning generalisable features. Among the various approaches that have been proposed to address this problem, GAN-based methods have shown promising results for domain adaptation.

The limitations mentioned in this section are addressed in this thesis, with a focus on the difficulty of acquiring large annotated datasets. To overcome this challenge, the use of synthetic data and domain randomisation is proposed as a cost-effective solution for data generation. The limitations of domain adaptation techniques in utilising real-world annotations in a semi-supervised manner are also addressed.

1.5 Hypotheses and Research Questions

This thesis proposes three hypotheses (H1, H2, H3), each of which requires a specific research question (RQ1, RQ2, RQ3) to be investigated. These are outlined in the following.

H1 (Chapter 2): It is possible to train a computer vision model exclusively using low-poly synthetic data and without requiring experienced 3D artists and advanced rendering engines.

> **RQ1**: Can domain randomisation compensate for the lack of realism in synthetic images and successfully train a computer vision model using

low-quality 3D assets?

H2 (Chapters 3, 4): The incorporation of real-world annotations may enhance the realism of images produced through domain adaptation techniques.

> **RQ2**: What are some ways that domain adaptation can leverage information from segmentation labels to generate images that are more realistic and that are consistent with the segmentation labels?

H3 (Chapters 5, 6): Models trained using domain adaptation are exposed only to a "transformed" version of the data distribution and never encounter genuine real-world images. Assigning soft labels to images from the real world may help the model to better understand the target domain.

> **RQ3**: How can real-world data be incorporated into a domain adaptation process to improve the generalisation capabilities of the model?

RQ1 is addressed in Chapter 2, RQ2 is addressed in Chapters 3 and 4. Finally, RQ3 is addressed in Chapters 5 and 6.

1.6 Thesis structure

The remainder of the thesis is structured as follows. In Chapter 2, a domain randomisation approach is applied to synthetic data for the task of counting vehicles, penguins, people, and fruit. In Chapter 3, domain adaptation is presented as an alternative to domain randomisation. This chapter addresses the detection of colorectal polyps by using synthetic images and an image translation model. In Chapter 4, a new label-aware method is proposed to generate improved synthetic images for the polyp segmentation task. In Chapter 5, real-world images and labels are incorporated into the model's training to reduce the domain gap in polyp segmentation. In Chapter 6, an example of a commercial application of the techniques from the previous chapters is presented. Label-aware domain adaptation and pseudo-labels are applied to the fashion pattern classification task. Chapter 7 provides a

summary of the research in this thesis, a discussion of the research questions and possible directions for future research on synthetic data.

Finally, Appendix A (Section 7.4) includes a technical framework that illustrates how to produce synthetic data using computer graphics software. The framework is designed to be easy to replicate, facilitating other researchers to quickly generate synthetic data.

Chapter 2

Domain randomisation

In this chapter, the technique of Domain Randomisation (DR) is employed to address Research Question 1 (RQ1) in the context of object counting. Object counting is a particularly well-suited task for the exploration of DR techniques because of the diversity of textures, occlusions, shapes, and high noise present in the majority of object counting datasets. DR can reduce the impact of the domain gap by generating highly variable samples at the cost of increasing the complexity of the task. The objective of increasing variability is to expand the spectrum of possibilities of the source domain whereby the real-world domain becomes just another variation. The synthetic samples generated with this technique tend to look less realistic because of the random textures, lighting, and backgrounds used. DR avoids photorealism, minimising the need for artistic design. The work reported in this chapter has been presented at the 29$^{\text{th}}$ Irish Conference on Artificial Intelligence and Cognitive Science (AICS) [37]. The code to replicate the experiments is available in the online repository of the paper.

2.1 Introduction

Object counting is a computer vision task the goal of which is to automatically estimate the number of objects in an image or video. It has gained a lot of interest in recent years because of its many potential uses: it can help to identify the congestion

level in a shopping centre [38] (crowd counting), the level of traffic on a road [39] (vehicle counting), the state of a harvest [20] (fruit counting), or even the state of a penguin colony [40] (habitat monitoring).

The challenges of object counting are that the model has to learn all the variations of the objects in terms of their size, shape, and pose whilst also dealing with occlusion and perspective effects. Furthermore, object counting algorithms tend to overfit because of the small amount of annotated data available, which degrades their performance when applying the model to other slightly different domains.

To address the problems above, some counting algorithms are trained/pretrained using synthetic data, which can be automatically annotated with perfect precision for a range of domains, including those where collecting data is problematic.

People counting in particular has benefited from modern video games [27] because they have abundant scenes of crowds in an urban context, which can be easily used to produce synthetic datasets for counting people. However, video games often focus on creating realistic environments filled with humans and vehicles, but they usually don't include penguins or apples. Because of this, the data produced by video game engines isn't helpful for realistic representations of these items. It is not practical to create realistic datasets for many different tasks because of the significant manual effort and production costs required [41]. However, some video games [27] allow modifying the source code and upload custom 3D models. High-quality models that are not present in the video game (e.g. an apple) can be uploaded as long as the 3D model and textures are available.

Furthermore, models trained with synthetic images from a particular domain perform poorly when tested on a different target domain because of the domain gap, which, as discussed in Chapter 1, has posed considerable obstacles to real-world adoption of synthetic data for computer vision applications. The main cause is that convolutional neural networks introduce a bias towards textures [28], memorising them instead of shapes. For object counting, understanding the shape of the items is of paramount importance to address the challenges of overlapping objects and

(a) Crowd counting

(b) Vehicle counting

(c) Harvest study

(d) Environmental survey

Figure 2.1: Synthetic images generated using the proposed domain randomisation approach

occlusions. Textures provide less useful information because they tend to differ from real-world textures, where the lighting and the details are more complex.

A domain randomisation approach for object counting is proposed in this chapter. DR can be used to generate synthetic datasets for counting people, vehicles, penguins and fruits. Figure 2.1 shows several examples of DR applied to different environments. The variability of the synthetic dataset is increased by applying random textures, backgrounds and lighting effects to the 3D scene. Additionally, a set of 3D transformations is applied to the 3D models. Transformations increase the variability of the 3D models while preserving their inner shape, making the task more complex during training but improving generalisation during testing. This approach achieves excellent performance in real-world datasets of multiple domains.

2.2 Related work

2.2.1 Object counting

Early object-counting algorithms mainly targeted crowd counting. They applied detection-based approaches such as Regions with CNN Features (R-CNN) [1] and You Only Look Once (YOLO) [42] to estimate the number of people in an image and demonstrate satisfactory accuracy in sparse scenes. However, the performance dropped on densely crowded scenes where people overlapped with heavy occlusion. Alternative regression-based methods [43] can extract features (textures, gradients, shapes) to overcome occlusion and learn a mapping function to evaluate how sparse the scene is, but they ignore the spatial information. In general, CNN-based approaches predict density maps to estimate the number of instances in the scene and use the spatial information contained in the density map [44]. Currently, most of the object counting state-of-the-art algorithms are based on fully convolutional networks [45] combined with other techniques such as analising the context [46], using the perspective information [47] or applying a multi-column architecture [38].

Later in this chapter, the counting performance is compared with state-of-the-art algorithms. In the context of crowd counting, PGCNet [48] uses perspective-guided convolution to better understand the spatial information of the scene. Instead of applying a multi-scale architecture, they present a single scale that understands the relationship between the camera and the crowd.

On the other hand, SANet [49] obtains the best results on the crowd counting datasets with less density by combining multi-scale features with a local consistency loss that enforces local structure limiting the pixel-wise error.

In the context of counting penguins, the best results are obtained by Marsden et al. [50]. They train a single model that learns counting across multiple modalities including people, penguins and cells. Using transfer learning, they enable cross-domain features that can extrapolate to all domains.

Finally, FCN-rLSTM[51] obtains the best vehicle counting results by learning

spatiotemporal features from multiple frames of a video. This method is very efficient on low-resolution videos where there is high occlusion and large perspective.

2.2.2 Object counting datasets

Many object-counting datasets have appeared in recent years [38, 40, 39, 52], especially for crowd counting. In general, they are annotated with dots indicating the position of the objects, e.g. crowd counting datasets define a dot on the head of each person (see Figure 2.2). The annotation of object counting datasets is expensive because it requires precise dot annotations performed by an expert, hence the datasets tend to be small, as shown in Table 2.1.

SHT A [38] is a crowd-counting dataset collected from internet images. It is composed of city scenes where each person in the crowd is annotated with one point on the centre of the head. Similarly, SHT B[38] contains city images exclusively from Shanghai annotated in the same manner.

The Counting in the Wild dataset [40] is composed of images from Anatartica displaying penguins. The dataset has been annotated through crowd-sourcing by tens of thousands of volunteers. The dataset shows a high degree of occlusion and scale variation.

TRANCOS [39] is a vehicle counting dataset recorded by surveillance cameras in Spain. The dataset shows traffic congestion situations with extreme overlapping. The dataset also includes a video version where each frame is annotated.

MinneApple [52] is a dataset for apple detection and segmentation. The data has been collected by the Horticultural Research Center at the University of Minnesota in the United States. The images are extracted from a video captured with a phone in the orchard. The camera moved at a slow speed to avoid motion blur.

Figure 2.2: Annotated sample from crowd counting JHU-CROWD++ [53] dataset.

Dataset	Images	Resolution	Avg. Count
SHT A (people) [38]	482	589 × 868	501.4
SHT B (people) [38]	716	768 × 1024	123.6
Penguins (penguins) [40]	80,095	2048 × 1536	7.2
TRANCOS (vehicles) [39]	1,244	640 × 480	36.5
MinneApple (apples) [52]	670	720 × 1280	42.1

Table 2.1: Details of the real-world datasets used to evaluate the proposed object counting method

2.3 Domain Randomisation for Object Counting

The objective of DR is to generate enough variations of synthetic data that the model views real data as just another variation, even if the variations used for training appear unrealistic to humans. Expanding the spectrum of possibilities also raises the complexity of the task, requiring a model with a higher capacity. If the model is trained on a sufficient number of environments, it will interpolate well to novel ones. This method can be considered as an evolved form of data augmentation.

This section presents the methodology for the generation of datasets utilising DR techniques and the training of a model trained on this dataset for object counting.

2.3.1 Scene creation

The DR datasets used for the experiments are generated using a mixture of 3D models, textures, background images, and lighting effects. The 3D software to render the scenes is Blender, which can be easily automated. 3D models used are low-poly models, i.e. they have less than 200 faces, as shown in Figure 2.5. Using highly realistic models, which can have thousands of faces, does not improve the results while significantly increasing the rendering time. Low-poly 3D models are used because they have a relatively smaller number of polygons and decrease the rendering time. For instance, rendering a 3D scene with 500 models with 200 faces each, takes 2 seconds. On the other hand, rendering 500 models with a higher quality (20,000 faces each) takes around 3 hours.

In terms of performance, a comparison between the low-poly dataset from Figure 2.1 and a high-poly dataset is made. The high-poly dataset is generated with MakeHuman [54] software (see Figure 2.3). The software allows the creation of high-quality 3D models with customized clothes and body styles (gender, height, skin colour, hair). An experiment shows that, although the high-poly images look more realistic, they have a counting performance of 23.5 MAE compared to 23.2 MAE with low-poly images on the SHT B [38] dataset. Figure 2.4 displays a sample from the high-poly dataset, which took 94 hours to generate to generate 10,000 images with around 20,000 faces.

In crowd counting datasets, persons that are occluded are omitted from the ground truth, or more precisely, datasets designated for crowd counting exclude individuals whose heads are not discernible. To replicate this, the dataset generation algorithm evaluates when a particular vertex on the forehead is within the camera's field of view and is not occluded by other objects. In the synthetic ground truth, only those objects with the specified vertex in the visible range are included in the count. For the other domains (penguins, vehicles, and apples) all the models included in the scene are considered in the synthetic ground truth, even if they are completely occluded. This hardly occurs because the number of objects is low in these contexts

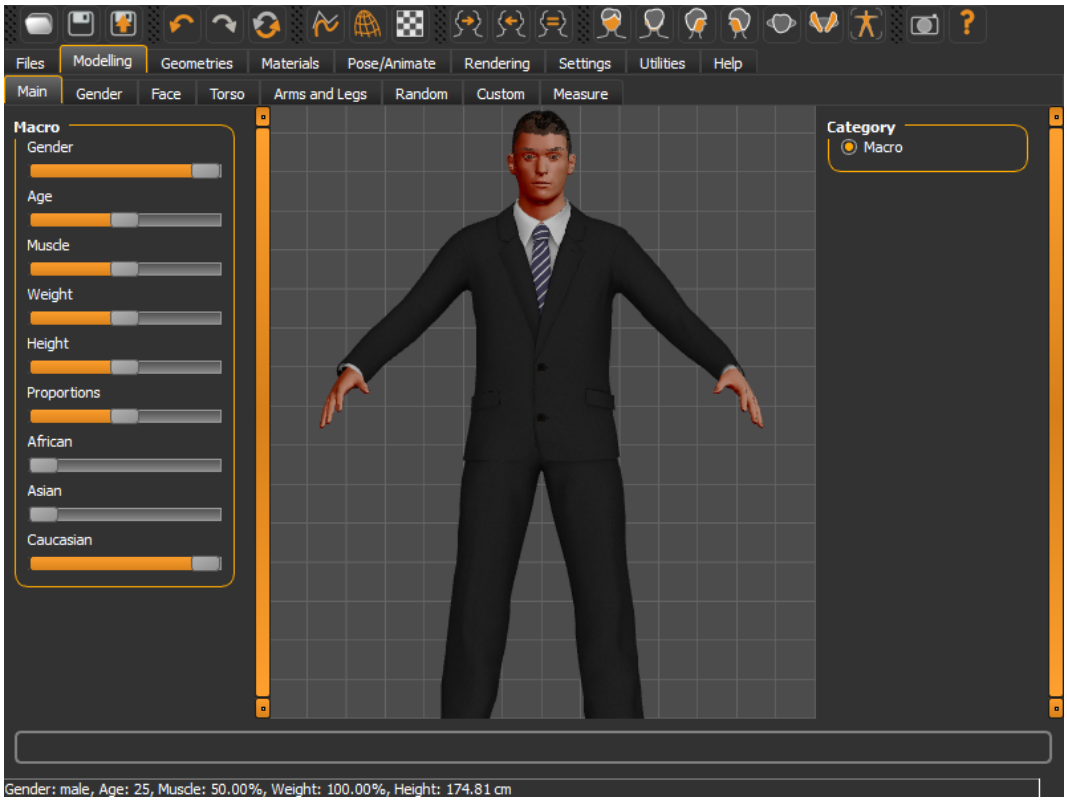


Figure 2.3: MakeHuman [54] user interface to generate high-poly models.

compared to crowd counting (see Table 2.1). On 50 samples from the penguins and vehicles synthetic datasets, only an average of 1.4 objects are occluded.

The low-poly 3D model's structure is modified to produce more variations, some of them unrealistic. The generated structures are, however, constrained to keep the basic shape, e.g. humans with one head and two legs, otherwise the model will not learn the inner properties of the object. Learning a vast amount of shapes improves generalisation to novel scenarios. Figure 2.8 shows the different 3D transformations used to produce the synthetic datasets: scale, randomisation, and extrusion. Scale smoothly expands/contracts all the vertices on the same axis. It is useful when objects tend to have very different sizes, e.g. adults tend to be twice as big as children.

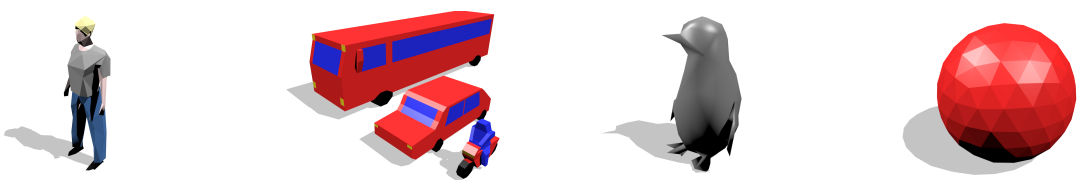
The scale of every 3D model is determined by $K \sim \mathcal{U}(1/N_o, 8/N_o)$ where $\mathcal{U}$ is a uniform distribution and N_o is the number of objects in the image. Randomising the vertices of the mesh translates all the vertices in different lengths and directions, uniformly by a factor of 40%. This method improves the performance in environments where the pose of the objects is variable, e.g. people can have multiple poses while

Figure 2.4: Sample image with high-poly models generated with MakeHuman [54].

vehicles do not. Extrusion alters the surfaces of the mesh to increase the thickness by adding depth. This helps to make objects bigger or smaller by adding bumps and holes. We used the built-in Solidify transform in Blender and modified the thickness by T where $T \sim \mathcal{U}(-0.1, 0.5)$.

Textures from the "Describable Textures Dataset" [55] are applied on the 3D models as shown in Figure 2.6. The dataset contains 5640 textures organised in 47 categories. Textures are mapped to the different parts of the 3D models, e.g. hair, skin, shirt, pants. This technique helps DR to transcend realism by producing unrealistic sets of randomly textured 3D models.

The 3D models are placed in the scene by sampling positions from a standard



(a) Human (b) Vehicles (c) Penguin (d) Apple

Figure 2.5: Low-poly 3D models used to create synthetic datasets

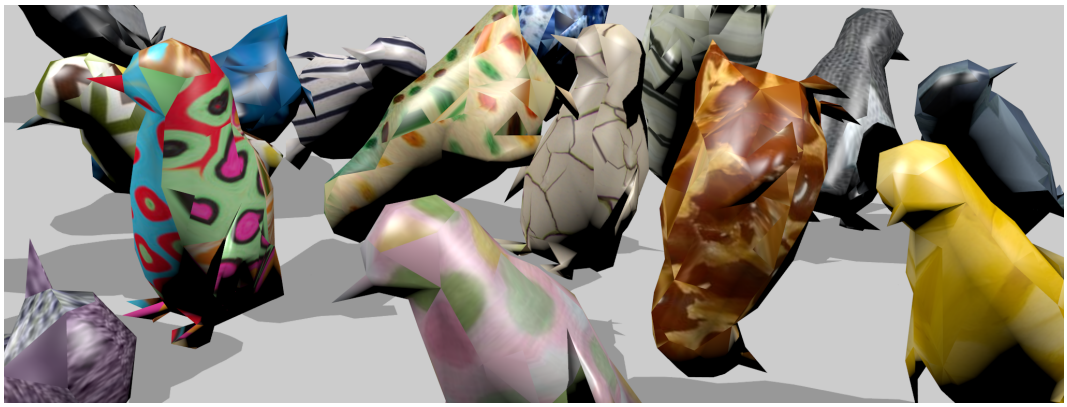


Figure 2.6: Random textures from the Describable Textures Dataset applied on a low-poly 3D model of a penguin

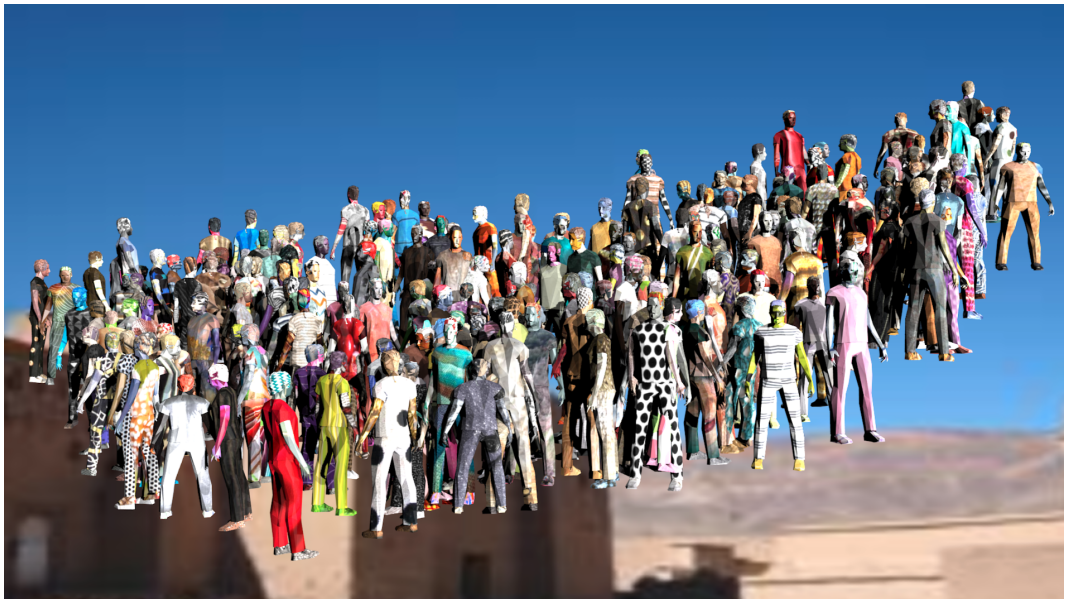


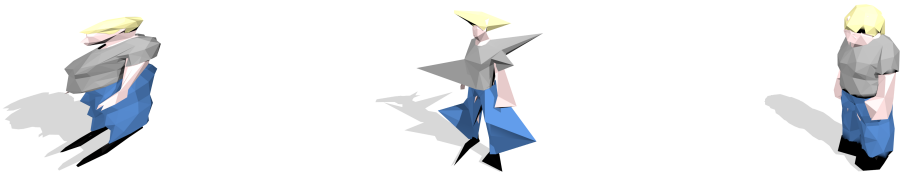
Figure 2.7: Synthetic crowd counting sample with random textures.

Gaussian mixture distribution as follows:

$$p(\mathbf{x}) = \sum_{i=1}^{K} \lambda_i \mathcal{N}(\mathbf{x} \mid \mu_i, \Sigma_i), \tag{2.1}$$

where $\mathbf{x}$ is the three-dimensional x, y, z position, λ_i are the mixture component weights, μ_i are the means, and $\Sigma_i = I$. The number of components K is sampled for each scene as $K \sim \mathcal{U}(1 + N_o/20, 2 + N_o/8)$ where $\mathcal{U}$ is a uniform distribution and N_o is the number of objects in the scene. The mean vectors are uniformly sampled from the visible region. Figure 2.7 shows a synthetic scene of people in a single cluster.

This method creates occlusion in the clusters but also produces large empty areas



(a) Scale (b) Randomisation (c) Extrude

Figure 2.8: 3D transformations used for domain randomisation.

where the background image is displayed. It also mimics how objects are distributed on the real world, e.g. people are not uniformly distributed, they tend to form small groups on the street[56]. Note that when the objects are distributed uniformly the test Mean Absolute Error (MAE, see Section 2.5 for details) rises to 63.4 on the SHT B dataset for crowd counting, compared with 23.2 using the Gaussian mixture approach.

Images from the Places2 dataset [57] are used as the background image. The dataset contains a wide range of scenes from 365 different environments including indoors, streets and nature. The fact that the background images are very different makes the task more complex but improves generalisation. Depending on the task, some image categories have been removed to avoid unlabeled instances of the relevant objects in the background, e.g. the "stadium-football" category when counting people or the "iceberg" category when counting penguins. Note that these efforts were not exhaustive and some conflicts between backgrounds and annotations may remain. Finally, a combination of coloured lights is randomly placed around the scene to produce different exposure levels and cast shadows around the 3D objects.

2.4 Methodology

2.4.1 Setup

Experiments are conducted utilising the standard train/test splits for each dataset, as described in Section 2.2.1. These specified splits correspond with those utilised in the official benchmarks. Contrary to the conventional approach of utilising the

training set, synthetic data and annotations are employed in this research. Note that the test set comprises data from the real-world domain, posing a challenge in generalising from a synthetic to a real domain. Outcomes from the experiments are compared with state-of-the-art methodologies as outlined in Section 2.2.1. This serves as a robust comparative framework for evaluating the model. The hardware infrastructure employed for these experiments comprised an NVIDIA RTX 2080 Ti graphics processing unit, equipped with 11GB of memory. Hyperparameters for the experiments are adopted from a publicly accessible baseline implementation, which can be located at the authors' online repository [58]. It should be noted that the batch size is modified from 16 to 8 due to constraints related to graphics card memory.

2.4.2 Model

The Distribution Matching for Crowd Counting [58] approach is used as a baseline. The authors of the paper use Optimal Transport to measure the similarity between the normalised predicted density map and the normalised ground truth density map. They also include a total variation loss to force the neighbouring pixels to have similar values. The baseline performance is particularly good in scenes where the density and overlapping of the objects are high. A ResNet50[15] model, pre-trained on ImageNet Large Scale Visual Recognition Challenge, is used as the base model for all of the counting tasks. It is observed that smaller versions of ResNet, such as ResNet18, suffice for real data. However, for the randomised synthetic dataset, ResNet50 improves the MAE for crowd counting by 17%. The hypothesis posits that a more complex model architecture, in this case, ResNet50, is required owing to the need for learning a broader range of variations. This complexity presumably requires a larger model capacity. Horizontal flips are applied to duplicate the amount of available synthetic images. In addition, training images are randomly cropped into multiple smaller images (512×512) to obtain more samples.

2.5 Experiments

In this section, the metrics for object counting are presented, and subsequently, the performance of DR for object counting is evaluated by testing models trained solely with synthetic data on real-world datasets. The experiments conducted include assessing the performance of the models on real-world datasets and investigating the impact of 3D transformations.

2.5.1 Metrics

Performance in object counting is measured using two main metrics: MAE (Mean Absolute Error) and MSE (Mean Squared Error). They compute the average L1 and L2 distance between the predicted count and the ground truth respectively. MAE and MSE are scale-dependent and therefore can not be used to make comparisons between datasets using different scales. The formula used to compute them is described as follows:

$$MAE = \sum_{i=1}^{n} \frac{|\mathbf{x}_i - \mathbf{y}_i|}{n} \tag{2.2}$$

$$MSE = \sum_{i=1}^{n} \frac{(\mathbf{x}_i - \mathbf{y}_i)^2}{n} \tag{2.3}$$

where x is the predicted count, y is the groundtruth and n is the number of images evaluated. The main difference between both metrics is that MAE is more robust against outliers whilst large errors have a greater effect on the MSE score.

2.5.2 Comparison with real-world datasets

Table 2.2 compares the performance of training with real data and synthetic data.

Results manifest a lower performance when compared to models trained on real data and annotations. The MAE observed on the SHT A dataset stands out as notably elevated. This can be attributed to the intrinsic complexities of the dataset,

Dataset	Real	DR
SHT A [38]	PGCNet [48]: 57.0	158.7
SHT B [38]	SANet [49]: 6.5	23.2
Penguins [40]	Marsden et al. [50]: 5.8	14.6
TRANCOS [39]	FCN-rLSTM [51]: 4.2	13.6

Table 2.2: Object counting performance (MAE) of real and DR synthetic data on multiple real-world domains.

which has a substantial average count of 501 individuals per image. Contrastingly, the outcomes on the TRANCOS dataset appear more promising. Given the dataset's average count of 36.5 vehicles, an error magnitude of 13.6, in the absence of any prior information, merits recognition as a significant accomplishment. It is postulated that the distinctive shape of cars when compared with entities such as humans or penguins, may account for this relative success. Regarding the MinneApple dataset, there are no publications that evaluate its performance specifically in relation to counting tasks.

Table 2.3 compares the current DR approach with Wang et al. [8] for crowd counting. Their method is based on DA applied on images from a realistic video game. Using real-world images to feed a GAN they improve the textures of the video game images. DA is successfully applied to domains where it is easy to obtain real-world images and produce synthetic data using a video game, e.g. urban environments involving people and vehicles. The current DR approach obtains similar results without using real-world data and very simple rendering techniques. Whilst the approach of [8] performs better than the current approach, it should be noted that our performance is achieved with a synthetic dataset that was generated completely automatically without the need for 3D artists.

Dataset	GCC [8]		Proposed approach	
	MAE	MSE	MAE	MSE
SHT A [38]	123.4	192.4	158.7	253.2
SHT B [38]	19.9	28.3	23.2	41.5

Table 2.3: Crowd counting performance using synthetic data comparison.

3D transformation	Scale		Extrude		Randomise	
	MAE	MSE	MAE	MSE	MAE	MSE
SHT A [38]	183.97	291.7	159.0	253.2	**158.7**	**253.2**
SHT B [38]	29.17	47.8	30.2	54.2	**23.2**	**41.5**
Penguins [40]	18.9	25.1	**14.6**	**20.1**	14.6	20.4
TRANCOS [39]	15.4	19.3	**13.6**	**17.3**	13.6	17.6
MinneApple [52]	22.4	29.4	21.3	27.5	**17.5**	**22.3**

Table 2.4: Domain randomisation techniques performance in MAE and MSE. Results in bold indicate best performance.

2.5.3 3D transformations analysis

3D transformations increase the variability of the dataset in terms of shape, improving the generalisation on novel domains. Table 2.4 shows how 3D transformations affect performance of the object counting task. For each experiment 2,000 synthetic images with the given 3D transformation are generated.

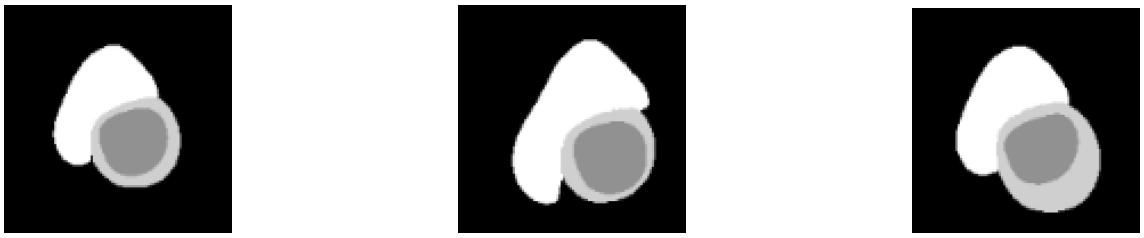
Also, note that when applying strong 3D transformations the training process takes longer because the task becomes more complex.

Overall, 3D transformation represent a relative improvement of 18% across all five datasets. The impact of 3D transformations depends on the nature of the object, e.g. variable pose and size. Randomising the vertices works better on environments with objects that can present different poses, e.g. people. The results obtained with the extrude transform are similar to the randomise ones because it also creates small irregularities in the shape, which improved the penguins counting performance by 22%. Extrude also exhibits good performance in environments where the objects are solid. In the case of vehicle counting, it improved the performance by 15%.

2.6 Conclusions

We can conclude that domain randomisation allows training an object counting model when no real-world data is available.

Table 2.2 shows that the performance of state-of-the-art models trained with real-world data is far from the purely synthetic proposed approach.



(a) Healthy hearth (b) Afflicted right ventricle (c) afflicted left ventricle

Figure 2.9: Samples from M&Ms cardiac segmentation dataset [59].

However, in evaluating the proposed approach against other models that also utilize synthetic data, the performance gap between the proposed approach and state-of-the-art algorithms trained on synthetic data is not significantly large. For instance, in Table 2.3, the GCC [8] model achieved a 19.9 MAE in the SHT B dataset [38], whereas the proposed method recorded 23.2 MAE.

Increasing the capacity of the model (Section 2.4.2), randomising the position of the objects accordingly (Section 2.3.1), and applying the right 3D transformations to the meshes (Section 2.4) have been found to be an effective solution addressing RQ1.

It is important to note that there is still significant potential for enhancement and improvement for domain randomisation. As future work, it could be interesting to investigate the application of domain randomisation to other tasks, particularly those with high noise and variance, such as people and vehicle segmentation. The technique's ability to increase the diversity of synthetic images, making the model more robust to real-world variations, could prove to be beneficial in these tasks.

However, domain randomisation is not always the most effective approach for all computer vision tasks, and there are some situations where it may not be useful at all. The literature indicates a strong inclination towards domain adaptation [60]. Despite the prevalence of studies focusing on domain adaptation, domain randomisation still has a small presence within the research area [61]. One example of a task where domain randomisation might not be effective is when the input data has a small variance. For example, if the task is to classify images of a single, well-defined object with a consistent colour and shape, then introducing artificial perturbations to the data may not provide any additional benefit. In this case, it is hypothesised that

the model may already be able to generalise to new, unperturbed examples of the same object, and adding more variability to the training data may actually make the task more difficult. Another example of a task where domain randomisation may not be useful is when the performance of the model is highly sensitive to the specific characteristics of the input data. There is a lack of empirical studies investigating the efficacy of domain randomisation in medical imaging. The hypothesis here is that medical imaging datasets might perform poorly when using domain randomisation. For instance, medical imaging datasets perform poorly when using domain randomisation because they are small and very sensitive to augmentations. Figure 2.9 illustrates this with an example of a ventricular segmentation ground truth for cardiac magnetic resonance images. This example highlights the inability of domain randomisation to tackle some tasks where the difference between a healthy heart and an afflicted heart is very subtle. Changing the textures or the shape of an image of this nature will inevitably distort the inner properties of some images, and may (inadvertently) change the classification of a cardiac image from healthy to diseased.

Chapter 3

Domain adaptation

In this chapter, Domain Adaptation (DA) is introduced as an alternative to domain randomisation. Domain adaptation is utilised as a preamble to the examination of Research Question 2 (RQ2) in the subsequent chapter. DA is particularly useful in situations where domain randomisation is not effective, such as when working with small datasets with specific characteristics, such as polyp segmentation in medical imaging. This chapter leverages domain adaptation techniques for polyp segmentation by producing realistic synthetic images using a combination of 3D technologies and generative adversarial networks. No annotations from medical professionals are used in the process because the method is fully unsupervised, achieving promising results on five real polyp segmentation datasets. In addition, this chapter introduces Synth-Colon, an entirely synthetic and freely available dataset that includes 19,917 realistic colon images and additional details about depth and 3D geometry. The work reported in this chapter has been published in the 29$^{\text{th}}$ Irish Conference on Artificial Intelligence and Cognitive Science (AICS) [62] and the code to replicate the experiments is available in the online repository of the paper.

3.1 Definitions

In this section, important definitions for the following chapters are provided.

Ground truth: Ground truth refers to the baseline or "true" data that serves

as a standard for comparison. This baseline data is associated with specific samples in a dataset and is used to train models or evaluate the accuracy and reliability of algorithms. Groundtruth data falls into two categories:

Real Labels: These are annotations performed manually by human experts. Real labels are often considered the gold standard because they incorporate human understanding, although they may be susceptible to human error and biases.

Synthetic Labels: In contrast, synthetic ground truth is generated algorithmically, without human intervention. For example, a computer program may automatically produce an image and its corresponding label. Synthetic ground truth can be produced at scale and is highly accurate.

Supervised Learning: Supervised learning refers to a class of machine learning algorithms that are trained using ground truth. Each input sample in the training dataset is accompanied by a corresponding output label, known as the annotation. During the training phase, the algorithm attempts to learn the mapping function from the input to the output. The objective is to make accurate predictions or decisions without human intervention. Supervised learning is commonly used in applications such as image recognition, email filtering, and predictive analytics.

Unsupervised Learning: Unsupervised learning refers to machine learning algorithms that operate without the need for labelled ground truth during the training phase. These algorithms seek to identify inherent structures, patterns, or features in the data, such as clusters or associations. Importantly, in the context of this thesis, the term "unsupervised learning" also includes scenarios where synthetic labels are used for training purposes. Such cases can also be described as **"synthetically supervised"**, acknowledging that no real, human-annotated ground truth is involved.

Semisupervised Learning: Semisupervised learning involves scenarios where the training dataset contains both labelled and unlabeled data. While the majority of data may be unlabeled, a smaller subset of labelled data is also included to guide the learning process. This approach combines elements of both supervised and unsupervised learning methods, aiming to use the strengths of each. Semi-supervised

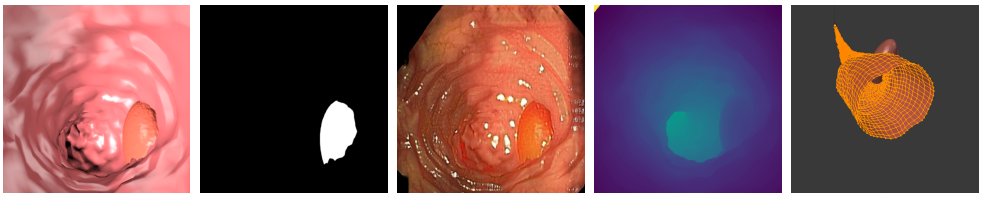


Figure 3.1: Synth-Colon dataset [62] samples include synthetic image, annotation, realistic image, depth map, and 3D mesh (from left to right).

learning is particularly useful in situations where annotating a fully labelled dataset is expensive or time-consuming, but a smaller set of labelled examples can still provide valuable guidance to the model.

3.2 Introduction

Colorectal cancer is the third most commonly diagnosed cancer type worldwide [63], representing approximately 10% of all cancer cases. This is in contrast to breast and lung cancers, which each account for about 12% of global cancer diagnoses, out of the total cancer cases globally. It can be treated with an early intervention which consists of detecting and removing polyps in the colon. The accuracy of the procedure strongly depends on the medical professional's experience and hand-eye coordination during the procedure, which can last up to 60 minutes. Computer vision can provide real-time support for doctors to ensure a reliable examination by double-checking all the tissues during the colonoscopy. This is usually done by highlighting the pixels around the areas of the video that are likely to represent a polyp, helping the doctor to spot hidden polyps during the procedure.

The data obtained during a colonoscopy is accompanied by a set of issues that prevent the creation of datasets for developing computer vision algorithms. First, there are privacy issues because it is considered personal data that can not be used without the consent of the patients. Second, there is a wide range of cameras and lights used to perform colonoscopies. Every device has its own focal length, aperture, and resolution. There are no large datasets with standardised parameters. Ultimately, polyp segmentation datasets are difficult to create because they depend

on the annotations of qualified professionals.

A synthetically supervised method to detect polyps is proposed in this chapter. The method does not require annotations and combines 3D rendering and a CycleGAN [35]. First, artificial colons and polyps are produced based on a set of parameters. Annotations of the location of the polyps are automatically generated by the 3D engine. Second, the synthetic images are used alongside real images to train a CycleGAN. The CycleGAN is used to make the synthetic images more realistic. Finally, a HarDNeT-based model [64] is trained with realistic synthetic data and self-generated synthetic labels.

The main benefit of the proposed method is that it does not require annotations from the real world. Additionally, the Synth-Colon dataset (see Figure 3.1) is publicly released. It is the largest synthetic dataset for polyp segmentation including additional data such as depth and 3D mesh.

While this thesis studies an approach for use with conventional colonoscopy, it is intended as a demonstrator of how a deep learning medical imaging system for detecting a new type of disease could be quickly put in place with little or no training data. As such, this project in its current form is not intended for deployment in clinical settings. It is also worth noting that there is a critical caveat that undermines the real-world applicability of the proposed polyp segmentation model: the composition of the benchmark datasets used to train it are very different from real-world colonoscopy images. These datasets predominantly feature images containing only one polyp and lack instances of either multiple polyps in a single image or images with no polyps. Such unrepresentative datasets produce biased models that exhibit low performance when applied to actual clinical scenarios. Specifically, a model trained solely on these datasets is predisposed to identifying a polyp in every frame it processes, because it has never been exposed to images without polyps. This bias would produce a high number of false positives, making it unusable. Even if said model exhibits good performance metrics on the benchmark datasets, its clinical applicability remains compromised due to this high incidence of false positives.

3.3 Related work

3.3.1 Colonoscopy datasets

In recent years, a variety of datasets have been introduced to aid the advancement of colorectal polyp segmentation. Figure 3.2 showcases the five datasets selected for algorithmic benchmarking in this thesis. These datasets are chosen not only because they are commonly utilised in state-of-the-art benchmarks but also because they offer a comprehensive range of cameras, lighting conditions, and textures. Next, the key points of each dataset are discussed:

- **CVC-ClinicDB** [65]: Released by the Computer Vision Center at the Universitat Autònoma de Barcelona, this dataset provides colonoscopy video frames tailored for colorectal cancer diagnostic assistance. Each image is paired with an annotated mask that highlights a polyp. One of the challenges of this dataset is the variability of lighting and the presence of artefacts.

- **CVC-T** [66]: Another contribution by the Computer Vision Center, CVC-Texture contains polyp and background texture patches intended for texture classification tasks. The dataset addresses the challenge of discriminating between polyp textures and normal textures but also includes detailed segmentation masks.

- **CVC-ColonDB** [67]: Released by the same entity, CVC-ColonDB is designed specifically for the detection and segmentation of polyps in colonoscopy images.

- **ETIS-LaribPolypDB** [68]: This dataset, curated by the ETIS Laboratory, offers colonoscopy images featuring various sizes and types of polyps. A distinctive challenge in this dataset is the presence of small polyps which may be hard to detect.

- **Kvasir** [69]: Introduced by a consortium of researchers, the Kvasir dataset is the largest and most used benchmark for polyp segmentation. The dataset

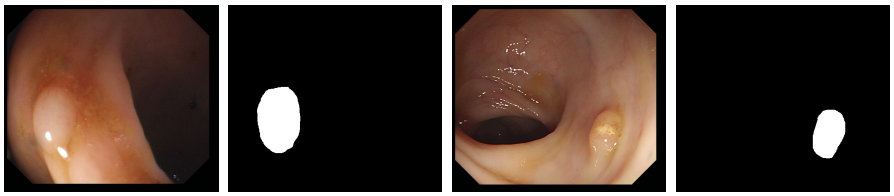


Figure 3.2: Real samples from CVC-ColonDB [67] with the corresponding annotation made by a medical professional indicating the location of cancerous polyps.

stands out due to its broad range of conditions and related challenges, such as differentiating between subtle abnormalities.

3.3.2 Polyp segmentation

Early approaches to polyp segmentation were based on the texture and shape of the polyps. Hwang et al. [70] used ellipse fitting techniques based on shape. However, colorectal polyps tend to be small and are not detected by these techniques. The similarities between the polyp texture and the background texture may cause complications in the detection process (see Figure 3.2).

The advent of convolutional neural networks [71] allowed better characterisation of the features of polyps and segmentation accuracy was substantially increased. Several authors have applied deep convolutional networks to the polyp segmentation problem. Brandao et al. [72] used a fully convolutional neural network based on the VGG [73] architecture to identify and segment polyps. Unfortunately, the small datasets available and the large number of parameters make large networks prone to overfitting. Zhou et al. [74] used an encoder-decoder network with dense skip pathways between layers that prevented the vanishing gradient problem of VGG networks. They also significantly reduced the number of parameters, reducing the amount of overfitting. More recently, Chao et al. [64] reduced the number of shortcut connections in the network to speed up inference time, a critical issue when performing real-time colonoscopies in high-resolution. They focus on reducing the memory traffic to access intermediate features, reducing the latency.

Later in this chapter, the proposed polyp segmentation technique is benchmarked against four previously published segmentation models that use different architectures.

First, Ronneberger et al.[75] use the U-Net network architecture to predict the pixels associated with the polyps. The model consists of an encoder that contracts followed by a symmetric interconnected decoder that expands, allowing precise localisation. The encoder has skip connections that transfer context information across layers. Second, Fang et al. [76] propose Selective Feature Aggregation that trains an encoder and two decoders. The decoders predict polyps areas and boundaries, improving the detection of polyps that have a texture similar to the colon. Convolutional layers include kernels of multiple sizes that extract features, which helps in identifying polyps that are smaller than usual. Third, PraNet[77] uses an attention mechanism to detect areas that are likely to contain a polyp and focus on them. It also uses multiple decoders in parallel. Finally, HardNet-MSEG[78] uses a low-traffic architecture. The solution proposed in this chapter is based on this model. More details are explained in Section 3.4.2.

3.3.3 Synthetic data for polyp segmentation

The basis of this thesis is addressing a fundamental limitation of using large neural networks: they require large amounts of annotated data. This problem is particularly acute in medical imaging due to problems in privacy, standardisation, and the lack of professional annotators. Table 3.1 shows the limited size and resolution of the datasets used to train and evaluate existing polyp segmentation models. The lack of large datasets for polyp segmentation can be addressed by generating synthetic data.

Thambawita et al. [79] use a generative adversarial network to produce new colonoscopy images and annotations. They added a fourth channel to SinGAN [80] to generate annotations that are consistent with the colon image. They then use style transfer to improve the realism of the textures. Their results are excellent considering the small quantity of real images and professional annotations that are used. Gao et al. [81] use a CycleGAN to translate colonoscopy images to polyp masks. The generator learns how to segment polyps by trying to fool a discriminator.

Synthetic images combined with generative networks have also been widely used

Table 3.1: Real polyp segmentation datasets size and resolution.

Dataset	#Images	Resolution
CVC-T [66]	912	574 x 500
CVC-ClinicDB [65]	612	384 x 288
CVC-ColonDB [67]	380	574 x 500
ETIS-LaribPolypDB [68]	196	1225 x 966
Kvasir [69]	1,000	Variable

in the depth prediction task [82, 83]. This task helps doctors to verify that all the surfaces in the colon have been analysed. Synthetic data is essential for this task because of the difficulties of obtaining depth information in a real colonoscopy.

Unlike previous works, the DA method described here is entirely unsupervised and does not require any human annotations. It automatically generates the annotations by defining the structure of the colon and polyps and transferring the location of the polyps to a 2D mask. The key difference between this approach and other state-of-the-art is the combination of 3D rendering and generative networks. First, the 3D engine defines the structure of the image and generates the annotations. Second, the adversarial network makes the images realistic.

Similar unsupervised methods have also been successfully applied in other domains like crowd counting. For example, Wang et al. [8] render crowd images from a video game and then use a CycleGAN to increase the realism.

3.4 Domain Adaptation for polyp segmentation

The proposed approach is composed of three steps: first, procedurally generate colon images and annotations using a 3D engine; second, feed a CycleGAN with images from real colonoscopies and synthetic images; finally, use the realistic images created by CycleGAN to train an image segmentation model.

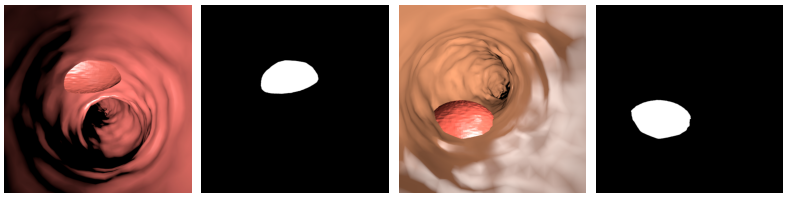


Figure 3.3: Synthetic colons with corresponding annotations rendered using a 3D engine.

3.4.1 CycleGAN

A standard CycleGAN comprises of two generators and two discriminators and is trained using real images from colonoscopies and synthetic images generated using the 3D engine as depicted in Figure 3.5.

CycleGAN is a variant of adversarial neural networks that comprises two generators and two discriminators. Each generator and discriminator duo is tasked with learning and establishing a distinct mapping between two separate domains. Generators undertake the role of translating images from one domain to the other. For instance, one generator might convert photographs into paintings, while its counterpart reverses this process. On the other hand, the discriminators evaluate these translations by differentiating between authentic and generated images within each domain. Their function is crucial for refining the performance of the generators, as they work in tandem to produce increasingly convincing translations. The key to CycleGAN is the concept of cycle consistency. Cycle consistency ensures that an image, once translated from one domain to another and back again, should closely resemble the original image. This concept is essential for the robustness of the model, as it minimises the risk of information loss during the translation process.

A key advantage of CycleGAN is its capacity for unpaired image-to-image translation. This means that images from the two domains do not have to share a direct correspondence or feature items in identical positions. This characteristic makes the model particularly adaptable for a broad array of applications. However, CycleGAN has some drawbacks. First, the model is memory-intensive, requiring significant computational resources for training four models (two generators and two discriminators). Second, achieving convergence is challenging because of the interactions between four

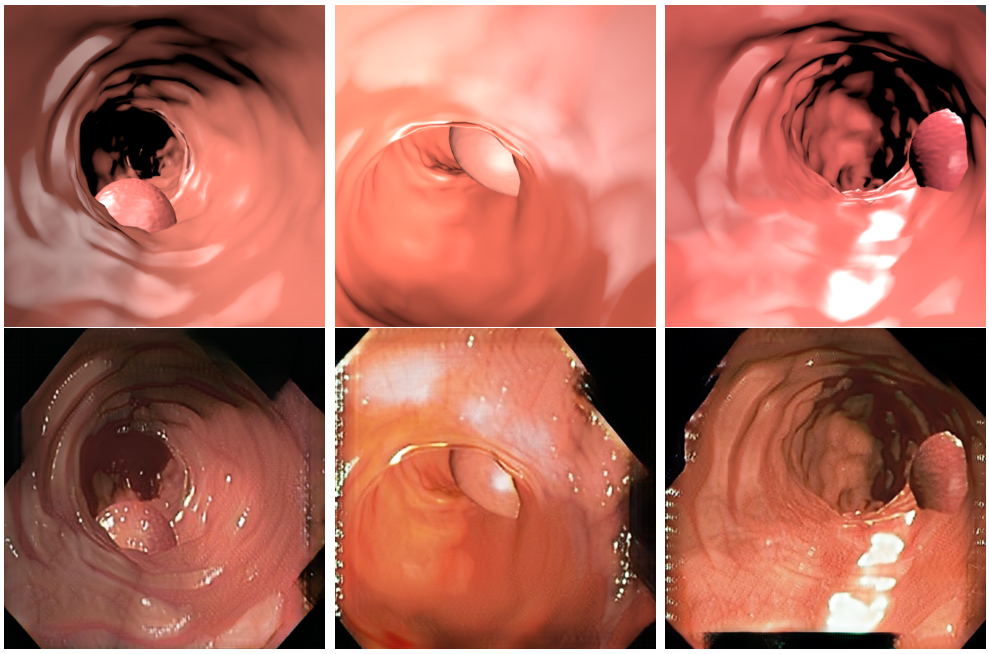


Figure 3.4: Synthetic images (first row) and realistic images generated by CycleGAN (second row).

different models. The models must reach a stable state while continually adapting to each other, a process that can be computationally expensive.

Figure 3.4 displays synthetic images before and after the CycleGAN domain adaptation. Note that the position of the polyps is not altered. Hence, the ground truth information generated by the 3D engine is preserved.

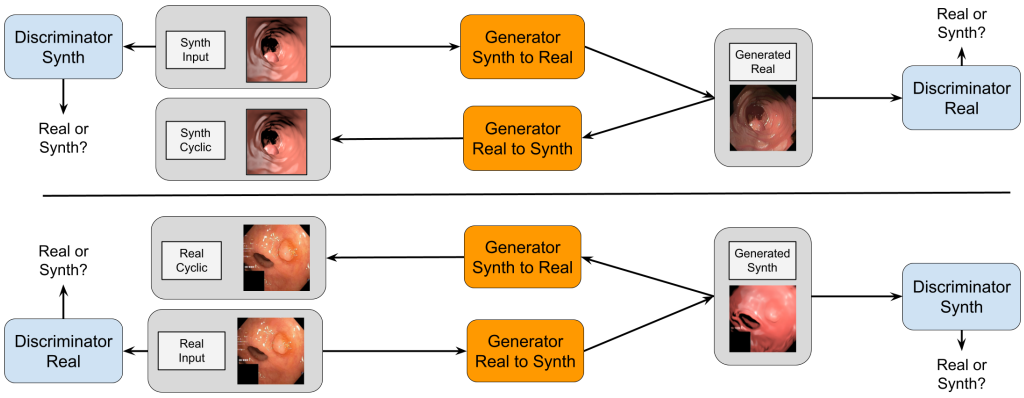


Figure 3.5: CycleGAN architecture. Two generator models are trained to fool two discriminator models by changing the domain of the images. Real images are passed to the "Generator Synth to Real" model, producing realistic colon images.

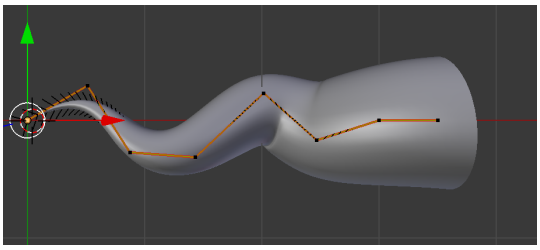


Figure 3.6: The structure of the colon is composed of 7 segments to simulate the curvature of the intestinal tract.

3.4.2 Polyp segmentation

After creating a synthetic dataset that has been adapted to the real colon textures, an image segmentation model is trained. A HarDNeT-MSEG [78] model architecture is used for segmenting the colonoscopy images . The architecture employs a simplified encoder-decoder framework based on HarDNet [64], a low-memory traffic CNN, with a Cascaded Partial Decoder, known for its efficiency in salient object detection. Low traffic refers to the small amount of data that is read or written to memory during the forward and backward pass. This significantly decreases the time it requires to infer predictions from an image, allowing the algorithm to work on real-time videos. The key to reducing the memory resources is using fewer parameters and sharing weights across layers. The Cascaded Partial Decoder implies that multiple steps process the data partially. Each step incrementally refines the output of the decoder, which is more efficient than a single step where the entire output must be generated all at once. This staged approach allows for intermediate corrections and adjustments, potentially reducing the computational load and making the process more accurate. The hyperparameter configuration is the same as in the original paper.

3.5 Methodology

3.5.1 3D colon generation

As in Chapter 2, Blender is used to create 3D-based synthetic data. The 3D colon structure is a cone composed of 2,454 faces. It is important to add a high number of faces in order to mimic the irregular geometry of the colon. Vertices are randomly displaced following a normal distribution in order to simulate the tissues in the colon. Additionally, the colon structure is modified by displacing seven segments as in Figure 3.6. A base colour [0.80, 0.13, 0.18] (RGB) is used for the textures, which is the average colour of the Kvasir dataset. For each sample, the colour is shifted randomly within a range of 40% for each channel value to other tones. Similar to the existing polyp datasets, there is only one polyp present in every image, which is placed inside the colon. The polyp may either adhere to the walls of the colon or remain suspended in the centre of the colon, without making contact with any surrounding surfaces. Polyps are distorted spheres with 128^2 faces to ensure that the surface is smooth, resembling real polyps. A uniform distribution with a range of 120% is used to define the scale of the polyps along each of the three axes. The meshes corresponding to the scaled polyps models are then distorted using the "Randomize" tool in Blender with a value of 2.0. The real size of the polyps is unknown since they are created in a virtual environment. However, their average area relative to the overall image area is similar to the Kvasir image to polyp ratio, ensuring a realistic range. Samples with polyps occupying less than 20,000 pixels are removed because there are no polyps with this size in the real-world datasets examined in this thesis.

The spherical morphology of the generated polyps allows them to represent various polyp types (sessile or pedunculated). The polyp type is defined by the proximity between the sphere and the cylinder wall. Sessile polyps will be represented when the sphere that represents the polyp is closer to the wall, while pedunculated polyps will be displayed when the sphere is far from the wall, although they will miss the stalked element. Additionally, the process of domain adaptation can further

modify these features, helping to compensate for the lack of pedunculated polyps. CycleGAN, for instance, can enhance a synthetic image featuring a polyp suspended in space, not in contact with the colon walls, by creating a stalk-like structure that links the polyp to the colon.

Note that this dataset is designed to perform well on benchmark datasets, which is why each image features just one polyp (since all benchmark datasets include images with one polyp, not zero or two). As pointed out in Section 3.2, this approach isn't ideal. For a synthetic dataset to be truly useful in real-world situations, it should include images with zero or several polyps.

Lighting is composed of a white ambient light, two white dynamic lights that project glare into the walls, and three negative lights that project black light at the end of the colon. Having a dark area at the end helps CycleGAN to understand the structure of the colon. The 3D scene must be similar to real colon images because otherwise, CycleGAN will not translate the images to the real-world domain. Figure 3.3 illustrates the images and ground truth generated by the 3D engine.

It is worth mentioning that real-world cameras tend to use different Field of View (FOV), or depth of field settings. It could be beneficial to incorporate colonoscopy-specific augmentations alongside the standard augmentations utilised. Exploring the integration of camera-specific parameters during the rendering presents a promising avenue for improving the performance of the dataset.

3.5.2 Setup

The experimental pipeline can be divided into multiple steps (see Figure 3.7). The initial phase involves the creation of a synthetic dataset, referred to as Synth-Colon, as described above.

Subsequently, for each of the benchmark datasets under consideration, a CycleGAN [35] model is trained to perform image-to-image translation from the synthetic to the real domain. The training process uses real colonoscopy images extracted from the training set, which are combined with the synthetic images generated earlier.

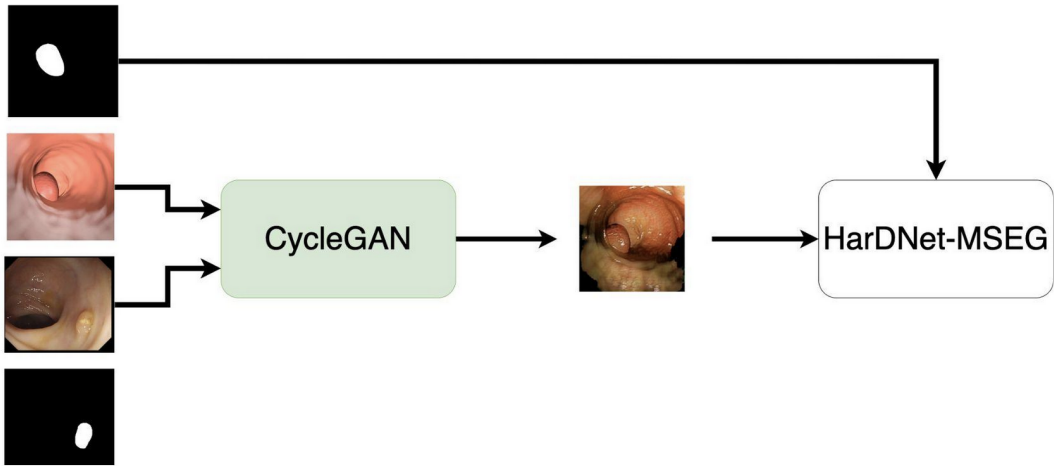


Figure 3.7: Methodology for training a CycleGAN and a HarDNet-MSEG model utilizing both authentic and synthetic datasets. It is important to observe that real annotations are not employed in this process.

The parameters and code used for training CycleGAN are from the official release issued by the original authors.

Experiments conducted on CycleGAN across a range of epochs reveal that optimal performance is obtained at the default parameter setting of epoch 200, defined by the original authors. When the model is trained for fewer epochs, the generated images tend to lack specific real-world features. Conversely, extending the training beyond epoch 200 leads to a phenomenon where the generator starts creating new polyps in the image. Such deviations from the ground truth make these images unsuitable for training the segmentation model.

Once the CycleGAN models have been trained adequately, they are then utilised to translate all synthetic images into the real domain. This translation helps bridge the gap between synthetic and real image distributions.

Then, HarDNet-MSEG [78] is trained using the translated synthetic images, alongside their corresponding synthetic ground truth labels. Note that this experimental methodology is applied across five distinct datasets defined in Section 3.3.1. Consequently, this results in the training of five individual CycleGAN models and an equal number of HarDNet-MSEG models—one for each dataset. The parameters and code used for training HarDNet-MSEG are from the official release issued by the original authors.

As this chapter is a preliminary study, the hyperparameters employed are the ones suggested by the original authors for both models. However, fine-tuning these parameters could lead to enhanced outcomes.

Finally, the segmentation models are evaluated on the corresponding test sets using the mean Dice score and mean intersection over union described in the Experiments section.

Note that the primary contribution of this paper lies in the development of Synth-Colon, which, when combined with pre-existing methodologies like CycleGAN and HarDNet-MSEG, illustrates an effective approach for training a model in the domain of polyp segmentation without real-world annotations.

3.6 Synth-Colon

Synth-Colon is a synthetic dataset for polyp segmentation. It is the first dataset generated using zero annotations from medical professionals. The dataset is composed of 19,917 images with a resolution of 500×500. Synth-Colon additionally includes realistic colon images generated with CycleGAN and the Kvasir training set images. Synth-Colon can also be used for the colon depth estimation task [83] because depth and 3D information is provided for each image. Figure 3.1 shows some examples from the dataset. In summary, Synth-Colon includes:

- Synthetic images of the colon and one polyp.
- Masks indicating the location of the polyp.
- Realistic images of the colon and polyps. Generated using CycleGAN and the Kvasir dataset.
- Depth images of the colon and polyp.
- 3D meshes of the colon and polyp in OBJ format.

Synth-Colon is available in the online repository of the paper [62]. The source

code is available for users to create their version of the dataset. Rendering all the images in this process takes four hours.

3.6.1 Video-Colon

Additionally, "Video-Colon" has been created to investigate polyp segmentation applied to real videos. Although the dataset is not utilised in this thesis, it is suggested that researchers use it in future work. The dataset includes videos simulating a colonoscopy with accompanying annotations of the polyp positions.

The dataset has been generated using the same setup and parameters as Synth-Colon, but in a closed loop where the camera is moved alongside the synthetic colon. Video-colon includes the Python code used to generate the dataset, the user is encouraged to generate its own Video-Colon with the desired parameters, like frame rate or video duration.

The process to produce the 3D colon begins with the creation of a 2D circle. The next step involves using Blender's transformation tools to randomly alter the circle's path in three dimensions, adding variation to the structure. This randomised line is then transformed into a cylindrical shape, establishing a basic 3D form similar to the colon's shape. The final step is to apply random adjustments to the surface of the cylinder. This is done to replicate the uneven and textured surface of the colon, adding realism to the model.

Comparable to the process outlined in Section 3.5.1 regarding Synth-Colon generation, there exist instances where certain polyps are not affixed to any colon walls. As previously articulated, it is anticipated that the implementation of domain adaptation will address this by generating a stalked element that connects the polyp to the wall.

Figure 3.8 shows the 3D scene in Blender The viewpoint is located outside the model to illustrate how the dataset is generated. Note that the floating polyps around the colon are not rendered in the dataset, since the camera only visualizes the inside part of the colon. The floating polyps are a residue of the generation

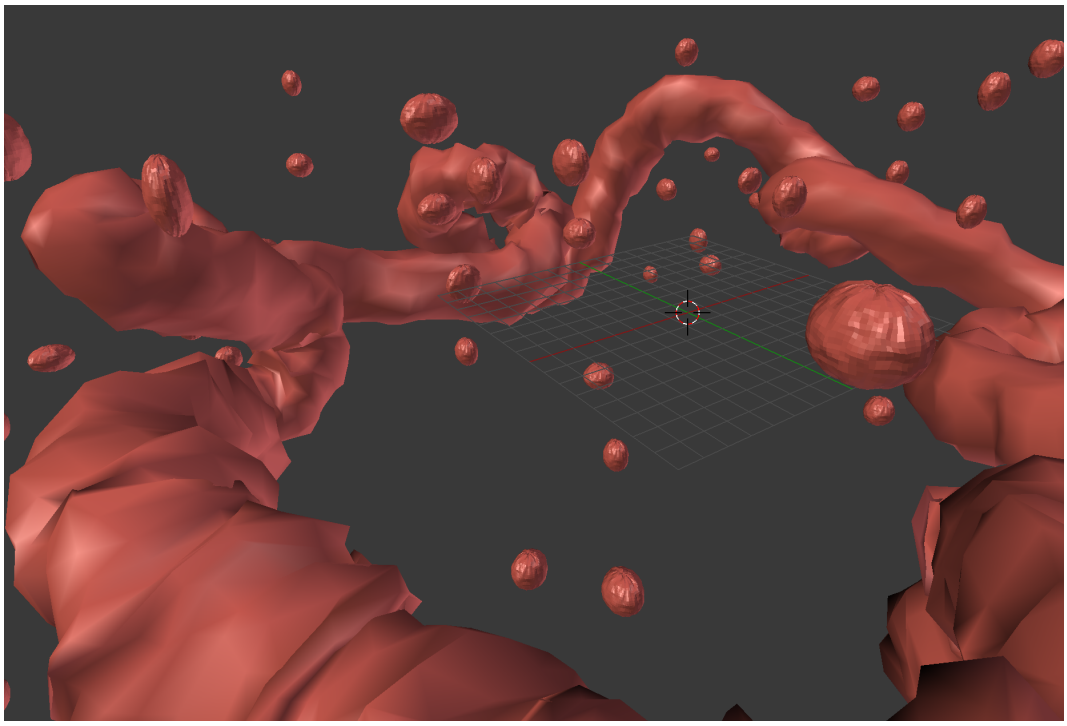


Figure 3.8: Video-colon synthetic dataset scene in Blender. The dataset is available at the online repository [84].

process, that accidentally placed polyps outside the colon, instead of embedded in the inner walls.

3.7 Experiments

This section aims to outline the metrics employed to evaluate the performance of the method, as well as to present an analysis of the method's performance when applied to real polyp datasets. Additionally, a study on the influence of the quantity of real data utilised during the training of the model will also be conducted.

3.7.1 Metrics

Two common metrics are used for evaluation. The mean Dice score, given by:

$$\text{mDice} = \frac{2 \times tp}{2 \times tp + fp + fn}, \tag{3.1}$$

and the mean intersection over union (IoU):

$$\text{mIoU} = \frac{tp}{tp + fp + fn}, \tag{3.2}$$

where in both forumlae, tp is the number of true positives, fp the number of false positives, and fn the number of false negatives.

Both metrics measure the similarity between the model's predictions and the actual polyp with a range of 0 to 1. A score of 1 indicates that the prediction and the ground truth are identical. The main difference between these two measures is that IoU places a heavier emphasis on penalising errors in the form of false positives and false negatives, as compared to DICE, which assigns equal weight to these errors. This is because both of these errors affect not just the numerator (intersection) but also significantly enlarge the denominator (union), thereby reducing the overall IoU score.

3.7.2 Evaluation on real polyp segmentation datasets

Domain adaptation is evaluated on five real polyp segmentation datasets. Table 3.2 shows the results obtained when training HarDNeT-MSEG [78] using synthetic data as described in Figure 3.7. Training the CycleGAN with only the images from the target dataset performs better than training the CycleGAN with all the datasets combined, indicating a domain gap among the real-world datasets.

The performance of the proposed approach, which has an average mDice of 0.55 across all five datasets, is far from HarDNet-MSEG [78] 0.82 mDice. It is worth noting that the proposed approach achieved a better performance than U-Net [75] on ColonDB. Note that U-Net is using labels while the proposed algorithm is entirely based on synthetic annotations.

On ETIS-LaribPolypDB, the performance dropped significantly to 0.25, demonstrating that the proposed approach doesn't generalise as good as the other models.

Table 3.2: Evaluation of domain adaptation on real-world datasets.

	CVC-T		ColonDB		ClinicDB		ETIS		Kvasir	
	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
U-Net [75]	0.710	0.627	0.512	0.444	0.823	0.755	0.398	0.335	0.818	0.746
SFA [76]	0.467	0.329	0.469	0.347	0.700	0.607	0.297	0.217	0.723	0.611
PraNet [77]	0.871	0.797	0.709	0.640	0.899	0.849	0.628	0.567	0.898	0.840
HarDNet-MSEG [78]	0.887	0.821	0.731	0.660	0.932	0.882	0.677	0.613	0.912	0.857
Proposed Approach	0.703	0.635	0.521	0.452	0.551	0.475	0.257	0.214	0.759	0.527

Table 3.3: Evaluation of the proposed approach on the Kvasir dataset when few real images are available. The performance is measured using the mean Dice metric. Note that zero images here means there is no domain adaptation via the CycleGAN. The table doesn't include results between 50 and 900 images because HarDNeT-MSEG obtains a better performance in all of those cases.

	Proposed Approach	HarDNeT-MSEG [78]
0 images	0.356	-
5 images	0.642	0.361
10 images	0.681	0.512
25 images	0.721	0.718
50 images	0.735	0.781
900 (all) images	0.759	0.912

3.7.3 Study with limited real data

In this section domain adaptation is compared with the fully supervised state-of-the-art HarDNeT-MSEG network when there are fewer training examples available. CycleGAN is trained without ground truth segmentation labels, on progressively larger sets of imagery, and this is compared with the supervised method trained on the same amount of labelled imagery. Table 3.3 shows the results of the experiment, which demonstrates that synthetic data is extremely useful for domains where annotations are very scarce. While CycleGAN can produce realistic images with a small sample of only five real images, supervised methods require many images and annotations to achieve good performance. Table 3.3 indicates that unsupervised approaches are useful when there are less than 50 real images and annotations.

3.8 Conclusions

This chapter demonstrates that domain adaptation can be effective in improving the performance of machine learning models on polyp segmentation when no annotations from doctors are used. 3D rendering is used to generate the structure of the colon and generative adversarial networks to make the images realistic and demonstrated that it can perform well in several datasets, even outperforming some fully supervised methods in some cases.

However, there is still significant potential for enhancement and improvement in this field. One major limitation of this approach is that it does not make use of the segmentation masks to translate the images. This implies that the model is unable to utilise the unique attributes of the polyp in the learning process and may potentially result in the displacement of the polyp's position in the transformed image. This limitation acknowledges Research Question 2 (RQ2) which concerns the use of groundtruth in the image translation process. In addition, this approach can be time-consuming because it requires two steps: training CycleGAN (8 hours) and training an image segmentation model (3 hours). This can be a significant drawback if the target domain is very large or if the model needs to be updated frequently with new data. These limitations are addressed in the next chapter.

Chapter 4

Label-aware domain adaptation

In this chapter, an improvement for domain adaptation is presented to address Research Question 2 (RQ2). The improved method, referred to as "CUT-seg", incorporates the use of ground truth information during the image translation process, improving the baseline used in the previous chapter by generating synthetic images that are more realistic and accurate. The work reported in this chapter has been presented in Expert Systems [85] and the code to replicate the experiments is available in the online repository of the paper.

4.1 Introduction

In contrast to the traditional approach of adapting the synthetic data in a separate step before training the machine learning model, CUT-seg involves training both the image translation model and the segmentation model concurrently. CUT-seg jointly trains a HarDNeT-based [64] segmentation model and a Contrastive Unpaired Translation (CUT) [86] image translation model. It transforms synthetic images to the real domain while, at the same time, learning to segment the polyps. As a result, even with only a single real image, CUT-seg performs better than the CycleGAN-based baseline.

Additionally, using the synthetic ground truth data during the image translation serves as a safeguard against the introduction of artefacts and the hallucinatory ten-

dencies often observed in generative networks [87]. In conventional image translation methodologies, generative algorithms can modify the polyp, thus yielding a label that is incongruent with the actual polyp location. These hallucinations significantly undermine the utility of such synthetic images. However, these problems can be prevented by incorporating ground truth data into the translation algorithm. This ensures that the generative model is aware of them and the resultant images not only contain a polyp but also position it accurately, in strict alignment with the label provided by the synthetic ground truth.

4.2 Related work

Given two sets of images from two domains, unpaired image translation is used to transform images from one domain to the other. Ideally, the content of the images is preserved while their style is transformed to the new domain. Image translation models are divided into two main groups: two-sided and one-sided.

Two-sided translation models [35, 88, 89] are bijective, i.e. they transform data from the source domain to the target domain and vice versa thanks to a cycle consistency loss that helps both generative models to converge [35]. These models are memory-intensive because they have to learn both translations. In general, they have at least four models (two generators and two discriminators), which also increases the training time.

One-sided translation models [86, 90] only learn one transformation. As a consequence, they are lighter than the two-sided models. CUT [86] extracts patches from the source image and learns the relationships between them in a self-supervised fashion. It uses a contrastive loss to maximise the mutual information between patches from the same region while minimising the similarity between negative patches from the same image. This loss helps preserve the spatial context of the image. CUT [86] is one of the faster and lighter unpaired image translation models because it is trained with patches from the same image, rather than from the rest of the dataset, thereby obviating the need to store negative samples from other

images in memory. Figure 4.1 shows how the patch-based contrastive loss works: the generator encoder learns that the head of the horse and the head of the zebra belong to the same "content", while the encoder increases the dissimilarity of other patches (horsehair, leg, grass). The generator encoder produces encodings, which are numerical representations of each patch, also known as embeddings. One of the advantages of learning from the encodings is that they are very lightweight and inexpensive to process, contributing to the speed and efficiency of CUT. In addition, CUT includes a discriminator that helps the generator to produce realistic content according to the target domain.

CUT uses a contrastive term to encourage spatial consistency in the generated image. The resulting model does not need a secondary set of generator-discriminator models to regularise the training. The CUT loss function is given by:

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) + \lambda_X \mathcal{L}_{\text{PatchNCE}}(G, H, X) + \lambda_Y \mathcal{L}_{\text{PatchNCE}}(G, H, Y), \tag{4.1}$$

where $\mathcal{L}_{\text{PatchNCE}}(G, H, X)$ is the contrastive term that encourages spatial consistency with the source image in X (synthetic image in this case). This term encourages input-output patches from a particular location in an image to be close to the feature space, and far apart from other patches in the image. G, and D are the weights of the generator and discriminator, respectively. H are the weights of a two-layer perceptron that projects the patches to the feature space and λ_X and λ_Y are hyperparameters that control the contribution of the corresponding contrastive terms.

In general, it offers better results than CycleGAN. Figure 4.2 shows how CUT can map the zebra patterns successfully, particularly in the head, neck, and torso, thanks to its patch-based contrastive loss. CycleGAN struggles to maintain a coherent stripes pattern because it uses a global loss that tries to focus on the whole image. Throughout the remainder of the thesis, CUT is utilisedutilized as a backbone for conducting image translation.

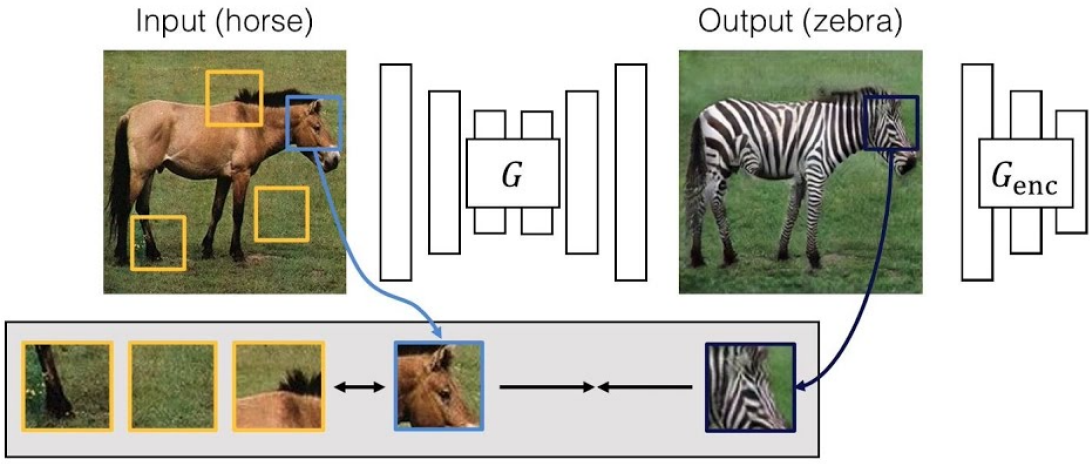


Figure 4.1: CUT diagram by Park et al. [86]. The model operates on patches from the images. It gathers negative examples from within the input image itself. The generator is composed by an encoder and a decoder that work together to translate images between domains while keeping its contents.

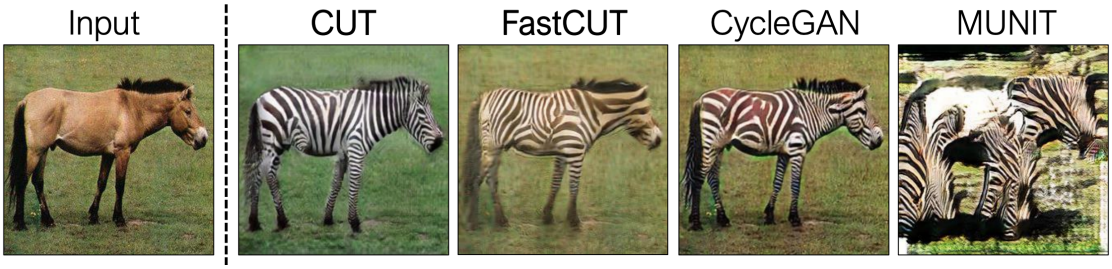


Figure 4.2: Qualitative analysis by Park et al. [86] of various image translation models applied to the horse-zebra transformation: CUT [35, 88]

4.3 Label-aware Domain Adaptation

CUT-seg attaches a HarDNeT-MSEG polyp segmentation architecture to a CUT generator [86]. The segmentation and generative models are jointly trained to generate realistic images with a polyp in a specific position matching the annotation mask.

The initial phase of CUT-Seg involves converting the synthetic image to a representation more similar to real-world polyps, achieved through learning a translation process between the two domains. Then, this transformed image is processed by a segmentation model, which identifies the pixels belonging to the polyp. The segmentation model benefits from the fact that the image, while not entirely real, has undergone modification to resemble real-world scenarios. The concluding step entails a comparison of the predicted segmentation mask with the original synthetic one. Utilising backpropagation, the error between these two masks serves as a means to update the parameters of both models. Consequently, the translation model progressively improves its ability to generate images with a more realistic appearance, while the segmentation model improves its proficiency in accurately identifying real polyps.

If the translation model fails to produce images that closely mimic real-world scenarios, the segmentation model's performance on actual real-world images will be reduced. This is because the segmentation model's training depends on the realism of the images generated by the translation model. Moreover, if the translation model creates images where the polyps do not align with the positions indicated in the synthetic mask, the segmentation model will learn incorrect information. In essence, the accuracy of the segmentation model in identifying polyps is heavily dependent on the translation model's capacity to generate realistic images with accurately placed polyps, mirroring the characteristics of the synthetic masks.

It is important to note that when evaluating the model's effectiveness with actual real-world images, only the segmentation algorithm is employed, as these images do not require conversion into the real-world domain. For this purpose, a distinct

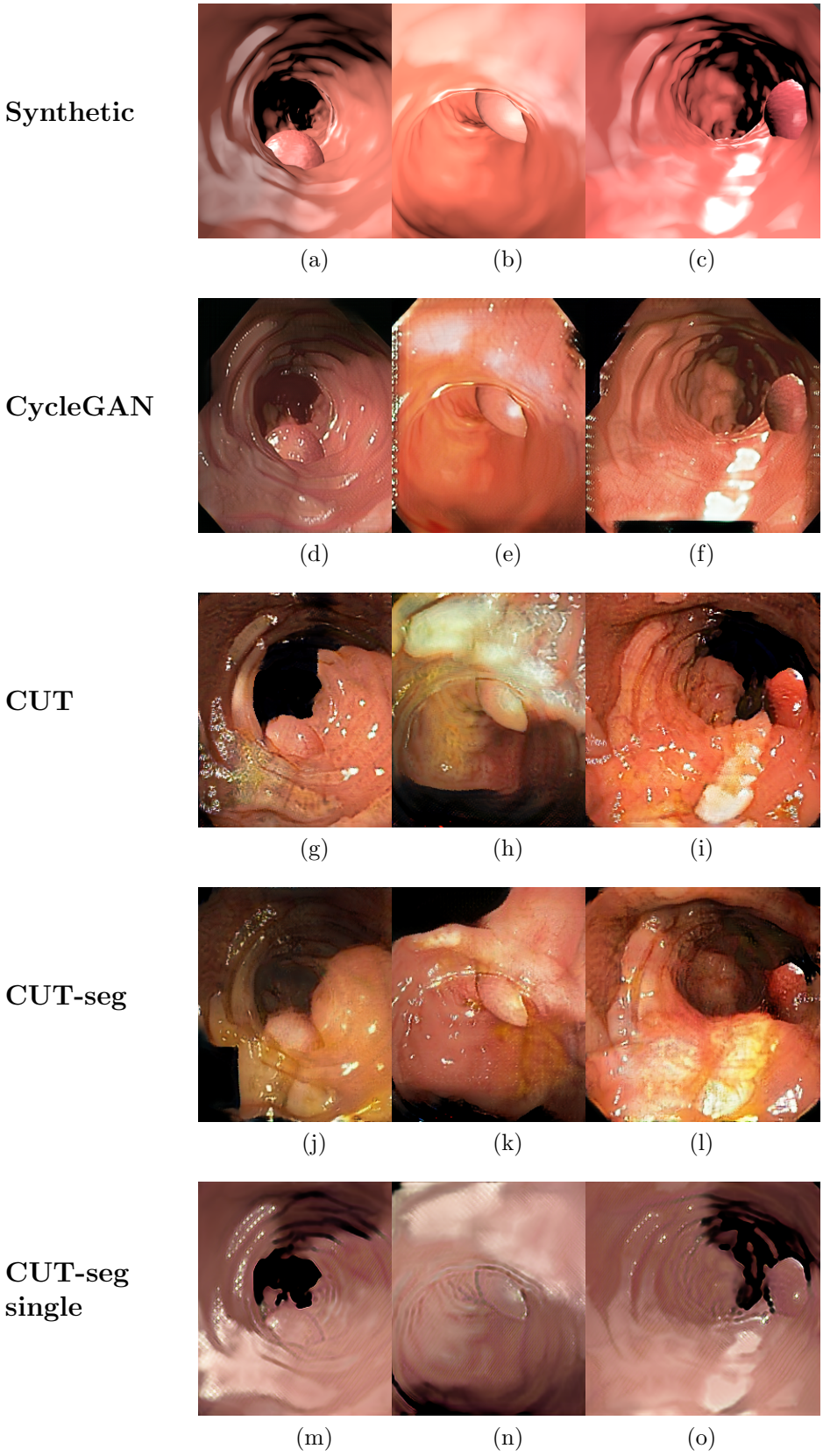


Figure 4.3: Synthetic images (first row), CycleGAN generated images (second row), CUT generated images (third row), CUT-seg generated images (fourth row), and CUT-seg generated images with only one real sample as reference (fifth row).

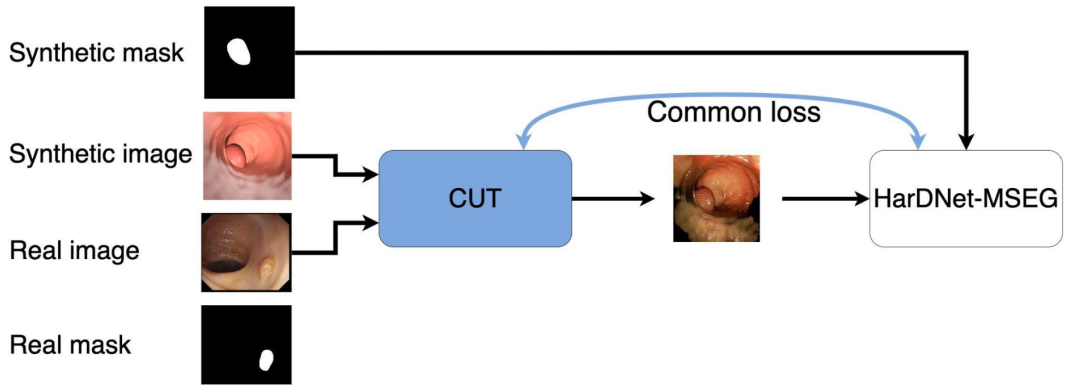


Figure 4.4: CUT-Seg uses a common loss that combines a translation and a segmentation task: generating realistic images while maintaining their coherence with the segmentation mask.

set of images, which were not part of the model's training dataset, is utilised. This approach ensures that the model's performance is tested on entirely new and unseen data, providing a more accurate measure of its capability to generalize and accurately detect polyps in real-world scenarios.

Figure 4.4 illustrates the multiple image sources that are used. Synthetic masks are used as a ground truth, while real masks are not used. The translation model uses a combination of synthetic and real images to produce adapted images, which are sent to the segmentation model.

Combining both models is possible when GPU availability is limited because, unlike CycleGAN, CUT is more memory-efficient, leaving considerable memory on the GPU for the segmentation model. Figure 4.3 shows examples of image translation using CUT alone (third row) and CUT-seg, jointly training CUT and the segmentation model (fourth row) and using only one real sample as reference (fifth row).

In particular, the term $\mathcal{L}_{\text{GAN}}(G, D, X, Y)$ is substituted in Eq. (4.1) with:

$$\mathcal{L}_{\text{GAN}}(G, D, X, Y) + \lambda_S \mathcal{L}_{\text{Seg}}(S), \tag{4.2}$$

where $\mathcal{L}_{Seg}(S)$ is the mean Dice loss on the segmentation masks inferred from the images generated by D, S is the segmentation model, and λ_S a hyperparameter that

controls the weight of the segmentation term.

The Adam optimiser is set with the default momentum values $\beta_1 = 0.9$ and $\beta_2 = 0.999$, a constant value for the learning rate of 1×10^{-5}, the optimal values are obtained using a grid-search.. The other hyperparameters configuration used is the same as in the original CUT paper [86]. The new hyperparameter introduced in this chapter λ_S is set to 0.1. Finding the optimum value for this parameter is particularly important because it regulates the balance between realism and consistency with the synthetic ground truth.

Further refinement of the remaining hyperparameters could potentially enhance the outcomes, although this aspect is not explored in this thesis.

4.4 Experiments

This section presents an evaluation of the proposed approach on real-world datasets and compares its performance to the original method described in Chapter 3. It also investigates the number of synthetic images required for the CUT-seg algorithm to perform effectively. Furthermore, the section presents results obtained when the algorithm is trained using only a single reference image. Experiments are evaluated using the same metrics as in Chapter 3: mean Dice (mDICE) and mean intersection over union (mIoU).

4.4.1 Evaluation on real polyp segmentation datasets

Table 4.1 shows the results obtained with the CycleGAN-based baseline and CUT-seg model. Neither the CycleGAN-based model nor the CUT-seg model uses any real human annotation, unlike the other approaches that are compared in the table. The models are evaluated on five real polyp segmentation datasets in a transductive setup. This is a common setup in zero-shot learning [91, 92, 93, 94] that explores the performance of a model when the unlabeled target data, the test set in this case, is available during training. Unlike the zero-shot setup, in this case, the source data

is also unlabeled, except for the synthetic images that come with free annotations. Transductive evaluation is a valuable setup to bypass the domain gap between target and source data, and better understand the performance of the algorithm in a specific target domain. In this case, this addresses an inherent challenge in the training dataset: the samples from all the datasets are mixed in a single training set. Note that in inductive evaluation, only the labelled source data is available.

CUT-seg displays not only superior performance compared with the CycleGAN baseline but also trains in 3 hours, which is significantly faster than the 11 hours of the CycleGAN approach. CUT-seg average mDIce results are +9% higher than CycleGAN, and the improvement on ClinicDB (+14%) and ETIS (+29%) shows the capabilities of CUT-seg on challenging datasets. It is worth noting that the performance is more stable across datasets since all the mDice results are above 0.5, and also the fact that the synthetic annotations have been generated automatically. However, not all datasets benefit from CUT-seg. For instance, Kvasir, which is the largest dataset, shows a reduced mDice by 5%. It is postulated that CUT-seg doesn't scale in performance when adding more real images to the training, while CycleGAN benefits from a large number of real images. Results in Section 4.4.3 support this hypothesis. Training CUT-seg with only the images from the target dataset performs better than training it with all the datasets combined, indicating a domain gap among the real-world datasets. In this setup, CUT-seg outperforms the CycleGAN baseline in most of the datasets despite requiring less computation. These results are not as good as the fully supervised state-of-the-art HarDNet-MSEG [78] model but considerably reduce the performance gap between methods that use manual annotations and methods that do not.

4.4.2 Synthetic dataset size

In this experiment, the number of synthetic images required for successful training of the CUT-seg algorithm is investigated. For this, the model is trained using 100 real images from the Kvasir dataset and a varying number of synthetic images. Figure 4.5

	CVC-T		ColonDB		ClinicDB		ETIS		Kvasir	
	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
U-Net [75]	0.710	0.627	0.512	0.444	0.823	0.755	0.398	0.335	0.818	0.746
SFA [76]	0.467	0.329	0.469	0.347	0.700	0.607	0.297	0.217	0.723	0.611
PraNet [77]	0.871	0.797	0.709	0.640	0.899	0.849	0.628	0.567	0.898	0.840
HarDNet-MSEG [78]	**0.887**	**0.821**	**0.731**	**0.660**	**0.932**	**0.882**	**0.677**	**0.613**	**0.912**	**0.857**
CycleGAN-based	0.703	0.635	0.521	0.452	0.551	0.475	0.257	0.214	0.759	0.527
CUT-seg	0.700	0.613	0.546	0.396	0.719	0.573	0.540	0.384	0.702	0.621

Table 4.1: Evaluation of the synthetic approach on real-world datasets. The metrics used are the mean Dice similarity index (mDice) and mean intersection over union (mIoU). The best results are highlighted in bold and the best results without human supervision are underlined.

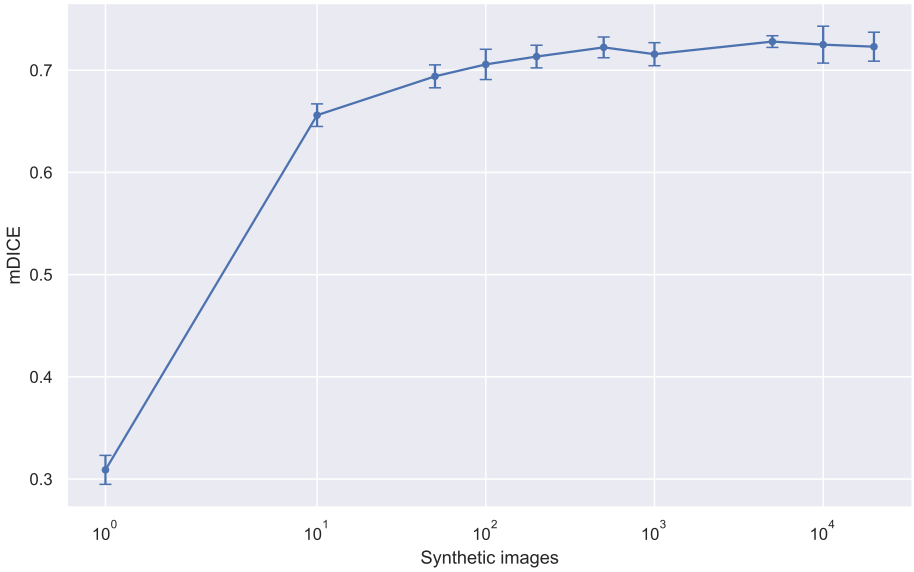


Figure 4.5: Evaluation of CUT-seg with varying amounts of synthetic data on the Kvasir dataset. The performance is measured using the mDice metric, each experiment reports the average mDice across three runs and the error bars indicate the standard deviation.

demonstrates that CUT-seg benefits from a large number of synthetic samples. While the best results are obtained with the largest amounts of samples, CUT-seg reaches near-peak performance when training with 100 or more synthetic images.

4.4.3 Single reference image

Using only a single real image (without ground truth) is sufficient to successfully translate images from the synthetic domain to the real world as shown in Figure 4.3 (fifth row). Table 4.2 shows that the performance improves in some datasets when using only a single real image instead of the full real dataset.

	CVC-T		ColonDB		ClinicDB		ETIS		Kvasir	
	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU	mDice	mIoU
All real images	0.700	0.613	0.546	0.396	**0.719**	**0.573**	**0.540**	**0.384**	0.702	0.621
One real image	**0.754**	**0.617**	**0.569**	**0.422**	0.636	0.563	0.412	0.334	**0.732**	**0.640**

Table 4.2: Comparison between training CUT-seg using the full dataset or a single real image. The metrics used are the mean Dice similarity index (mDice) and mean Intersection over Union (mIoU). The best results are highlighted in bold.

One reason for this phenomenon is due to the algorithm's ability to generate 256 distinct patches from the single reference image, effectively increasing the diversity of the training data. Another reason is that real datasets contain images that are more representative than others. Training with all images in the dataset will inevitably use samples that do not characterise the dataset. When training with an image that represents the style of the target domain, the performance will improve. However, if the selected single image is too different from the typical polyp image, the performance will be lower than training with the entire dataset.

It is worth mentioning, however, that the results depend strongly on which real image is used as reference and the initialisation of the model. Training $N = 10$ runs on the Kvasir dataset with different reference images produce a mean of 0.70 mDice and a standard deviation of 0.02. When training with all the images, the mean is 0.71 and the standard deviation is 0.01. Future work should explore why some images improve the performance, and how to identify these images.

4.5 Conclusions

In this chapter, an end-to-end model for polyp segmentation is proposed. It jointly learns to generate realistic images and segment polyps and demonstrates that joint training allows for faster learning and provides better results than the two-stage counterpart, i.e. training a generative model and segmentation model separately. CUT-Seg demonstrates consistent results over various domains, exceeding a 0.5 mDice score in all five datasets. This indicates that the model typically identifies half of the significant polyp area accurately, which is a level of accuracy that will

ensure robustness. Consequently, this shows that the algorithm is more resilient and has superior generalisation capabilities compared to CycleGAN, which is not as robust. In addition, this chapter addresses RQ2 by demonstrating that attaching a segmentation model to a translation model, and optimising them together, is an effective way to introduce ground truth information into the training loop. In addition, results show that CUT-Seg can learn a domain with a single reference image. The performance of single-image CUT-Seg depends strongly on how representative the image is.

However, despite these advancements, there remains potential for further improvement in the algorithm's performance. One potential area for improvement is in the exposure of the model to real-world images. Currently, the model is only trained on synthetic data that has been adapted to resemble the target domain. While this approach is effective in improving performance, exposure to real-world images may further enhance the model's ability to generalise in the target domain. This is explored in the following chapter.

Chapter 5

Semi-supervised domain adaptation

In this chapter, an improvement to the approach presented in the previous chapter is introduced by incorporating real-world images and labels into the polyp segmentation training process using semi-supervised learning techniques. This aims to address Research Question 3 (RQ3) by offering guidance on how to effectively incorporate real data in training. The work reported in this chapter has been published at the International Joint Conference on Neural Networks 2023 (IJCNN) [95].

In this chapter, the ideas and design are created by both the author of this thesis and Eric Arazo, who is a co-author of the above-mentioned paper. The author of this thesis is responsible for collecting and analysing the data. Although the paper is mainly written by the author of this thesis, Eric Arazo provided valuable feedback and guidance to improve the work.

5.1 Introduction

While synthetic data has already been shown to be useful in the self-supervised setup in Chapters 3 and 4, its application to the semi-supervised setup is yet to be explored. This setup allows the segmentation model to be exposed to both real images and real segmentation masks during training, which narrows the gap between the training and test distributions.

An end-to-end approach to polyp segmentation, referred to as Pseudo-Label CUT-

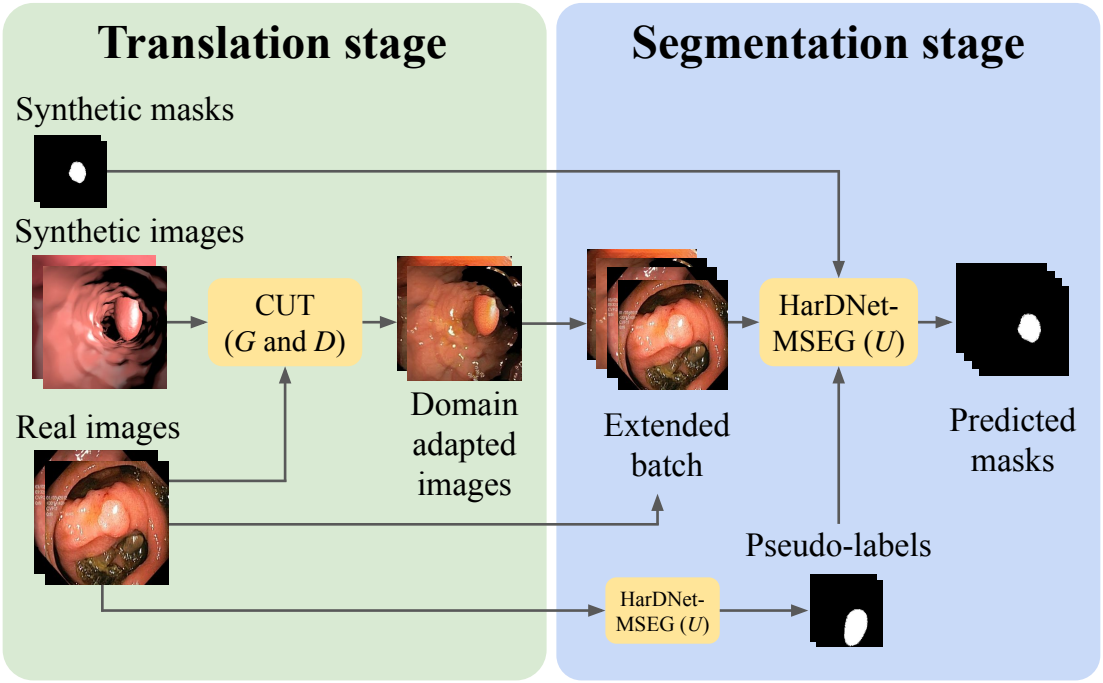


Figure 5.1: Schematic of PL-CUT-Seg in the self-supervised scenario: two-stage architecture for polyp segmentation that leverages real-world data without annotations. PL-CUT-Seg utilises synthetic images, which are translated to the real-world domain, and incorporates pseudo-labeled real-world images within each mini-batch to narrow the gap between the training and testing distributions. This forces the segmentation model, U, to learn features that better generalise to real-world images.

Seg (PL-CUT-Seg), is described in this chapter that learns to adapt synthetic images to the realistic domain and generate segmentation masks that locate the polyps in an image. In particular, PL-CUT-Seg addresses the main weaknesses of Chapter 4 by narrowing the distribution gap between the training domain, i.e. synthetic images, and the test domain, i.e. colonoscopy images. Conversely, as shown in Figure 5.1, the segmentation stage is exposed to both the translated images with the synthetic masks and the real images with model predictions as masks, i.e. pseudo-labels. Additionally, an enhanced version of the model is proposed, PL-CUT-Seg+, and we also introduce an interpolation-based regularisation technique that aims to reduce the domain gap between the real and the synthetic images, which results in improved polyp detection accuracy. Inspired by previous work on pseudo-label approaches [96], a confidence mask is introduced that indicates which regions of the mask should be considered, i.e. the most confident predictions.

This chapter explores the application of synthetic data to polyp segmentation

under different levels of label availability. This approach is based on a careful design of the training batches and regularisation to improve the integration of synthetic data for polyp segmentation in self-supervised and semi-supervised learning, which highlights the potential for exploiting synthetic data as a solution to the data scarcity challenge in medical imaging. Pseudo-labels substantially reduce the domain gap between real and synthetic data by exposing the segmentation model to real images. As a result, the proposed model PL-CUT-Seg reaches state-of-the-art performance in most of the semi-supervised and self-supervised setups in the standard datasets used in polyp segmentation. Additionally, a semi-supervised learning setup is used for polyp segmentation as a valuable benchmark for evaluating the generalisation of the approaches when the availability of data is reduced.

This chapter provides several key contributions to the field of domain adaptation for polyp segmentation. First, an end-to-end model is proposed. It effectively integrates synthetic and real data under different levels of supervision such as semi- and self-supervised learning. Second, a novel semi-supervised approach is presented. It combines pseudo-labels, confidence masks, and mixup in a unified framework to address the main challenges encountered when training with real and synthetic images. Finally, a thorough analysis is conducted on the components of the model and their impact on the generalisation of the model across various datasets for polyp segmentation.

5.2 Related work

Recent research has substantially improved the capacity of neural networks to learn when the labelled data is scarce (semi-supervised learning) or unavailable (self-supervised learning). Semi-supervised learning approaches can be divided into two categories: consistency regularisation and pseudo-labeling approaches. The former leverage the unlabeled data by encouraging the model to produce similar outputs for a single image under different perturbations [97, 98, 99]; the latter uses the predictions of the model as labels for the unlabeled samples [100, 101, 102].

Current state-of-the-art methods combine both alternatives in a holistic approach to semi-supervised learning [96, 103, 104]. Self-supervised learning, on the other hand, initially addressed the lack of labels by using proxy tasks as a supervisory signal, e.g. predicting image rotations [105], inferring colour [106], or predicting the relative position of image patches [107]. Later methods, however, show a shift towards instance-based learning: these guide the training by encouraging features from patches of one image to be closer together while "pushing away" the features from patches from different images [108, 109, 110].

Advances in semi-supervised learning approaches have permeated the semantic segmentation task [111, 112, 113] resulting in methods that propagate the information from a few annotated masks to the rest of the unlabeled samples. Despite the existence of some works applying these techniques to medical imaging [114, 115, 116], few have explored the polyp segmentation task under label scarcity. Only one work explores this paradigm [117], which proposes a non-synthetic approach that can leverage unlabeled data alongside labelled data and investigates the behaviours of this approach with different levels of labelled samples. It utilises focused and dispersive extraction modules to manage the diversity in the location and shape of polyps. Additionally, the model leverages unlabeled data through a discriminator in an adversarial training framework, enhancing the segmentation network's performance. The approach includes consistent regularization to optimize the segmentation networks and an auxiliary adversarial learning method to improve semantic segmentation accuracy. The approach presented here includes synthetic data in the semi-supervised learning process.

5.3 Method

Available datasets for colorectal polyp segmentation are limited in size and fully labelled, and each image x_i has an associated ground truth mask y_i. As a result, semi-supervised and self-supervised learning for this domain are under-explored, and the techniques to exploit additional unlabeled data are underdeveloped. Since the

main characteristic of these setups is the lack of ground truth labels, i.e. segmentation masks, $\mathcal{D}_l$ is defined as the set of labelled samples where each sample x_i has an associated mask y_i and $\mathcal{D}_u$ as the set of unlabeled samples where the corresponding masks are unavailable.

In particular, an additional set $\mathcal{D}_s$ of synthetic image-masks pairs (x_i, y_i) is leveraged. This includes computer-generated data that do not require colonoscopies or annotation, which are invasive, laborious and time-consuming processes. The main challenge when using this data is the domain disparity between synthetic and real images. To address this, PL-CUT-Seg is proposed, a model that leverages the predictions from the segmentation network as pseudo-labels along with the available ground truth masks $y_i \in \mathcal{D}_l$. This model is inspired by the CUT-Seg approach described in the previous chapter, which consists of an unpaired image-to-image translation stage [86] that addresses the image-to-image translation part of the problem and a segmentation stage that generates the masks from adapted synthetic images. By introducing pseudo-labels into the pipeline, PL-CUT-Seg can better integrate the unlabeled samples from the real set $\mathcal{D}_u$ with the synthetic data in $\mathcal{D}_s$.

The process begins with training the generative model using both real and synthetic images. During its training phase, the model produces adapted colonoscopy images that appear more realistic. These images are then forwarded to the segmentation model, which also processes a combination of annotated images and images with pseudo-labels.

Figure 5.2 provides an overview of the model structure. See below a comprehensive list with the precise meaning of the annotations used in the diagram:

- x_i: Colonoscopy image.
- y_i: Segmentation mask indicating the pixels that belong to the polyp.
- z_i: Adapted image from the synthetic domain to the real domain using domain adaptation.
- $\hat{y}_i$: Segmentation mask that is used as a pseudo-label.

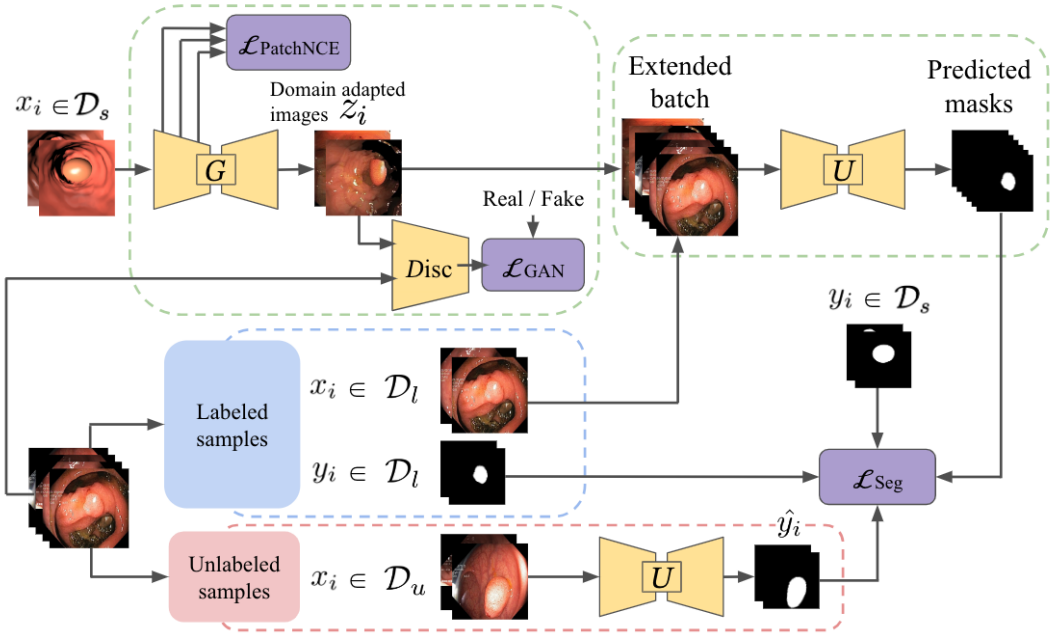


Figure 5.2: Self- and semi-supervised model architecture that utilises a combination of adapted synthetic images, real-world images with annotations, and real-world images with pseudo-labels in each mini-batch. This approach exposes the model to both real-world data and synthetic data and enables it to learn features that are better suited for real colonoscopy images. The blue and red dashed lines encapsulate the labeled and unlabeled samples respectively, the green dashed lines encapsulate the domain adaptation stage, and the orange ones indicate the segmentation stage.

- $\mathcal{D}$: Real domain, includes labeled data $\mathcal{D}_l$ and unlabeled data $\mathcal{D}_u$. Each data sample has an image x_i and an annotation y_i, except $\mathcal{D}_u$ that is unlabeled.
- $\mathcal{D}_s$: Synthetic domain, includes synthetically labeled data. Each data sample has an image x_i and a synthetic annotation y_i.
- G: Generator model.
- Disc: Discriminator model.
- U: Segmentation model.
- $\mathcal{L}_{\text{GAN}}$: Generative loss, includes the loss of the generator and the discriminator.
- $\mathcal{L}_{\text{PatchNCE}}$: Contrastive loss.
- $\mathcal{L}_{\text{Seg}}$: Segmentation loss.

The segmentation stage of PL-CUT-Seg consists of a segmentation network U that generates the binary masks $\hat{y}_i$ that highlight polyps in colonoscopy images x_i. Unlike common approaches to polyp segmentation, PL-CUT-Seg trains on combinations of real labelled and unlabelled images, and synthetic images. Inspired by CUT-Seg in Chapter 4, the transformed synthetic images z_i are used to train U and reduce the need for manual annotation of the real images obtained from colonoscopies: $U(z_i) = \hat{y}_i$. Refer to Section 4.2 for the general loss function. The domain gap between the transformed images z_i and the real images is addressed in two ways: by introducing pseudo-labels and by exposing the segmentation network U to real images and through additional regularisation in the form of interpolation training.

The introduction of pseudo-labels $\hat{y}_i$ as annotations for the unlabeled samples $x_i \in \mathcal{D}_u$ is the main contributor to the improved performance of PL-CUT-Seg. Model predictions are used as segmentation masks and combine the unlabeled samples with the labelled real samples and the synthetic samples in the segmentation stage. The pseudo-labels are initially defined as all-background masks and are updated with model predictions $p_i = U(x_i)$ where $x_i \in \mathcal{D}_u$ through an additional forward pass at the end of every training epoch. The improved loss function corresponding to the segmentation stage is:

$$\mathcal{L}_{\text{Seg}} = \sum_{x_i \in \mathcal{D}} \mathcal{L}_{\text{DICE}}(y_i, p_i), \tag{5.1}$$

where $\mathcal{D} = \mathcal{D}_l \cup \mathcal{D}_u \cup \mathcal{D}_S$, p_i are the model predictions for x_i, and y_i correspond to the available label for the labelled samples, i.e. $x_i \in \mathcal{D}_l$ or $x_i \in \mathcal{D}_S$, and to the pseudo-label $\hat{y}_i$ for the unlabeled, i.e. $x_i \in \mathcal{D}_u$. Then $\mathcal{L}_{\text{DICE}}$ is the DICE loss widely used in semantic segmentation [118]. The generative and segmentation losses are described in Chapter 4.

The domain gap between real and synthetic samples is addressed by using samples from both domains as inputs for the segmentation stage and create each batch B with half of the samples from the real domain and half of them from the synthetic domain: $B = \{s_1, \cdots, s_{\frac{M}{2}}, t_1, \cdots, t_{\frac{M}{2}}\}$, where $s_i \in \mathcal{D}_l \cup \mathcal{D}_u$, $t_i \in \mathcal{D}_S$, and M is the batch size. In the case of self-supervised learning, all real images are associated with

a pseudo-label $\hat{y}_i$, and in the case of semi-supervised learning half of the real samples come from the labelled subset $\mathcal{D}_l$ with the corresponding manual annotations and the other half from the unlabeled set $\mathcal{D}_u$ with the corresponding pseudo-label.

Additionally, for improved results, PL-CUT-Seg+ is proposed, which incorporates interpolation training as a regularisation technique to reduce the domain gap and a pseudo-labeling masking step to reduce the effect of incorrect predictions. Interpolation-based techniques have shown to be an effective solution in semi-supervised learning to reduce the gap between labelled and unlabeled distributions in the training set [99] and to avoid overfitting the labelled samples [100]. The interpolation strategy is inspired by MixUp data augmentation [119], where the model is trained on convex combinations of samples and the corresponding labels, and by previous research boosting semi-supervised learning with interpolation training [100, 96]. Concretely, batch B is extended with the random interpolation of the images in B, which results in $\{s_1, \cdots, s_{\frac{M}{2}}, t_1, \cdots, t_{\frac{M}{2}}, i_1, \cdots, i_M\}$, where $i_n = \lambda s_p + (1-\lambda)s_p$, λ is randomly drawn from a beta distribution as introduced in [119] (see Section 5.4 for details on the parameters used in the beta distribution), and s_p and s_q are randomly selected from B. The masks are interpolated following the same strategy. Unlike [120], which applied interpolation training in polyp segmentation, soft segmentation masks are used where each pixel is the interpolated value of the two corresponding masks. This avoids the need to set up a threshold to obtain binary masks after the interpolation.

Finally, following recent methods on semi-supervised learning [96, 103], the model training is guided only with the most confident predictions, thus avoiding the use of incorrect predictions. This is often implemented as a filtering step where samples associated with lower confidence model outputs are discarded. This is adapted to the semantic segmentation step to mask out the pixels with lower confidence during the computation of the loss. As a result, the DICE loss only considers the pixels of the unlabeled samples that are associated with higher confidence values. Concretely, PL-CUT-Seg+ only considers the pixels associated with predictions with

over 0.999 confidence, for the background and polyps. The optimal threshold value is determined after conducting a logarithmic grid search within the range of 0.9 to 0.999999. It is observed that thresholds higher than 0.999 are too restrictive, preventing the model from learning, while lower thresholds allow too many incorrect pseudo-labels. To prevent the introduction of noise during training, it is necessary to have high confidence values. On the other hand, lower confidence values may have detrimental effects on the performance of the model, as they can result in forwarding incorrect masks to the segmentation model.

End-to-end trained models tend to outperform those that are trained in several stages. This is often attributed to better cooperation between features from different layers and additional freedom from the end-to-end model to learn better-suited features to the end task. As demonstrated in the previous chapter, this also applies to colorectal polyp segmentation and synthetic data. Hence, the two stages of PL-CUT-Seg are integrated into a single backpropagation loop and trained end-to-end.

The overall end-to-end model loss function integrates both the image-to-image translation and the segmentation loss terms. Figure 5.3 shows how the image translation step benefits from real-world data by producing realistic images. Then it trains in an adversarial two-step fashion: on the first step, a binary loss function is employed to optimise the discriminator D from the CUT model to distinguish between real images (from $\mathcal{D}$ or $\mathcal{D}_S$) and images generated by the generator (z); and in the second step, this same binary loss is used to optimise the generator G to generate images that would be classified as real by the discriminator. In this last step, the segmentation term, $\mathcal{L}_{\text{Seg}}$, is also included in the optimisation step, as well as the contrastive terms, $\mathcal{L}_{\text{PatchNCE}}(G, D, X)$ and $\mathcal{L}_{\text{PatchNCE}}(G, D, Y)$, where X is the set of synthetic domain images, and Y the set of real domain images. Given the design of the approach, the translation loss only updates the parameters from G and D, while the segmentation loss updates both the parameters in the segmentation network U and in the discriminator and generator G and D from the translation stage. Note that the segmentation loss is weighted with a hyperparameter λ_S to

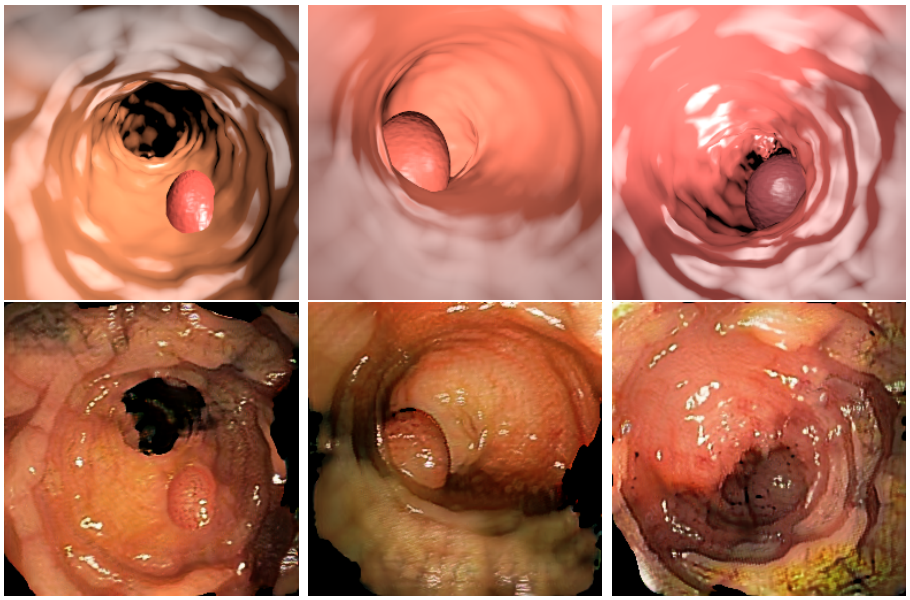


Figure 5.3: Synthetic images (first row) converted to the real-world domain (second row).

control the influence that the segmentation loss has on the overall training of the model.

5.4 Experiments

This section describes the proposed semi- and self-supervised paradigms, the design of the different ablation studies, and the hyperparameters used throughout all the experiments. It provides a thorough analysis of the different elements involved in PL-CUT-Seg and PL-CUT-Seg+ and their role in training the model, exploring their robustness to different levels of label availability, and comparing the model performance to state-of-the-art approaches.

5.4.1 Setup

The experiments in this section explore different levels of supervision by varying the percentage of labelled real samples (size of $\mathcal{D}_l$ with respect to $\mathcal{D}_u$) in the dataset. For the self-supervised setup, it is assumed that all the labels in the real dataset are unavailable ($\mathcal{D}_l = 0$) and train the model with $\mathcal{D}_u$. For the semi-supervised setup, β is defined as the percentage of labelled data in the training set: $\frac{\mathcal{D}_l}{\mathcal{D}_l+\mathcal{D}_u} \times 100$.

Following the work in [117], values of $\beta = 15\%$ and $\beta = 30\%$ are used.

The most widely adopted benchmark for colorectal polyp segmentation consists of a combination of five datasets for the testing stage and a larger set of non-overlapping samples from two of these datasets for the training stage [77, 121, 122]. The testing sets consist of the following datasets: CVC-300 [66] with 60 samples, CVC-ClinicDB [65] with 62 samples, CVC-ColonDB [123] with 380 samples, ETIS [68] with 196 samples, and Kvasir [69] with 100 samples. The training set is constituted of 1450 samples: 550 samples from CVC-ClinicDB and 900 from Kvasir. For the bulk of the experiments, the training set is split into two sets, one for each dataset, with an evaluation of the corresponding testing sets. The performance of the model is evaluated in all the datasets as well as the effects of training with both datasets combined. The amount of synthetic samples is maintained across experiments following the procedure from Chapters 3 and 4: 19,917 samples from a 3D model.

Training

Following the setup in Chapter 4, both networks are trained end-to-end in PL-CUT-Seg: the CUT [86] model followed by the HarDNet-MSEG [78] segmentation network initialised with ImageNet-trained weights on the encoder. The number of epochs is higher than in the original CUT paper because pseudo-labels require more time to converge due to their noisy nature. It is observed that the model converges at epoch 300. Each batch comprises 2 real images and 2 fake images in the translation stage as an input for the CUT model G and D, and 2 translated images and 2 real images in the input of the segmentation model U. When training in the semi-supervised setup, half of the real images are from the labelled set $\mathcal{D}_l$ and half the unlabeled set $\mathcal{D}_u$. The batch size at the input of the segmentation model doubles when applying mixup since the non-interpolated samples are kept. The optimal value of the mixup alpha parameter should change for different datasets and use 2.0 for Kvasir, 0.5 for CVC-ClinicDB, and 0.5 when training with both datasets.

In all the cases the data augmentation and preprocessing techniques applied in the previous chapter are maintained: the 320×320 input images are randomly cropped to 256×256, horizontally and vertically flipped 50% of the times, and randomly rotated between 0 and 360 degrees. All samples are normalised with a (0.5, 0.5, 0.5) per channel mean and standard deviation. Finally, the hyperparameters λ_X and λ_Y, described in Chapter 4, are set to 1 as suggested by Park et al. [86]. Enabling these hyperparameters helps the model maintain the spatial location in the synthetic images. This is necessary to leverage the masks in the synthetic set $\mathcal{D}_s$ generated from the 3D model. The optimal value for λ_X and λ_Y is verified to be optimal by conducting a grid search within the range of 0.001 to 10.

The average duration of the training is 4 hours, which is higher than in previous chapters because the computation of the pseudo-labels requires an extra inference step.

5.4.2 Ablation study

An ablation study is conducted to investigate the effect of the techniques employed in PL-CUT-Seg+. Table 5.1 compares CUT-Seg with the approach proposed in this chapter and provides evidence of the effect of each of the elements introduced. In particular, it presents the results on the Kvasir dataset for self-supervised and semi-supervised setups with 15% and 30% labelled data, which corresponds to the setup used in the benchmarks. These results indicate that the utilisation of pseudo-labels ("+ Pseudo-labels") greatly improves the performance of the framework, in particular by providing the model with exposure to real-world images. Furthermore, the use of Mixup and confidence masks results in further enhancement of the performance achieved with pseudo-labels. The last two rows in Table 5.1, corresponding to PL-CUT-Seg and PL-CUT-Seg+, show improved performance across the different levels of labelled samples.

Labeled samples:	0%		15%		30%	
	mDICE	IoU	mDICE	IoU	mDICE	IoU
CUT-Seg	63.82	55.24	77.91	70.58	80.31	73.18
+ PL (Pseudo-labels)	77.41	68.54	84.08	76.61	85.39	78.32
+ PL + MixUp	77.72	68.72	84.32	77.07	85.87	78.71
+ PL + MixUp + Conf. Mask	**78.08**	**68.77**	**85.52**	**78.19**	**86.94**	**79.58**

Table 5.1: Results for Kavasir validation set for the self-supervised setup and the semi-supervised setup with 30% of labelled samples. Best mDICE and IoU scores are reported. The best results are highlighted using bold text.

5.4.3 Comparison with other self- and semi-supervised approaches

Here we present a comparison of PL-CUT-Seg and PL-CUT-Seg+ with other state-of-the-art algorithms using the Kvasir and CVC-ClinicDB datasets. Table 5.2 shows that the proposed methods achieve improved performance in semi-supervised and self-supervised setups and includes two state-of-the-art approaches for the fully supervised setup (100% labelled samples) as a reference. The performance of PL-CUT-Seg and PL-CUT-Seg+ surpasses other approaches when trained and evaluated on the Kvasir dataset and reduces the performance gap between CUT-Seg and CAL on CVC-ClinicDB. This shows the potential of introducing synthetic data in the training of polyp segmentation approaches. This is especially noticeable as the number of labels decreases.

Additionally, the generalisation ability of the proposed method is evaluated by training on a larger set constituted by both Kvasir and CVC-ClinicDB training sets, and evaluating on the five test sets: Kvasir, CVC-ClinicDB, ETIS, CVC-ColonDB, and CVC-300. Table 5.3 shows that the proposed approach surpasses CUT-Seg, the previous attempt to introduce synthetic data in the pipeline, across all the datasets. Both PL-CUT-Seg and PL-CUT-Seg+ reach competitive results across all levels of annotations and datasets while CUT-Seg fails to generalise to certain datasets, especially in the setup with 15% of labelled samples. The additional regularisation in PL-CUT-Seg+ provides a considerable improvement in most of the cases but seems

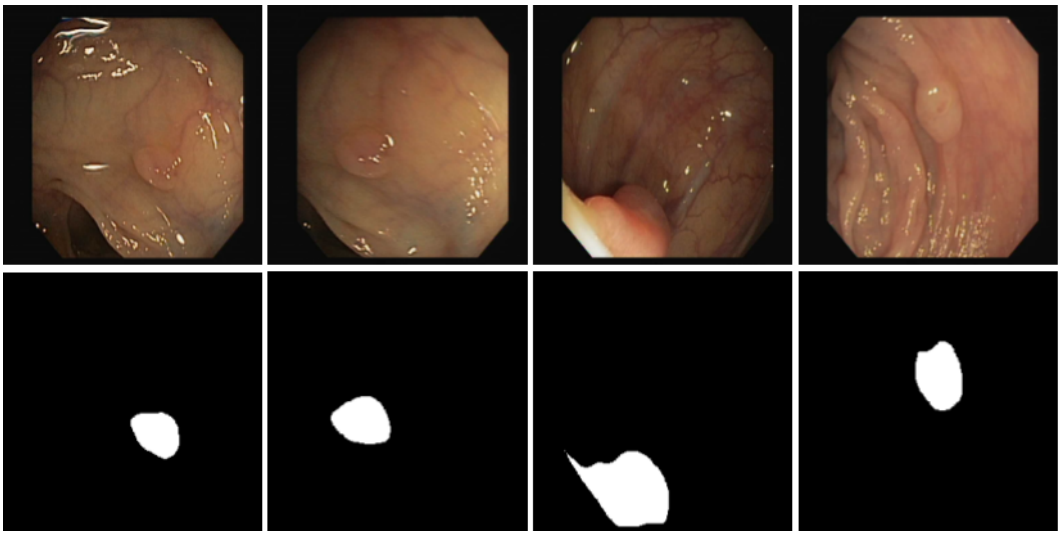


Figure 5.4: CVC-ClinicDB segmentation sample with PL-CUT-Seg+ with 30% of annotations.

to degrade model performance when no labels are available in the dataset, i.e. the self-supervised setup. This suggests that the increased difficulty of training only with unlabeled samples might require weaker regularisation techniques.

Figure 5.4 shows qualitative results on the CVC-ClinicDB dataset with 30% of labels.

5.5 Conclusion

In conclusion, this chapter presented a novel framework for self- and semi-supervised polyp segmentation that achieves excellent results in low data regimes, as seen in Tables 5.3 and 5.2. As proposed in Hypothesis 3 (H3), exposing the segmentation model to both real and synthetic images is crucial for achieving improved performance. This was achieved through the use of pseudo-labels, which allow the model to be exposed to real images in addition to synthetic images.

In the case of CVC-ClinicDB, adding less than 100 annotated images almost obtains state-of-the-art performance, as seen in Table 5.2. In other words, with a minimal amount of annotation combined with synthetic data, it is possible to locate polyps with similar performance to the fully supervised approach that requires 600 annotated colonoscopy images.

		Kvasir		CVC-ClinicDB	
	Labels	mDICE	IoU	mDICE	IoU
HarDNet-MSEG [78]	100%	87.50	79.08	**94.05**	**89.67**
UACANet [121]	100%	**91.20**	**85.90**	92.60	88.00
CAL* [117]	30%	80.95	71.63	**89.29**	**82.57**
CUT-Seg	30%	80.31	73.18	75.43	67.20
PL-CUT-Seg	30%	85.87	78.71	83.99	76.92
PL-CUT-Seg+	30%	**86.94**	**79.58**	85.10	78.08
CAL* [117]	15%	76.76	67.23	**82.18**	**74.98**
CUT-Seg	15%	77.91	70.58	73.93	66.59
PL-CUT-Seg	15%	84.32	77.07	81.48	74.40
PL-CUT-Seg+	15%	**85.52**	**78.19**	81.22	73.90
CUT-Seg	0%	63.82	55.24	52.25	44.17
PL-CUT-Seg	0%	77.72	68.72	**64.61**	**55.83**
PL-CUT-Seg+	0%	**78.08**	**68.77**	56.84	46.70

Table 5.2: Results from training on Kavasir and CVC separately and testing on individual validation sets for the self-sup and semi-supervised setups for Kvasir and CVC. Best mDICE and IoU scores are reported. The best results are highlighted using bold text. Note that the results from CAL (marked with a star *), come from the original paper and the semi-supervised setup might differ slightly.

	Labels	Kvasir	CVC-Cl.	ETIS	CVC-Co.	CVC-300
HarDNet-MSEG [78]	100%	91.29	93.20	67.70	73.10	88.70
UACANet [121]	100%	91.20	92.60	76.60	75.10	91.00
CUT-Seg	30%	64.08	49.33	24.09	31.61	36.27
PL-CUT-Seg	30%	**86.82**	75.82	42.35	**67.59**	**84.04**
PL-CUT-Seg+	30%	86.54	**77.97**	**46.02**	63.89	79.40
CUT-Seg	15%	62.70	34.71	10.31	12.50	5.88
PL-CUT-Seg	15%	85.03	74.71	36.18	**61.79**	**85.44**
PL-CUT-Seg+	15%	**85.71**	**75.95**	**43.68**	58.05	70.93
CUT-Seg	0%	67.23	54.40	31.06	31.45	36.09
PL-CUT-Seg	0%	**78.82**	**66.35**	**33.05**	**52.83**	**56.31**
PL-CUT-Seg+	0%	74.00	57.15	29.82	43.96	44.78

Table 5.3: Results from training on the Kavasir and CVC-ClinicDB training sets merged and tested on individual validation sets for the self-sup and semi-supervised setups. Best mDICE scores are reported. The best results are highlighted using bold text.

Furthermore, the results indicate that synthetic data is particularly useful in scarce label setups, where the availability of data is limited. PL-CUT-Seg was demonstrated to be effective for the task of polyp segmentation. However, it is important to note that further research is needed to evaluate the robustness and generalisability of this approach across other tasks and real-world datasets.

Chapter 6

Domain adaptation for fashion pattern classification

In this chapter, we move away from medical image processing and investigate how synthetic data can be utilised for a different computer vision task in a domain that utilises a large real-world dataset, as opposed to previous studies which employed smaller curated datasets only for research purposes. The domain adaptation techniques from Chapters 4 and 5 are now applied to a different task and domain: fashion pattern classification. This chapter provides evidence that Research Question 3 (RQ3) holds true for other domains that have access to an abundance of real-world data. The work reported in this chapter has been developed in collaboration with Zalando and has been published at the 22nd Scandinavian Conference on Image Analysis (SCIA) [124].

6.1 Introduction

In 2021, 75% of EU internet users bought goods or services online [125]. One of the main drivers of increased e-commerce engagement has been convenience, allowing customers to browse and purchase a wide variety of categories and brands in a single site. If important product metadata is either missing or incorrect, it becomes difficult for customers to find products as the number of available products on e-commerce

sites grows. Online stores typically offer a set of filters (e.g. pattern, colour, size, or sleeve length) that make use of such metadata and help customers to find specific products. If such critical information is missing or incorrect then the product cannot be effectively merchandised. Machine learning has been used for fashion e-commerce in recent works to analyse product images, e.g. clothes retrieval [126], detecting the outline [127], or to find clothes that match an outfit [128]. This chapter is focused on the visual classification task which consists of classifying patterns in catalogue images of clothing. Patterns describe the decorative design of clothes, and they are important because they are widely used by customers to find products online.

Fashion pattern classification is challenging. Fashion images often include models in different poses with complex backgrounds. Achieving high performance requires large annotated datasets [129, 130, 131]. However, public datasets are only available for non-commercial use or do not cover the specific attributes required, while generating private datasets with fine-grained and balanced annotations is expensive. In addition, publicly available fashion datasets typically have underrepresented classes with only a few samples. For example, in the Deep Fashion dataset [129] there are 6633 images with the "solid" pattern while only 242 images contain the "lattice" pattern. Categories that are underrepresented during training achieve a lower performance, thus reducing the overall performance. These problems are addressed by generating Zalando SDG (Zalando Synthetic Data Generation), a synthetic dataset for fashion pattern classification composed of 31,840 annotated images. Figure 6.1 shows fashion visual pattern examples in the synthetic and real-world domains.

However, as seen in previous chapters, synthetic images are not a precise reflection of the real-world domain in which the model will operate. The classification task is not an exception. Even if the synthetic images use realistic lighting and textures that look realistic to humans, the classification model will tend to over-optimise against the traits of the synthetic domain and will fail to generalise well to real-world patterns.

This problem is tackled by using the technique described in Chapter 5. Similar to

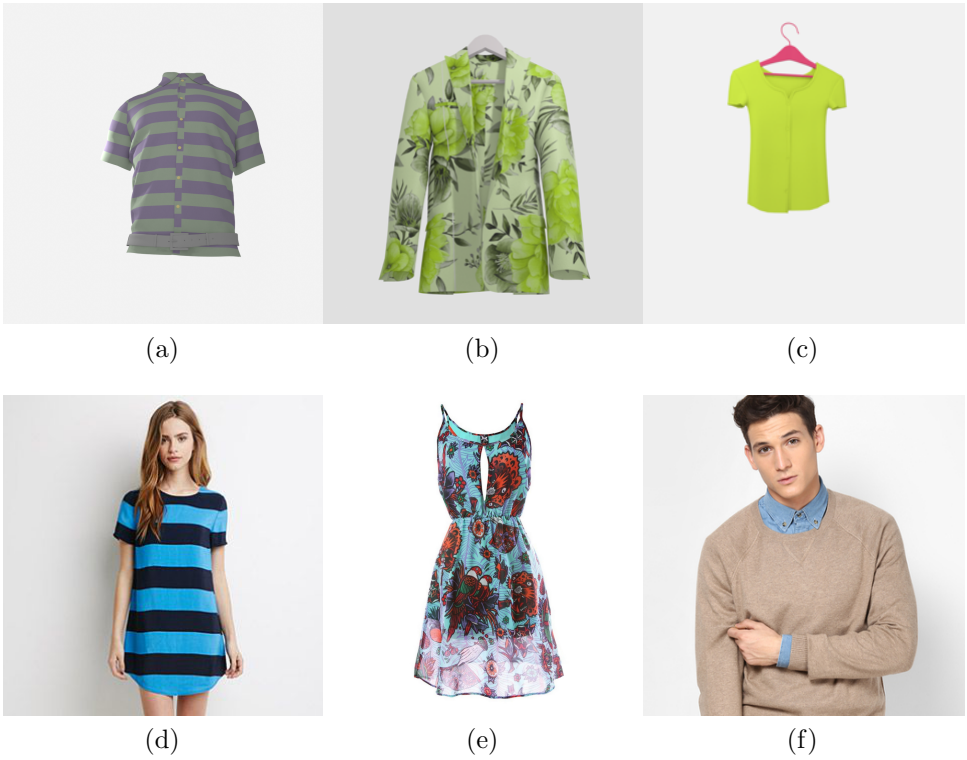


Figure 6.1: Synthetic samples from Zalando SDG dataset (a,b,c) and real samples from the DeepFashion dataset [129] (d,e,f) representing the striped, floral, and plain categories.

PL-CUT-Seg, this chapter introduces Fashion CUT, a domain adaptation approach for classification that does not require ground truth labels. This approach can be used in situations where there is availability of abundant annotated data in the source domain (synthetic images) but a target domain (real images) where no labels are available.

Fashion CUT prevents the patterns from being distorted during the translation step. This problem can appear when complex patterns are shifted to a different domain, they can be distorted to a level that they no longer adhere to the original pattern label for the synthetic image. For example, when an image with the "camouflage" pattern is translated from the synthetic to the real domain, the pattern could be accidentally distorted to "floral".

Fashion CUT provides an end-to-end unsupervised framework to train classification models on unbalanced datasets that outperforms other state-of-the-art unsupervised domain adaptation algorithms, achieving higher accuracy and improved

results. The contribution of this Chapter is the design of an architecture with a generative model and a classifier that is trained to make synthetic images look realistic while preserving the class patterns. In the final stage of the training, inspired by the work described in Chapter 5 real-world pseudo-labelled images are used to improve the model generalisation to real images. The novel architecture performs the image translation task while jointly training a classification model. This approach allows for more efficient and effective training as the translation and classification tasks work together to improve overall performance.

The remainder of the chapter is organised as follows: Section 2 explains the domain adaptation approach; Section 3 presents the Zalando SDG synthetic dataset and experiments, and Section 4 concludes the chapter.

6.2 Related work

In the context of unsupervised domain adaptation for classification, non-adversarial approaches consist of matching feature distributions in the source and the target domain by transforming the feature space to map the target distribution. A relevant Gong et al. [132] found that gradually shifting the domains during training improved the method's stability.

Recent methods are based on generative adversarial networks [36] because of their unsupervised and unpaired nature. Generative domain adaptation approaches rely on a domain discriminator that distinguishes the source and target domains [133] and updates the generator to produce better images. Our approach improves existing adversarial approaches by optimizing a classifier alongside the generator, producing realistic data that retrain the source category.

In this chapter, the proposed Fashion CUT is compared with the top state-of-the-art algorithms on the unsupervised domain adaptation task, which are widely used. To ensure a comprehensive analysis, relevant algorithms from each category are selected, both non-adversarial and adversarial, based on their prominence and performance:

- Xu et al. [134] proposes (Fourier Domain Adaptation) FDA that uses a Fourier-based transformation in the frequency domain to perform the adaptation in a very inexpensive and efficient manner. It doesn't require training any neural network and its adaptation capabilities outperform most other non-trainable methods.

- Chen et al. [135] achieves state-of-the-art accuracy in most domain adaptation benchmark datasets with an adversarial approach. The authors discover that eigenvectors with the largest singular values dominate feature transferability between domains, often at the expense of other eigenvectors crucial for discriminability. To address this issue, they introduce Batch Spectral Penalisation (BSP), a method designed to penalise the largest singular values, thus boosting feature discriminability.

- Zhang et al. [136] introduces another adversarial technique that uses a new metric called Margin Disparity Discrepancy (MDD) that is used to compare the distribution of two domains. The authors successfully bridge the gap between domain adaptation theory and algorithms, offering a more robust approach that achieves state-of-the-art accuracy in domain adaptation tasks.

6.3 Fashion CUT

Fashion CUT has two components: 1) an image translation network that generates realistic images; and 2) a classifier that enforces the generated images to keep the class patterns. The overall architecture is shown in Figure 6.2.

Acquiring paired images from both domains can be difficult to achieve in the fashion domain since image pairs have to occupy precisely the same pixels in the image and have the same pattern. As such, CUT [86] is used for the image translation step, as in Chapters 4 and 5. Synthetic images do not have to match the exact position or texture of real images in the dataset because CUT is an unpaired translation method. CUT learns a mapping that translates unpaired images from the source domain to

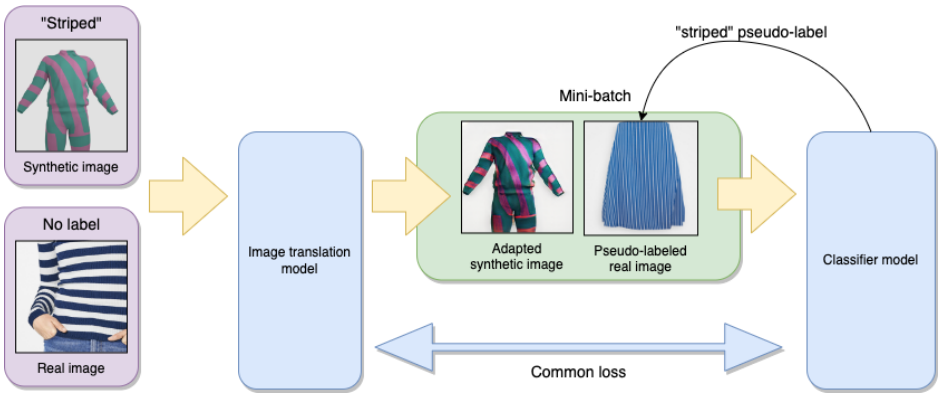


Figure 6.2: The proposed architecture includes a translation model (CUT) and a classifier model (ResNet50), which are optimised together via a common loss that ensures realistic images with reliable annotations. Pseudo-labeled real images are included in each mini-batch to improve the classifier generalisation.

the target domain. It operates on patches that are mapped to a similar point in learned feature space using an InfoNCE [137] contrastive loss. In addition, CUT uses less GPU memory than other two-sided image translation models (e.g. CycleGAN) because it only requires one generator and one discriminator. By reducing memory usage, the joint training of an additional classifier becomes tractable on low-cost GPU setups with less than 16GB of memory.

While CUT produces realistic images, the class patterns can be lost or mixed with other classes since CUT does not enforce that these category features are consistent across the image translation. The generator's only objective is to produce realistic images that resemble the real-world domain, but it ignores the nature of each pattern. Any pattern distorted during the translation will impact the performance of a classifier trained on this synthetic data. Figure 6.3 showcases unsuccessful examples of mixed patterns by the generator. For example, the "herringbone" pattern in Figure 6.3(a) is lost during the image translation, becoming a blurry pattern in Figure 6.3(d). Figure 6.4 shows successful translations using Fashion CUT.

To enforce stability in the generated patterns, a ResNet50 model pre-trained on ImageNet is added to predict the category of the images generated by CUT. The classifier is optimized alongside the CUT generator to fulfil both classification and translation tasks.

Figure 6.4 shows how the classifier preserves the pattern features in comparison

(a) (b) (c)

(d) (e) (f)

Figure 6.3: Synthetic images (a,b,c) and unsuccessfully adapted images using CUT (d,e,f) due to shifted patterns by the generator when not imposing class constraints.

(a) (b) (c)

(d) (e) (f)

Figure 6.4: Synthetic images (a,b,c) and adapted domain images using Fashion CUT (d,e,f).

to vanilla CUT. Training both models simultaneously is faster and provides better results than training them separately. The loss function of the proposed architecture is given by:

$$\begin{aligned} \lambda_g \mathcal{L}_{GAN}(G, D, X, Y) + \lambda_c \mathcal{L}_{classifier}(C) + \\ \lambda_{\mathrm{X}} \mathcal{L}_{PatchNCE}(G, D, X) + \lambda_{\mathrm{Y}} \mathcal{L}_{PatchNCE}(G, D, Y) \end{aligned} \tag{6.1}$$

where $\mathcal{L}_{GAN}(G, D, X, Y)$ is the generator loss, $\mathcal{L}_{classifier}(C)$ is the cross-entropy loss of the classifier inferred from the images generated by the generator. $\mathcal{L}_{PatchNCE}(G, D, X)$ and $\mathcal{L}_{PatchNCE}(G, D, Y)$ are the contrastive losses that encourage spatial consistency for the synthetic and real images, respectively. G is the generator model, D the discriminator model, X the synthetic domain images, Y the real domain images, and C the classification model. λ_g, λ_c, λ_{X}, and λ_{Y} are hyperparameters that control the weight of the generator, the classifier, and both contrastive losses, respectively.

In the experiments, half of the synthetic mini-batch is replaced with images from the target domain. As real-world annotations are not available for generated images, pseudo-labels are predicted by the classifier as in Chapter 5. The model suffers from the cold start problem when introducing pseudo-labels in the early epochs because the classifier struggles to converge. The classifier requires at least 1 epoch of synthetic samples to generate reliable pseudo-labels for real-world images. The best results are obtained when enabling pseudo-labels at the end of epoch 2, which takes approximately 5 hours due to the size of the datasets.

6.4 Experiments

This section describes the synthetic dataset generation process and the two experimental setups used to evaluate Fashion CUT.

Figure 6.5: For each render the scene contains the provided 3D object, added environment and spotlights, and a process applied to the materials to randomise their properties (e.g. colours, scale).

6.4.1 Zalando SDG dataset

The Zalando SDG dataset is composed of 31,840 images of 7 classes: plain, floral, striped, dotted, camouflage, gradient, and herringbone. As with all synthetic datasets in this thesis, it was generated using the framework described in Chapter 7.4. A basic set of professionally modelled 3D objects from CGTrader marketplace is used, representing a variety of fashion silhouettes (e.g. shirt, dress, trousers) and implemented a procedural material for each of the 7 target classes, the properties of the patterns are defined using code. Each procedural material is implemented as a Blender shader node, where multiple properties can be exposed and controlled via Blender Python API. Examples of such properties include pattern scale, colour or colour pairing, orientation and image-texture. This setup allows an arbitrary amount of different images for each 3D object and class pair to be generated programmatically. The background, lighting, and camera position are randomised, as seen in Figure 6.5. It was decided not to use physically based renderers, as they are more resource-intensive. Instead, a trade-off was made in terms of rendering accuracy for speed and the real-time Blender Eevee render engine [138] was adopted as an alternative.

The procedural materials can be applied to any new 3D object. As such they provide a powerful generalised approach to data creation, and the generated images do not require any manual human validation as long as the procedural randomisation guarantees that each possible output belongs to the expected target domain class.

Method	Accuracy
No adaptation	0.441
BSP [135]	0.499
MDD [136]	0.540
AFN [134]	0.578
Fashion CUT	0.613
Fashion CUT with pseudo-labels	**0.628**

Table 6.1: Comparison of unsupervised domain adaptation algorithms on Zalando SDG dataset. The metric used is accuracy.

6.4.2 Evaluation on Zalando SDG dataset

An end-user pattern classification mode is trained using datasets from both 31,840 synthetic fashion imagery (the source domain, which includes ground truth labels), and 334,165 real-world fashion images (the target domain, which has no ground truth labels and is used solely to learn the domain adaptation transformation). The performance of the algorithms is evaluated using a validation set and a test set composed of 41,667 annotated real images each. The metric used is accuracy and all algorithms use a ResNet50 [15] as the classifier. Fashion CUT is optimised using Adam with learning rate 10^{-5}. Both λ_{ncex}, and λ_{ncey} are set to 1.0. $\lambda_g = 0.1$ and $\lambda_{classifier} = 0.1$ for $N = 5$ epochs. The number of epochs used has been determined empirically based on when the classification model ceased to improve. It is worth noting that results are very sensitive to the value of λ_g because it regulates the level of realism in the generated images. Optimal hyperparameters are identified through a grid search.

Table 6.1 compares the performance of domain adaptation algorithms trained only on 31,840 synthetically generated images and evaluated on the 41,667 real fashion images.

First, the performance of training without domain adaptation is measured. In other words, the classifier is trained only on synthetic images. The performance is poor because the model had insufficient information about real-world images.

Second, Zalando SDG is evaluated on other domain adaptation algorithms in the fashion domain. All experiments are performed in the environment provided by

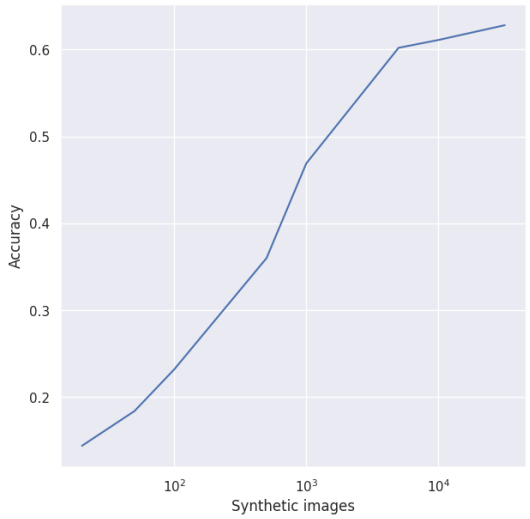


Figure 6.6: Evaluation of Fashion Cut with varying amounts of the Zalando SDG dataset and 10,000 unlabeled real images. The accuracy metric is used in the chart.

Jiang et al. [139]. The Fashion CUT approach outperforms the other algorithms for the pattern classification task. Finally, pseudo-labels are used to improve the results with minor changes in the training.

6.4.3 Synthetic dataset size

This experiment explores the number of synthetic images required to successfully train the proposed domain adaptation algorithm. For this experiment, the model is trained using 10,000 unlabeled real images and changing the number of synthetic images. Figure 6.6 shows that Fashion CUT's performance benefits from large synthetic datasets. Results show that at least 5,000 synthetic images are required to outperform other algorithms in visual pattern classification.

6.5 Conclusions

This chapter demonstrates that the approach from Chapters 4 and 5 can be applied to other tasks (classification) and other domains (fashion). Combining synthetic data generation with domain adaptation can successfully classify patterns in clothes without real-world annotations. Experiments confirm that Fashion CUT outperforms other domain adaptation algorithms in the fashion domain. In addition, pseudo-labels

proved to be beneficial for domain adaptation in the advanced stages of the training.

A key conclusion from this work is that there is no one-size-fits-all technique for using synthetic data to train deep learning models. It is always necessary to adapt to the specific requirements of the task and domain. In this case, annotations were not available, and a self-supervised approach was chosen. In other cases, where labels are available, semi-supervised techniques may be used.

Chapter 7

Conclusions

This concluding chapter is divided into three sections. The first section revisits the hypotheses and research questions outlined in Chapter 1. The second section summarises the key findings of the thesis and possible future work. Lastly, the third section offers final thoughts and recommendations for researchers interested in working with synthetic data in the context of computer vision.

7.1 Hypothesis and research questions

In this section, the hypotheses presented in Chapter 1 are revisited in relation to the research discussed in the subsequent chapters. The research questions related to each hypothesis are also discussed to give a clear understanding of the overall contributions of the thesis.

Hypothesis 1: *It is possible to train a computer vision model exclusively using low-poly synthetic data and without requiring experienced 3D artists and advanced rendering engines.*

Research Question 1: Can domain randomisation compensate for the lack of realism in synthetic images and successfully train a computer vision model using low-quality 3D assets?

The experiments presented in Chapter 2 demonstrate that it is feasible to train a

computer vision model using synthetic data that is not realistic. The key to achieving success with this approach is to generate synthetic data that encompasses a wide range of possibilities, even if they appear unrealistic. This can be accomplished by randomising factors such as lighting, textures, backgrounds, and object geometry. Rather than prioritising high-quality lighting, textures, and 3D models, the focus should shift to increasing the variety of data. Figure 2.3 demonstrates that is possible to train an object counting system via randomisation by replacing the position of the objects accordingly (Section 2.3.1) and applying the right 3D transformations to the meshes (Section 2.4). Additionally, another crucial finding on Section 2.4.2 is that the capacity of the model should be substantial, as it is required to learn a larger domain.

Hypothesis 2: *The incorporation of real-world annotations may enhance the realism of images produced through domain adaptation techniques.*

Research Question 2: What are some ways that domain adaptation can leverage information from segmentation labels to generate images that are more realistic and that are consistent with the segmentation labels?

The studies presented in Chapter 3 and Chapter 4 illustrate that the incorporation of label information can be achieved through the attachment of a task-specific model, which functions on the outputs of the adaptation model. Specifically, Table 4.1 shows how the segmentation accuracy improves in most datasets (29% on ETIS dataset) compared to non-guided models. This task-specific model ensures that a correlation is maintained between the adapted images and the ground truth. It is essential to optimise both models simultaneously. Additionally, manually tunning the ratio between the adaptation and task performance is crucial to attaining optimal results, as discussed in Section 4.3. Lastly, to guarantee efficient implementation, it is necessary to utilise lightweight models like CUT [86] (Section 4.2), as the utilisation of heavy models may lead to a shortage of GPU memory.

Hypothesis 3: *Models trained using domain adaptation are exposed only to a "transformed" version of the data distribution and never encounter genuine real-world images. Assigning soft labels to images from the real world may help the model to better understand the target domain.*

Research Question 3: How can real-world data be incorporated into a domain adaptation process to improve the generalisation capabilities of the model?

Chapters 4, 5, and 6 extensively investigate these questions. It is demonstrated that incorporating real-world data into machine learning models can be achieved through either a semi-supervised approach when some annotations are available for the real-world data (Chapter 4), or by using pseudo-labels (Chapter 5), when a large amount of non-annotated real-world data is available (Chapter 6). The ability to generalise is investigated by applying the techniques in two very different domains - polyp segmentation and fashion classification. The use of synthetic data in semi-supervised learning can greatly benefit from techniques such as mixup, which improves the model's generalisation capabilities As seen in Table 5.3. On the other hand, the use of pseudo-labels is more effective when they are introduced at advanced stages of the training process (Section 6.3).

7.2 Research contributions and future work

This thesis investigated all three research questions comprehensively, yielding new insights but also novel deep-learning models for a variety of tasks that benefit from using synthetic data. The key findings of this research and opportunities for future research are summarised in the following list:

- A significant contribution of this research is the demonstration that altering the geometry of 3D objects in the domain randomisation process can improve the results. A collection of 3D transformations is proposed, with recommendations for their usage depending on the domain. Table 2.4 shows how the object

counting performance is altered by each transformation, subject to the domain. In addition, the code for generating the domain randomisation datasets for counting tasks is also made available for public use (Chapter 7.4) **(RQ1)**.

- One of the primary contributions of this research is the release of the Synth-Colon dataset (utilised in Chapters 3, 4, and 5), which is publicly accessible. There is still potential for improvement in terms of enhancing non-synthetic supervised results for colorectal cancer detection. As the first 3D-based synthetic dataset for polyp segmentation, Synth-Colon can also be employed for future domain adaptation studies. Furthermore, the dataset also includes information about the depth of each image, which was not utilised in this research but was generated as it was straightforward, this data may be used in the field of depth estimation in colonoscopies [83] **(RQ2)**.

- Another key contribution of this research is the ability to train a polyp segmentation model without the use of real-world annotations as seen in Tables 3.2, 4.1, and 5.3. This finding is promising for other fields where synthetic data can be utilised and, with the appropriate domain adaptation techniques to enhance the performance on real-world data. It is recommended to investigate polyp segmentation applied to real videos, as videos are highly challenging to label due to the high number of frames (∼30/s) **(RQ2)**.

- The final contribution of this research is a novel semi-supervised approach that combines pseudo-labels, confidence masks, and mixup in a unified framework to address the main challenges encountered when training with real and synthetic images. Table 5.3 shows how training with 15% of annotations boosts the performance of the polyp segmentation model, reaching values close to the fully-supervised approach. In addition, Table 5.2 proves that adding confidence masks and mixup (PL-CUT-Seg+) further improves the performance of the Kvasir dataset. This approach allows the model to effectively use non-annotated real-world images in synthetic training, by utilising pseudo-labels, which are

generated by the model itself. As future work, it would be valuable to investigate the potential of improving the state-of-the-art supervised algorithms using synthetic data and domain adaptation techniques **(RQ3)**.

7.3 Recommendations

In this section, key recommendations about the utilisation of synthetic data in the field of computer vision are presented:

- Utilising techniques such as domain randomisation are beneficial for synthetic datasets that lack realism. This is supported by the object counting results presented in Chapter 2. Efforts to generate synthetic images that closely resemble real-world images may be futile, as the model may simply memorise the synthetic traits.

- The utilisation of 3D-based synthetic images is only beneficial in scenarios where there is a scarcity of real images. The presence or absence of annotations is insignificant as they can be artificially generated. Table 3.3 shows that synthetic data only performs better when the segmentation model is trained with less than 50 real images.

- It is crucial to approach the use of synthetic data with flexibility and adaptability to the conditions of the task. The conditions of the task such as high variability or sensitivity to augmentations should be taken into account when determining the appropriate use of synthetic data. Additionally, the availability of real-world images and annotations should also be considered. Tables 5.3 and 6.1 show how pseudo-labels have a beneficial effect when combined with synthetic data in cases where the number of real images or annotations is limited. In such cases, the combination of synthetic data with real-world images and annotations can be considered to enhance the training process.

7.4 Final reflections

This thesis has successfully highlighted the significant role of synthetically generated images from 3D models in advancing the field of computer vision. A key insight from this research is that the realism of 3D models isn't always necessary. Simplistic models, when enhanced through domain randomisation, prove to be quite effective. This approach opens new avenues in the utilization of simple 3D models for image generation.

In addition, it has been demonstrated that concurrent training of the model for the main task and the image translation model yields advantageous outcomes, particularly when there is a shared ground truth between the models. This synergy enhances the model's performance and applicability.

Another significant finding is the beneficial impact of pseudo-labels when used with synthetic images. The integration of pseudo-labels improves the results, emphasising their utility in this domain. This thesis does not explore synthetic videos, leaving a potential area for future research. Although a video dataset is provided, its application remains unexplored in this study.

The insights gained from this research have broader implications, particularly in scenarios characterised by a scarcity of images, such as in the diagnosis of rare diseases. The application of 3D-based synthetic data and the understanding of the domain gap are pivotal in these contexts. This thesis sets the foundation for more research in these topics, opening up possibilities for innovative uses in computer vision and other fields.

Bibliography

[1] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. "Mask R-CNN". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2961–2969.

[2] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. "Microsoft COCO: Common Objects in Context". In: *European Conference on Computer Vision*. Springer. 2014, pp. 740–755.

[3] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Nature, 2022.

[4] Yann LeCun and Yoshua Bengio. "Convolutional Networks for Images, Speech, and Time Series". In: *The Handbook of Brain Theory and Neural Networks* 3361.10 (1995), p. 1995.

[5] Abien Fred Agarap. "Deep Learning Using Rectified Linear Units (ReLU)". In: *arXiv preprint arXiv:1803.08375* (2018).

[6] David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. "Learning Representations by Back-propagating Errors". In: *Nature* 323.6088 (1986), pp. 533–536.

[7] Jack Kiefer and Jacob Wolfowitz. "Stochastic Estimation of the Maximum of a Regression Function". In: *The Annals of Mathematical Statistics* (1952), pp. 462–466.

[8] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. "Learning from Synthetic Data for Crowd Counting in the Wild". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019.

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. "The Cityscapes Dataset for Semantic Urban Scene Understanding". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[10] German Ros, Laura Sellart, Joanna Materzynska, David Vazquez, and Antonio M. Lopez. "The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[11] Frank Rosenblatt. "The Perceptron: a Probabilistic Model for Information Storage and Organization in the Brain." In: *Psychological Review* 65.6 (1958), p. 386.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet Classification with Deep Convolutional Neural Networks". In: *Advances in Neural Information Processing Systems (NIPS)* 25 (2012), pp. 1097–1105.

[13] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. "Dropout: a Simple Way to Prevent Neural Networks from Overfitting". In: *The Journal of Machine Learning Research* 15.1 (2014), pp. 1929–1958.

[14] Ricardo Ribani and Mauricio Marengoni. "A Survey of Transfer Learning for Convolutional Neural Networks". In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images Tutorials (SIBGRAPI-T)*. IEEE. 2019, pp. 47–57.

[15] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. "Deep Residual Learning for Image Recognition". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 770–778.

[16] Fisher Yu and Vladlen Koltun. "Multi-scale Context Aggregation by Dilated Convolutions". In: *arXiv preprint arXiv:1511.07122* (2015).

[17] Sergey Ioffe and Christian Szegedy. "Batch Normalization: Accelerating Deep Network Training by Reducing Internal Covariate Shift". In: *International Conference on Machine Learning (ICML)*. pmlr. 2015, pp. 448–456.

[18] Alexander Buslaev, Vladimir I. Iglovikov, Eugene Khvedchenya, Alex Parinov, Mikhail Druzhinin, and Alexandr A. Kalinin. "Albumentations: Fast and Flexible Image Augmentations". In: *Information* 11.2 (2020). ISSN: 2078-2489. URL: https://www.mdpi.com/2078-2489/11/2/125.

[19] Dean A Pomerleau. "Efficient Training of Artificial Neural Networks for Autonomous Navigation". In: *Neural Computation* 3.1 (1991), pp. 88–97.

[20] Maryam Rahnemoonfar and Clay Sheppard. "Deep Count: Fruit Counting Based on Deep Simulated Learning". In: *Sensors* 17.4 (2017), p. 905.

[21] Blender Online Community. *Blender - a 3D Modelling and Rendering Package*. Blender Foundation. Stichting Blender Foundation, Amsterdam, 2018. URL: https://download.blender.org/release/.

[22] Nikolaus Mayer, Eddy Ilg, Philip Hausser, Philipp Fischer, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 4040–4048.

[23] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick Van Der Smagt, Daniel Cremers, and Thomas Brox. "Flownet: Learning Optical Flow with Convolutional Networks". In: *Proceedings*

of the IEEE International Conference on Computer Vision (ICCV). 2015, pp. 2758–2766.

[24] Angel X Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, and Hao Su. "Shapenet: an Information-rich 3d Model Repository". In: *arXiv preprint arXiv:1512.03012* (2015).

[25] N. Mayer, E. Ilg, P. Häusser, P. Fischer, D. Cremers, A. Dosovitskiy, and T. Brox. "A Large Dataset to Train Convolutional Networks for Disparity, Optical Flow, and Scene Flow Estimation". In: *IEEE International Conference on Computer Vision and Pattern Recognition (CVPR)*. arXiv:1512.02134. 2016. URL: http://lmb.informatik.uni-freiburg.de/Publications/2016/MIFDB16.

[26] Nikolaus Mayer, Eddy Ilg, Philipp Fischer, Caner Hazirbas, Daniel Cremers, Alexey Dosovitskiy, and Thomas Brox. "What Makes Good Synthetic Training Data for Learning Disparity and Optical Flow Estimation?" In: *International Journal of Computer Vision* 126.9 (2018), pp. 942–960.

[27] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. "Playing for Data: Ground Truth from Computer Games". In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 102–118.

[28] Robert Geirhos, Patricia Rubisch, Claudio Michaelis, Matthias Bethge, Felix A Wichmann, and Wieland Brendel. "ImageNet-trained CNNs Are Biased Towards Texture; Increasing Shape Bias Improves Accuracy and Robustness". In: *arXiv preprint arXiv:1811.12231* (2018).

[29] Stevo Bozinovski and Ante Fulgosi. "The Influence of Pattern Similarity and Transfer Learning upon Training of a Base Perceptron B2". In: *Proceedings of Symposium Informatica*. Vol. 3. 1976, pp. 121–126.

[30] Josh Tobin, Rachel Fong, Alex Ray, Jonas Schneider, Wojciech Zaremba, and Pieter Abbeel. "Domain Randomisation for Transferring Deep Neural Net-

works from Simulation to the Real World". In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE. 2017, pp. 23–30.

[31] Jonathan Tremblay, Aayush Prakash, David Acuna, Mark Brophy, Varun Jampani, Cem Anil, Thang To, Eric Cameracci, Shaad Boochoon, and Stan Birchfield. "Training Deep Networks with Synthetic Data: Bridging the Reality Gap by Domain Randomisation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*. 2018, pp. 969–977.

[32] Mei Wang and Weihong Deng. "Deep Visual Domain Adaptation: a Survey". In: *Neurocomputing* 312 (2018), pp. 135–153.

[33] Leon A Gatys, Alexander S Ecker, and Matthias Bethge. "A Neural Algorithm of Artistic Style". In: *arXiv preprint arXiv:1508.06576* (2015).

[34] Logan Engstrom. *Fast Style Transfer*. https://github.com/lengstrom/fast-style-transfer/. 2016.

[35] Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. "Unpaired Image-to-Image Translation Using Cycle-Consistent Adversarial Networks". In: *IEEE International Conference on Computer Vision (ICCV)*. 2017.

[36] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. "Generative Adversarial Networks". In: *Communications of the ACM* 63.11 (2020), pp. 139–144.

[37] Enric Moreu, Kevin McGuinness, Diego Ortego, and Noel E O'Connor. "Domain Randomisation for Object Counting". In: *Irish Conference on Artificial Intelligence and Cognitive Science (AICS)* (2021). URL: https://github.com/enric1994/dr4oc.

[38] Yingying Zhang, Desen Zhou, Siqin Chen, Shenghua Gao, and Yi Ma. "Single-image Crowd Counting via Multi-column Convolutional Neural Network". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 589–597.

[39] Ricardo Guerrero-Gómez-Olmedo, Beatriz Torre-Jiménez, Roberto López-Sastre, Saturnino Maldonado-Bascón, and Daniel Onoro-Rubio. "Extremely Overlapping Vehicle Counting". In: *Iberian Conference on Pattern Recognition and Image Analysis*. Springer. 2015, pp. 423–431.

[40] Carlos Arteta, Victor Lempitsky, and Andrew Zisserman. "Counting in the Wild". In: *European Conference on Computer Vision*. Springer. 2016, pp. 483–498.

[41] P Radau, Y Lu, K Connelly, G Paul, A Dick, and G Wright. "Evaluation Framework for Algorithms Segmenting Short Axis Cardiac MRI". In: *The MIDAS Journal-Cardiac MR Left Ventricle Segmentation Challenge* 49 (2009).

[42] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. "You Only Look Once: Unified, Real-time Object Detection". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 779–788.

[43] Haroon Idrees, Imran Saleemi, Cody Seibert, and Mubarak Shah. "Multi-source Multi-scale Counting in Extremely Dense Crowd Images". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2013, pp. 2547–2554.

[44] Viet-Quoc Pham, Tatsuo Kozakaya, Osamu Yamaguchi, and Ryuzo Okada. "Count Forest: Co-voting Uncertain Number of Targets Using Random Forest for Crowd Density Estimation". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2015, pp. 3253–3261.

[45] Lu Zhang, Miaojing Shi, and Qiaobo Chen. "Crowd Counting via Scale-adaptive Convolutional Neural Network". In: *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*. IEEE. 2018, pp. 1113–1121.

[46] Dan Guo, Kun Li, Zheng-Jun Zha, and Meng Wang. "Dadnet: Dilated-attention-deformable Convnet for Crowd Counting". In: *Proceedings of the 27th ACM International Conference on Multimedia*. 2019, pp. 1823–1832.

[47] Junyu Gao, Qi Wang, and Xuelong Li. "Pcc Net: Perspective Crowd Counting via Spatial Convolutional Network". In: *IEEE Transactions on Circuits and Systems for Video Technology* (2019).

[48] Zhaoyi Yan, Yuchen Yuan, Wangmeng Zuo, Xiao Tan, Yezhen Wang, Shilei Wen, and Errui Ding. "Perspective-guided Convolution Networks for Crowd Counting". In: *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*. 2019, pp. 952–961.

[49] Xinkun Cao, Zhipeng Wang, Yanyun Zhao, and Fei Su. "Scale Aggregation Network for Accurate and Efficient Crowd Counting". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018, pp. 734–750.

[50] Mark Marsden, Kevin McGuinness, Suzanne Little, Ciara E Keogh, and Noel E O'Connor. "People, Penguins and Petri Dishes: Adapting Object Counting Models to New Visual Domains and Object Types Without Forgetting". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2018, pp. 8070–8079.

[51] Shanghang Zhang, Guanhang Wu, Joao P Costeira, and José MF Moura. "FCN-rLSTM: Deep Spatio-temporal Neural Networks for Vehicle Counting in City Cameras". In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 3667–3676.

[52] Nicolai Häni, Pravakar Roy, and Volkan Isler. "Minneapple: a Benchmark Dataset for Apple Detection and Segmentation". In: *IEEE Robotics and Automation Letters* 5.2 (2020), pp. 852–858.

[53] Vishwanath A Sindagi, Rajeev Yasarla, and Vishal M Patel. "JHU-CROWD++: Large-Scale Crowd Counting Dataset and a Benchmark Method". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2020).

[54] Leyde Briceno and Gunther Paul. "Makehuman: a review of the modelling framework". In: *Proceedings of the 20th Congress of the International Ergonomics Association (IEA 2018) Volume V: Human Simulation and Virtual*

Environments, Work With Computing Systems (WWCS), Process Control 20. Springer. 2019, pp. 224–232.

[55] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. "Describing Textures in the Wild". In: 2014.

[56] Xiaodong Liu, Weiguo Song, Libi Fu, Wei Lv, and Zhiming Fang. "Typical Features of Pedestrian Spatial Distribution in the Inflow Process". In: *Physics Letters A* 380.17 (2016), pp. 1526–1534.

[57] Bolei Zhou, Agata Lapedriza, Aditya Khosla, Aude Oliva, and Antonio Torralba. "Places: a 10 Million Image Database for Scene Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2017).

[58] Boyu Wang, Huidong Liu, Dimitris Samaras, and Minh Hoai. "Distribution Matching for Crowd Counting". In: *Advances in Neural Information Processing Systems*. 2020. URL: https://github.com/cvlab-stonybrook/DM-Count.

[59] Carlos Martin-Isla, Victor M Campello, Cristian Izquierdo, Kaisar Kushibar, Carla Sendra-Balcells, Polyxeni Gkontra, Alireza Sojoudi, Mitchell J Fulton, Tewodros Weldebirhan Arega, and Kumaradevan Punithakumar. "Deep Learning Segmentation of the Right Ventricle in Cardiac MRI: the M&Ms Challenge". In: *IEEE Journal of Biomedical and Health Informatics* (2023).

[60] Hao Guan and Mingxia Liu. "Domain adaptation for medical image analysis: a survey". In: *IEEE Transactions on Biomedical Engineering* 69.3 (2021), pp. 1173–1185.

[61] Ziyang Chen, Yongsheng Pan, Yiwen Ye, Hengfei Cui, and Yong Xia. "Treasure in Distribution: A Domain Randomization based Multi-Source Domain Generalization for 2D Medical Image Segmentation". In: *arXiv preprint arXiv:2305.19949* (2023).

[62] Enric Moreu, Kevin McGuinness, and Noel E O'Connor. "Synthetic Data for Unsupervised Polyp Segmentation". In: *Irish Conference on Artificial*

Intelligence and Cognitive Science (AICS) (2021). URL: https://github.com/enric1994/synth-colon.

[63] International Agency for Research on Cancer. *Cancer Today - Online Analysis Table*. World Health Organization. 2020.

[64] Ping Chao, Chao-Yang Kao, Yu-Shan Ruan, Chien-Hsiang Huang, and Youn-Long Lin. "Hardnet: a Low Memory Traffic Network". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 2019, pp. 3552–3561.

[65] Jorge Bernal, F Javier Sanchez, Gloria Fernandez-Esparrach, Debora Gil, Cristina Rodriguez, and Fernando Vilarino. "WM-DOVA Maps for Accurate Polyp Highlighting in Colonoscopy: Validation Vs. Saliency Maps from Physicians". In: *Computerized Medical Imaging and Graphics* 43 (2015), pp. 99–111.

[66] David Vazquez, Jorge Bernal, F Javier Sanchez, Gloria Fernandez-Esparrach, Antonio M Lopez, Adriana Romero, Michal Drozdzal, and Aaron Courville. "A Benchmark for Endoluminal Scene Segmentation of Colonoscopy Images". In: *Journal of Healthcare Engineering* 2017 (2017).

[67] Nima Tajbakhsh, Suryakanth R Gurudu, and Jianming Liang. "Automated Polyp Detection in Colonoscopy Videos Using Shape and Context Information". In: *IEEE Transactions on Medical Imaging* 35.2 (2015), pp. 630–644.

[68] Juan Silva, Aymeric Histace, Olivier Romain, Xavier Dray, and Bertrand Granado. "Toward Embedded Detection of Polyps in Wce Images for Early Diagnosis of Colorectal Cancer". In: *International Journal of Computer Assisted Radiology and Surgery* 9.2 (2014), pp. 283–293.

[69] Debesh Jha, Pia H Smedsrud, Michael A Riegler, Pål Halvorsen, Thomas de Lange, Dag Johansen, and Håvard D Johansen. "Kvasir-SEG: a Segmented Polyp Dataset". In: *International Conference on Multimedia Modeling*. Springer. 2020, pp. 451–462.

[70] Sae Hwang, JungHwan Oh, Wallapak Tavanapong, Johnny Wong, and Piet C De Groen. "Polyp Detection in Colonoscopy Video Using Elliptical Shape Feature". In: *2007 IEEE International Conference on Image Processing*. Vol. 2. IEEE. 2007, pp. II–465.

[71] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. "Deep Learning". In: *Nature* 521.7553 (2015), pp. 436–444.

[72] Patrick Brandao, Evangelos Mazomenos, Gastone Ciuti, Renato Caliò, Federico Bianchi, Arianna Menciassi, Paolo Dario, Anastasios Koulaouzidis, Alberto Arezzo, and Danail Stoyanov. "Fully Convolutional Neural Networks for Polyp Segmentation in Colonoscopy". In: *Medical Imaging 2017: Computer-Aided Diagnosis*. Vol. 10134. International Society for Optics and Photonics. 2017, 101340F.

[73] Karen Simonyan and Andrew Zisserman. "Very Deep Convolutional Networks for Large-scale Image Recognition". In: *arXiv preprint arXiv:1409.1556* (2014).

[74] Zongwei Zhou, Md Mahfuzur Rahman Siddiquee, Nima Tajbakhsh, and Jianming Liang. "UNet++: a Nested U-Net Architecture for Medical Image Segmentation". In: *Deep Learning in Medical Image Analysis and Multimodal Learning for Clinical Decision Support*. Springer, 2018, pp. 3–11.

[75] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. "U-net: Convolutional Networks for Biomedical Image Segmentation". In: *International Conference on Medical Image Computing and Computer-assisted Intervention*. Springer. 2015, pp. 234–241.

[76] Yuqi Fang, Cheng Chen, Yixuan Yuan, and Kai-yu Tong. "Selective Feature Aggregation Network with Area-boundary Constraints for Polyp Segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2019, pp. 302–310.

[77] Deng-Ping Fan, Ge-Peng Ji, Tao Zhou, Geng Chen, Huazhu Fu, Jianbing Shen, and Ling Shao. "Pranet: Parallel Reverse Attention Network for Polyp Segmentation". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention*. Springer. 2020, pp. 263–273.

[78] Chien-Hsiang Huang, Hung-Yu Wu, and Youn-Long Lin. *HarDNet-MSEG: a Simple Encoder-Decoder Polyp Segmentation Neural Network that Achieves over 0.9 Mean Dice and 86 FPS*. 2021. arXiv: 2101.07172 [cs.CV].

[79] Vajira Thambawita, Pegah Salehi, Sajad Amouei Sheshkal, Steven A Hicks, Hugo L Hammer, Sravanthi Parasa, Thomas de Lange, Pål Halvorsen, and Michael A Riegler. "SinGAN-Seg: Synthetic Training Data Generation for Medical Image Segmentation". In: *PloS one* 17.5 (2022), e0267976.

[80] Tamar Rott Shaham, Tali Dekel, and Tomer Michaeli. "SinGAN: Learning a Generative Model from a Single Natural Image". In: *Computer Vision (ICCV), IEEE International Conference on*. 2019.

[81] Haoqi Gao and Koichi Ogawara. "Adaptive Data Generation and Bidirectional Mapping for Polyp Images". In: *2020 IEEE Applied Imagery Pattern Recognition Workshop (AIPR)*. IEEE. 2020, pp. 1–6.

[82] Faisal Mahmood, Richard Chen, and Nicholas J Durr. "Unsupervised Reverse Domain Adaptation for Synthetic Medical Images via Adversarial Training". In: *IEEE Transactions on Medical Imaging* 37.12 (2018), pp. 2572–2581.

[83] Anita Rau, PJ Eddie Edwards, Omer F Ahmad, Paul Riordan, Mirek Janatka, Laurence B Lovat, and Danail Stoyanov. "Implicit Domain Adaptation with Conditional Generative Adversarial Networks for Depth Prediction in Endoscopy". In: *International Journal of Computer Assisted Radiology and Surgery* 14.7 (2019), pp. 1167–1176.

[84] Enric Moreu. *Video-Colon*. 2023. URL: https://github.com/enric1994/video-colon.

[85] Enric Moreu, Eric Arazo, Kevin McGuinness, and Noel E O'Connor. "Joint One-sided Synthetic Unpaired Image Translation and Segmentation for Colorectal Cancer Prevention". In: *Expert Systems* 40.6 (2022), e13137. URL: https://github.com/enric1994/cut-seg.

[86] Taesung Park, Alexei A. Efros, Richard Zhang, and Jun-Yan Zhu. "Contrastive Learning for Unpaired Image-to-Image Translation". In: *European Conference on Computer Vision (ECCV)*. 2020.

[87] Matt L Sampson and Peter Melchior. "Spotting Hallucinations in Inverse Problems with Data-Driven Priors". In: *arXiv preprint arXiv:2306.13272* (2023).

[88] Xun Huang, Ming-Yu Liu, Serge Belongie, and Jan Kautz. "Multimodal Unsupervised Image-to-image Translation". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[89] Hsin-Ying Lee, Hung-Yu Tseng, Jia-Bin Huang, Maneesh Singh, and Ming-Hsuan Yang. "Diverse Image-to-Image Translation via Disentangled Representations". In: *Proceedings of the European Conference on Computer Vision (ECCV)*. 2018.

[90] Sagie Benaim and Lior Wolf. "One-Sided Unsupervised Domain Mapping". In: *Conference on Neural Information Processing Systems (NIPS)*. 2017.

[91] Yanwei Fu, Timothy M Hospedales, Tao Xiang, and Shaogang Gong. "Transductive Multi-view Zero-shot Learning". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)* (2015).

[92] Elyor Kodirov, Tao Xiang, Zhenyong Fu, and Shaogang Gong. "Unsupervised Domain Adaptation for Zero-shot Learning". In: *IEEE International Conference on Computer Vision (ICCV)*. 2015.

[93] Jie Song, Chengchao Shen, Yezhou Yang, Yang Liu, and Mingli Song. "Transductive Unbiased Embedding for Zero-shot Learning". In: *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018.

[94] Ziyu Wan, Dongdong Chen, Yan Li, Xingguang Yan, Junge Zhang, Yizhou Yu, and Jing Liao. "Transductive Zero-shot Learning with Visual Structure Constraint". In: *Advances in Neural Information Processing Systems (NeurIPS)* (2019).

[95] Enric Moreu, Eric Arazo, Kevin McGuinness, and Noel E O'Connor. "Self-Supervised and Semi-Supervised Polyp Segmentation Using Synthetic Data". In: *2023 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2023, pp. 1–9.

[96] Kihyuk Sohn, David Berthelot, Nicholas Carlini, Zizhao Zhang, Han Zhang, Colin A Raffel, Ekin Dogus Cubuk, Alexey Kurakin, and Chun-Liang Li. "Fixmatch: Simplifying Semi-Supervised Learning with Consistency and Confidence". In: *Advances in Neural Information Processing Systems (NIPS)* 33 (2020), pp. 596–608.

[97] Antti Tarvainen and Harri Valpola. "Mean Teachers Are Better Role Models: Weight-Averaged Consistency Targets Improve Semi-Supervised Deep Learning Results". In: *Advances in Neural Information Processing Systems (NIPS)* 30 (2017).

[98] David Berthelot, Nicholas Carlini, Ian Goodfellow, Nicolas Papernot, Avital Oliver, and Colin A Raffel. "Mixmatch: a Holistic Approach to Semi-Supervised Learning". In: *Advances in Neural Information Processing Systems (NIPS)* 32 (2019).

[99] Vikas Verma, Kenji Kawaguchi, Alex Lamb, Juho Kannala, Arno Solin, Yoshua Bengio, and David Lopez-Paz. "Interpolation Consistency Training for Semi-Supervised Learning". In: *Neural Networks* 145 (2022), pp. 90–106.

[100] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. "Pseudo-Labeling and Confirmation Bias in Deep Semi-Supervised Learning". In: *2020 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2020, pp. 1–8.

[101] Ahmet Iscen, Giorgos Tolias, Yannis Avrithis, and Ondrej Chum. "Label Propagation for Deep Semi-Supervised Learning". In: *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019, pp. 5070–5079.

[102] Paola Cascante-Bonilla, Fuwen Tan, Yanjun Qi, and Vicente Ordonez. "Curriculum Labeling: Revisiting Pseudo-Labeling for Semi-Supervised Learning". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 35. 8. 2021, pp. 6912–6920.

[103] Bowen Zhang, Yidong Wang, Wenxin Hou, Hao Wu, Jindong Wang, Manabu Okumura, and Takahiro Shinozaki. "Flexmatch: Boosting Semi-Supervised Learning with Curriculum Pseudo Labeling". In: *Advances in Neural Information Processing Systems (NIPS)* 34 (2021), pp. 18408–18419.

[104] Mahmoud Assran, Mathilde Caron, Ishan Misra, Piotr Bojanowski, Armand Joulin, Nicolas Ballas, and Michael Rabbat. "Semi-Supervised Learning of Visual Features by Non-Parametrically Predicting View Assignments with Support Samples". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 8443–8452.

[105] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. "Unsupervised Representation Learning by Predicting Image Rotations". In: *International Conference on Learning Representations (ICLR)*. 2018.

[106] Gustav Larsson, Michael Maire, and Gregory Shakhnarovich. "Learning Representations for Automatic Colorization". In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 577–593.

[107] Carl Doersch, Abhinav Gupta, and Alexei A Efros. "Unsupervised Visual Representation Learning by Context Prediction". In: *International Conference on Computer Vision (ICCV)*. 2015, pp. 1422–1430.

[108] Xinlei Chen and Kaiming He. "Exploring Simple Siamese Representation Learning". In: *Conference on Computer Vision and Pattern Recognition (CVPR)*. 2021, pp. 15750–15758.

[109] Kibok Lee, Yian Zhu, Kihyuk Sohn, Chun-Liang Li, Jinwoo Shin, and Honglak Lee. "I-Mix: a Domain-Agnostic Strategy for Contrastive Representation Learning". In: *International Conference on Learning Representations (ICLR)*. 2021.

[110] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick. "Momentum Contrast for Unsupervised Visual Representation Learning". In: *Computer Vision and Pattern Recognition (CVPR)*. 2020.

[111] Jianlong Yuan, Yifan Liu, Chunhua Shen, Zhibin Wang, and Hao Li. "A Simple Baseline for Semi-supervised Semantic Segmentation with Strong Data Augmentation". In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021, pp. 8229–8238.

[112] Yuchao Wang, Haochen Wang, Yujun Shen, Jingjing Fei, Wei Li, Guoqiang Jin, Liwei Wu, Rui Zhao, and Xinyi Le. "Semi-Supervised Semantic Segmentation Using Unreliable Pseudo-Labels". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 4248–4257.

[113] Shikun Liu, Shuaifeng Zhi, Edward Johns, and Andrew J Davison. "Bootstrapping Semantic Segmentation with Regional Contrast". In: *International Conference on Learning Representations (ICLR)* (2021).

[114] Hong Shang, Zhongqian Sun, Wei Yang, Xinghui Fu, Han Zheng, Jia Chang, and Junzhou Huang. "Leveraging Other Datasets for Medical Imaging Classification: Evaluation of Transfer, Multi-Task and Semi-Supervised Learning". In: *International Conference on Medical Image Computing and Computer-Assisted Intervention (MICCAI)*. Springer. 2019, pp. 431–439.

[115] Dong Wang, Yuan Zhang, Kexin Zhang, and Liwei Wang. "FocalMix: Semi-Supervised Learning for 3D Medical Image Detection". In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. 2020.

[116] Tariq Bdair, Benedikt Wiestler, Nassir Navab, and Shadi Albarqouni. "ROAM: Random Layer Mixup for Semi-Supervised Learning in Medical Images". In: *IET Image Processing* 16.10 (2022), pp. 2593–2608.

[117] Huisi Wu, Guilian Chen, Zhenkun Wen, and Jing Qin. "Collaborative and Adversarial Learning of Focused and Dispersive Representations for Semi-Supervised Polyp Segmentation". In: *IEEE/CVF International Conference on Computer Vision (ICCV)*. 2021.

[118] Luisa F Sanchez-Peralta, Luis Bote-Curiel, Artzai Picon, Francisco M Sanchez-Margallo, and J Blas Pagador. "Deep Learning to Find Colorectal Polyps in Colonoscopy: a Systematic Literature Review". In: *Artificial Intelligence in Medicine* 108 (2020), p. 101923.

[119] H. Zhang, M. Cisse, Y.N. Dauphin, and D. Lopez-Paz. "mixup: Beyond Empirical Risk Minimization". In: *International Conference on Learning Representations (ICLR)*. 2018.

[120] Xiaoqing Guo, Chen Yang, Yajie Liu, and Yixuan Yuan. "Learn to Threshold: Thresholdnet with Confidence-guided Manifold Mixup for Polyp Segmentation". In: *IEEE Transactions on Medical Imaging* 40.4 (2020), pp. 1134–1146.

[121] Taehun Kim, Hyemin Lee, and Daijin Kim. "UACANet: Uncertainty Augmented Context Attention for Polyp Segmentation". In: *Proceedings of the 29th ACM International Conference on Multimedia*. 2021, pp. 2167–2175.

[122] Bo Dong, Wenhai Wang, Deng-Ping Fan, Jinpeng Li, Huazhu Fu, and Ling Shao. "Polyp-PVT: Polyp Segmentation with Pyramid Vision Transformers". In: *arXiv preprint arXiv:2108.06932* (2021).

[123] Jorge Bernal, Javier Sanchez, and Fernando Vilarino. "Towards Automatic Polyp Detection with a Polyp Appearance Model". In: *Pattern Recognition* 45.9 (2012), pp. 3166–3182.

[124] Enric Moreu, Alex Martinelli, Martina Naughton, Philip Kelly, and Noel E O'Connor. "Fashion CUT: Unsupervised Domain Adaptation for Visual Pattern Classification in Clothes Using Synthetic Data and Pseudo-labels". In: *Scandinavian Conference on Image Analysis*. Springer. 2023, pp. 314–324.

[125] Sara Lone, N Harboul, and JWJ Weltevreden. "European E-commerce Report". In: *Ecommerce Europe* (2021).

[126] Xiaodan Liang, Liang Lin, Wei Yang, Ping Luo, Junshi Huang, and Shuicheng Yan. "Clothes Co-parsing via Joint Image Segmentation and Labeling with Application to Clothing Retrieval". In: *IEEE Transactions on Multimedia* 18.6 (2016), pp. 1175–1186.

[127] Ziwei Liu, Sijie Yan, Ping Luo, Xiaogang Wang, and Xiaoou Tang. "Fashion Landmark Detection in the Wild". In: *European Conference on Computer Vision (ECCV)*. Springer. 2016, pp. 229–245.

[128] Vignesh Jagadeesh, Robinson Piramuthu, Anurag Bhardwaj, Wei Di, and Neel Sundaresan. "Large Scale Visual Recommendations from Street Fashion Images". In: *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 2014, pp. 1925–1934.

[129] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations". In: *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016.

[130] Negar Rostamzadeh, Seyedarian Hosseini, Thomas Boquet, Wojciech Stokowiec, Ying Zhang, Christian Jauvin, and Chris Pal. "Fashion-gen: the Generative Fashion Dataset and Challenge". In: *arXiv preprint arXiv:1806.08317* (2018).

[131] Hui Wu, Yupeng Gao, Xiaoxiao Guo, Ziad Al-Halah, Steven Rennie, Kristen Grauman, and Rogerio Feris. "The Fashion IQ Dataset: Retrieving Images by Combining Side Information and Relative Natural Language Feedback". In: *CVPR* (2021).

[132] Boqing Gong, Yuan Shi, Fei Sha, and Kristen Grauman. "Geodesic Flow Kernel for Unsupervised Domain Adaptation". In: *2012 IEEE Conference on Computer Vision and Pattern Recognition*. IEEE. 2012, pp. 2066–2073.

[133] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. "Domain-adversarial Training of Neural Networks". In: *The journal of machine learning research* 17.1 (2016), pp. 2096–2030.

[134] Ruijia Xu, Guanbin Li, Jihan Yang, and Liang Lin. "Larger Norm More Transferable: An Adaptive Feature Norm Approach for Unsupervised Domain Adaptation". In: *The IEEE International Conference on Computer Vision (ICCV)*. 2019.

[135] Xinyang Chen, Sinan Wang, Mingsheng Long, and Jianmin Wang. "Transferability Vs. Discriminability: Batch Spectral Penalization for Adversarial Domain Adaptation". In: *International Conference on Machine Learning (ICML)*. PMLR. 2019, pp. 1081–1090.

[136] Yuchen Zhang, Tianle Liu, Mingsheng Long, and Michael Jordan. "Bridging Theory and Algorithm for Domain Adaptation". In: *International Conference on Machine Learning*. PMLR. 2019, pp. 7404–7413.

[137] Michael Gutmann and Aapo Hyvärinen. "Noise-contrastive Estimation: a New Estimation Principle for Unnormalized Statistical Models". In: *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*. JMLR Workshop and Conference Proceedings. 2010, pp. 297–304.

[138] Ezra Thess Mendoza Guevarra. *Modeling and Animation Using Blender: Blender 2.80: the Rise of Eevee*. Apress, 2019.

[139] Jiang Junguang, Chen Baixu, Fu Bo, and Long Mingsheng. *Transfer-Learning-library*. https://github.com/thuml/Transfer-Learning-Library. 2020.

[140] Enric Moreu. *Technical Framework for 3D Synthetic Image Generation*. Gist repository. 2023. URL: https://gist.github.com/enric1994/40e6e70a70930782ba0d998c01d5ac48.

Appendix A: Technical framework for 3D synthetic image generation

A.1. Introduction

This appendix serves as an introduction for those interested in creating their synthetic datasets. It covers topics such as installing computer graphics software, creating a 3D scene, configuring lighting, and exporting images efficiently. In addition to technical details, this chapter also provides specific recommendations on the best way to create synthetic datasets for training computer vision models. It should be noted that it is not necessary to be a 3D artist to create synthetic images. As seen in previous chapters, highly detailed images are not required, as machine learning techniques can adapt the images so that they are useful for training purposes.

It is also important to note that generating synthetic data is only one aspect of the training process. Synthetic data alone is insufficient for successfully training a computer vision model. The use of synthetic data needs careful consideration and adaptation of existing models and training regimes. This forms the basis for the various investigations detailed throughout this thesis. All the datasets used in this thesis were generated according to the general principles and considerations presented in this appendix. All the code and resources presented in this technical framework can be found in an online repository [140].

A.2. Installing a computer graphics software engine

There is a wide range of computer graphics software available for generating synthetic data. Most of these programs offer basic functions such as importing 3D models and applying various types of lighting effects. Computer graphics software differs in terms of the specific types of rendering engines used, which is beyond the scope of this thesis, and whether they are more oriented towards creating video games or realistic scenes. Another important difference is the programming language used. In general, it is possible to control the software through a graphical interface; however, for our purpose, it is crucial to use code directly as we want to automate the creation of datasets that will have thousands of images.

The computer graphics software used for the generation of 3D synthetic image data is Blender [138]. Blender is an open-source project with a large community supporting its use. One of the main reasons why Blender is used in this thesis is that it has a Python API. This means that the code used to generate synthetic data is consistent with the code that trains computer vision models ensuring a streamlined end-to-end software pipeline.

New versions of Blender are released every few months, whereas new versions typically include updated render engines. In the context of this research, and as can be seen in the next chapter, the level of realism does not necessarily have a strong effect on the performance of the synthetically trained computer vision model. That is why an older version (Blender 2.79b) is used for this research. It includes the basic functionalities we need and a simple render engine that prioritises speed over realism. The software is free and can be downloaded online [21].

A.3. Blender setup

After installing and launching Blender version 2.79b it is important to confirm that the correct version is being used, which can be checked on the splash screen.

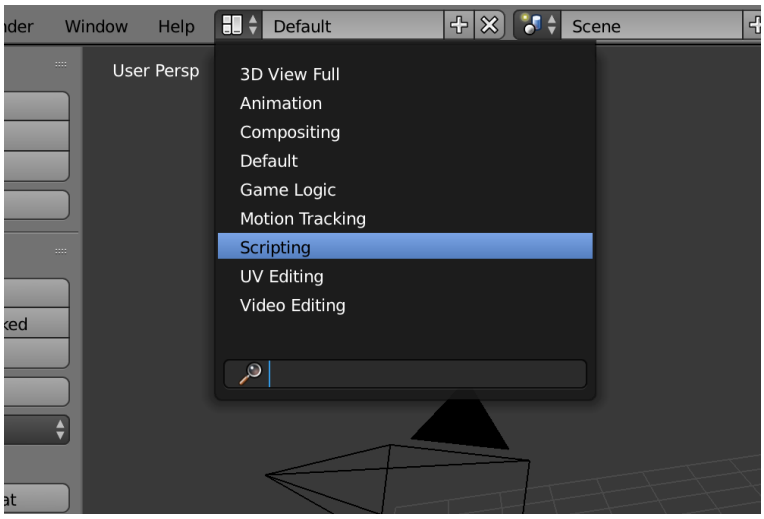


Figure 1: Enable scripting mode to automate Blender using Python.

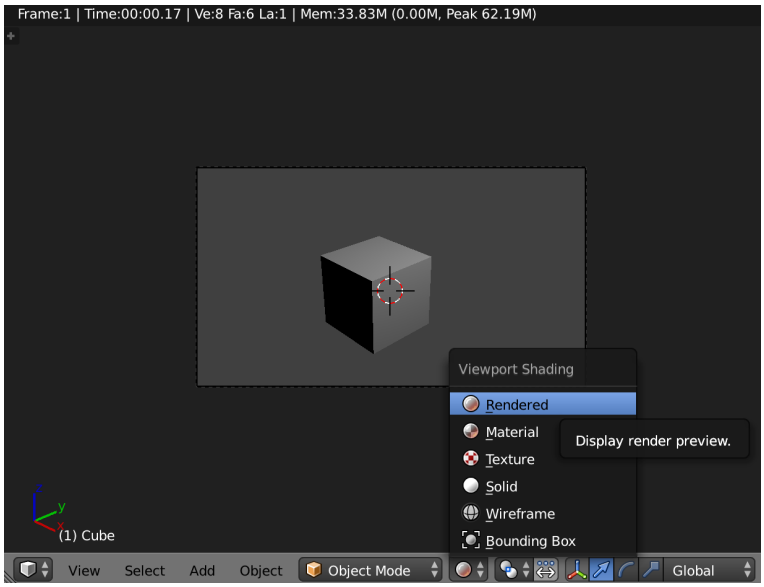


Figure 2: Enable render preview to see precisely what will be rendered.

The default Blender scene includes a cube, a light, and a camera. It is necessary to activate "Scripting mode" to control Blender with Python. To do this, it is necessary to click on "Default" and select "Scripting" as in Figure 1. This will display a new panel on the left where the code can be edited. Additionally, a command line interface will be displayed at the bottom, which can be useful when trying out commands.

Next, it is necessary to enable the camera view to see precisely what will be rendered. This can be done by clicking "View" and then "Camera". In order to display exactly what is going to be rendered, click on the "Viewport Shading" button, which is next to "Object Mode" and then select "Rendered" as in Figure 2.

Since the cube that was loaded by default after initialising Blender will not be used, run the following command in the command line to hide it.

```
bpy.data.objects["Cube"].hide = True
```

After typing this into the command line and pressing Enter, the cube should disappear.

A.4. Importing 3D models

One option when using synthetic data is to create 3D objects from scratch. Designing objects in Blender is relatively easy, especially if the focus is not on realism. Creating a car, a person, or a tree can be achieved easily and with little practice as long as only low-detailed versions are required. The second option is to import 3D objects from files with stadardised formats such as OBJ/STL/FBX, which can be freely downloaded from the internet. For example, Shapenet [24] is a public dataset that offers 51,300 unique 3D models.

One important consideration is the number of faces of the 3D models to be used. Models with many faces (more than 2,000) will increase the memory used by Blender and, memory may be an issue if it is desired to load many objects with many faces. Again, as demonstrated in this thesis, the level of realism is not necessarily important. The number of faces of a model can be checked via the stats that appear at the top right of the Blender interface after clicking on a model.

For illustration purposes, the 3D model "pr2_head_tilt.stl" provided in the online repository [140] is used as an example.

In order to import the model in the scene, the following code should be executed on the command line, remembering to modify the path with the location of the 3D model to be downloaded.

```
bpy.ops.import_mesh.stl(filepath="/path/to/pr2_head_tilt.stl")
```

Note that the 3D model does not include any textures and that it is too small and needs to be scaled. It can be called by a factor of 10 in all three axes with the

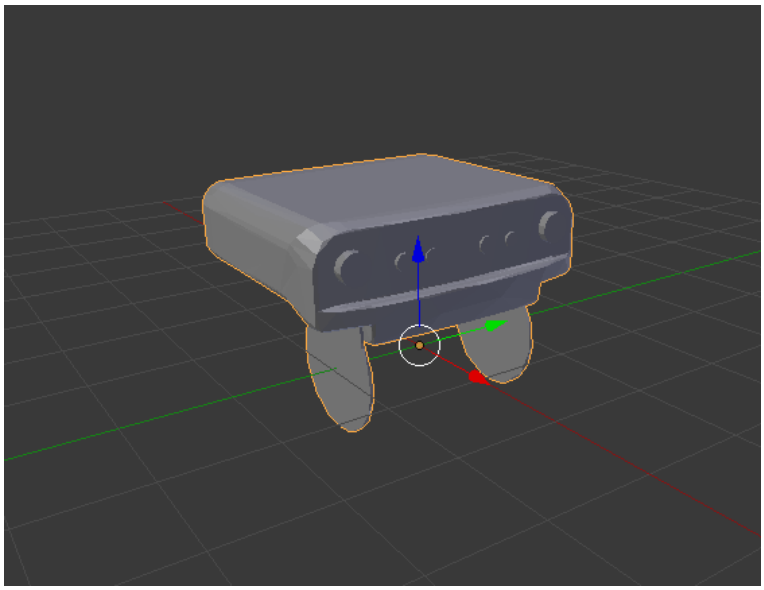


Figure 3: 3D model loaded and re-scaled.

following command:

```
model=bpy.data.objects["Pr2 Head Tilt"]
model.scale=(10,10,10)
```

After running the commands, the Blender interface should display a 3D model (see Figure 3).

A.5. Lighting

Lighting is another aspect that is important to consider in order not to compromise on performance. Some lights, particularly those that create shadows, can be computationally expensive. When adding a light, it is important to consider that it will interact with all objects in the scene. If the aim is to generate datasets within relatively short timeframes, it is crucial to use only two or three light sources. The following command will add a static light and a directional light to the scene. The directional light will be tilted at a 45-degree angle across all three axes. Note that the 3D model in Figure 3 doesn't have any visible illumination because the Blender default viewport doesn't display lights. The final rendered version will include these lights.

```
bpy.ops.object.lamp_add(type="SUN")
bpy.ops.object.lamp_add(type="HEMI")
bpy.data.objects["Hemi"].rotation_euler = (0.78, 0.78, 0.78)
```

A.6. Textures

The loaded 3D model has no texture. A simple and inexpensive way to add colour to 3D models is to map images onto the 3D model. Datasets such as the freely available Describable Textures Dataset [55] offer a wide variety of textures that can be used to make 3D models more closely resemble the real world.

The following script creates a material, an image, and a texture and assigns them to the 3D model. Before executing the code, it is important to download the image "4.2.03.tiff" from the online repository [140].

```
mat = bpy.data.materials.new(name="material")
model.data.materials.append(mat)
tex = bpy.data.textures.new("texture", "IMAGE")
slot = mat.texture_slots.add()
slot.texture = tex
bpy.data.images.new("image", 0, 0)
bpy.data.images["image"].source = "FILE"
tex.image = bpy.data.images["image"]
tex.image.filepath = "/path/to/4.2.03.tiff"
mat.texture_slots[0].texture_coords = "ORCO"
bpy.ops.image.reload()
```

The resulting image will be displayed as illustrated in Figure 4. It is important not to load images with a very high resolution as this will slow down the rendering process.

A.7. Scene background

Sometimes it is important to be able to provide a background for the scene. To illustrate how to do this, we make the cube that was hidden in the first step visible

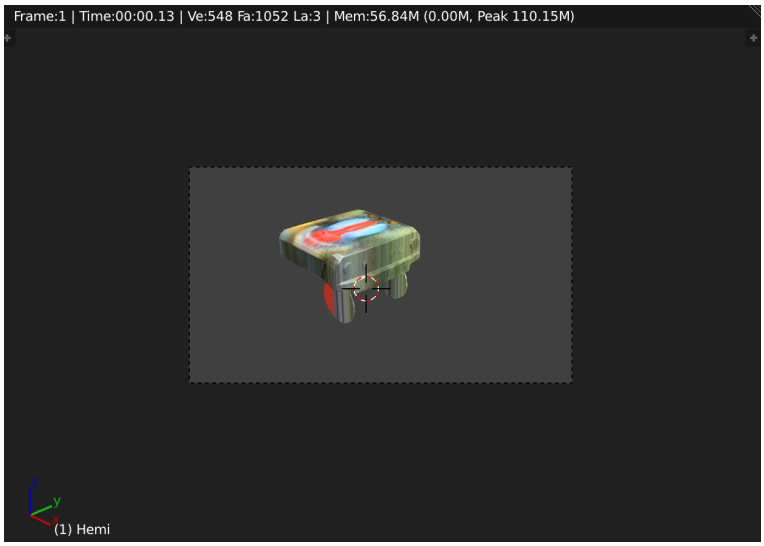


Figure 4: 3D model with an image texture

again, and then position it behind the 3D model and assign a texture. The example background image "4.1.05.tiff" is available in the online repository [140]. Then, the following code should be executed on the command line:

```
cube = bpy.data.objects["Cube"]
cube.hide = False
cube.scale = (8, 8, 0.1)
cube.location = (-2.5, 2, -1)
cube.rotation_euler = (1.22, 0, 0.78)
mat_bg = bpy.data.materials.new(name="material_bg")
cube.material_slots[0].material = mat_bg
tex_bg = bpy.data.textures.new("texture_bg", "IMAGE")
slot = mat_bg.texture_slots.add()
slot.texture = tex_bg
bpy.data.images.new("image_bg", 0, 0)
bpy.data.images["image_bg"].source = "FILE"
tex_bg.image = bpy.data.images["image_bg"]
tex_bg.image.filepath = "/path/to/4.1.05.tiff"
mat_bg.texture_slots[0].texture_coords = "ORCO"
bpy.ops.image.reload()
```

It is important to carefully consider the backgrounds to be used. In the research reported in this thesis, they must be sourced from the real world. Real backgrounds

and textures will help the model learn the properties of the real-world domain. Therefore, it may be advisable to avoid hand-drawn or synthetic images. The Places dataset [57] is an excellent source of real-world backgrounds. However, it is necessary to ensure that there are no images containing objects that are similar to those that the computer vision model is designed to address. For example, if generating a dataset for semantic segmentation on vehicles, backgrounds that feature cars or bicycles should be avoided as they cannot be properly annotated.

A.8. Rendering

This section outlines the procedure for creating a 2D image from a 3D scene. For extensive computer-generated datasets, it is crucial to use a cost-effective method to generate images to avoid excessive dataset creation time. Producing high-quality images is unnecessary since they will be downscaled before being input into the neural network. For instance, the image resolution used in this research is always below 512×512 pixels, limited by the computational resources available. However, various rendering options in Blender, such as motion blur or anti-aliasing, may prove beneficial for certain datasets. The script below renders the current scene in 512×512-pixel resolution using the Blender default configuration:

```
bpy.data.scenes["Scene"].render.resolution_percentage = 100
bpy.data.scenes["Scene"].render.resolution_x = 512
bpy.data.scenes["Scene"].render.resolution_y = 512
bpy.context.scene.render.filepath = "/path/to/first_render.png"
bpy.ops.render.render(write_still=True)
```

The rendered image should look as in Figure 5.

A.9. Annotations

An important aspect of synthetic data is that it implicitly includes annotations, the format of which depends on the type of dataset being generated. For object

Figure 5: Rendered synthetic image.

counting datasets, the annotations can simply be the count of objects in the scene, which can be stored in the image file name (e.g. `crowdcounting.01.24.png`). For object classification datasets, the annotations should indicate the object that has been loaded as the label. For image segmentation datasets, a second image must be rendered corresponding to a segmentation mask.

The most effective approach for generating segmentation masks involves hiding the background, modifying the horizon colour to black and subsequently assigning a new white material with the "shadeless" attribute to the 3D model. The following code renders the segmentation mask, where white corresponds to the 3D object and black corresponds to the background.

```
cube.hide = True
cube.hide_render = True
bpy.data.worlds["World"].horizon_color = (0,0,0)
mat_mask = bpy.data.materials.new(name="material_mask")
mat_mask.diffuse_color = (1,1,1)
mat_mask.use_shadeless = True
model.material_slots[0].material = mat_mask
```

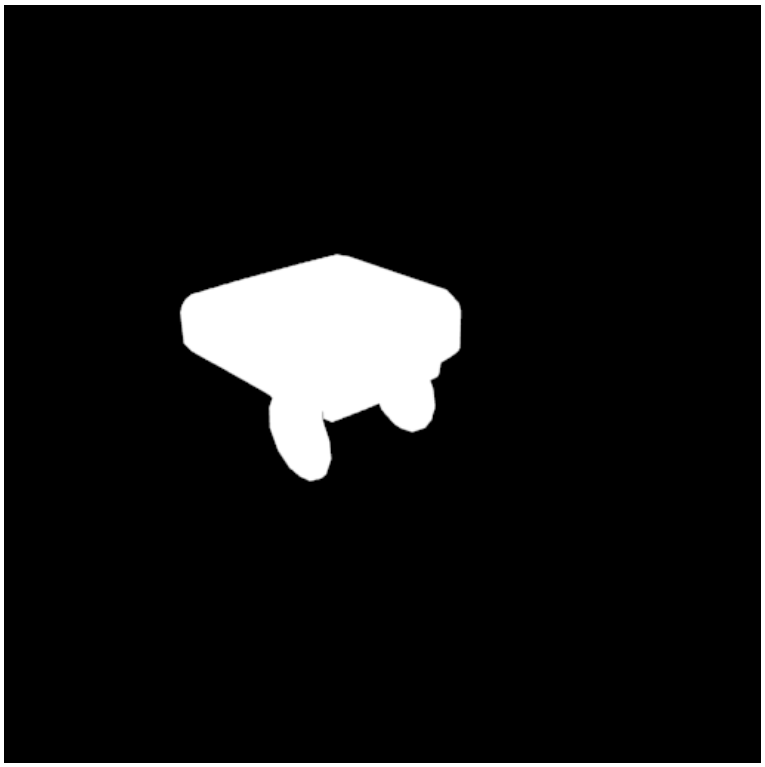


Figure 6: Rendered synthetic mask.

```
bpy.context.scene.render.filepath = "/path/to/mask.png"
bpy.ops.render.render(write_still=True)
```

The rendered mask should look as in Figure 6.

A.10. Randomisation

Given the technical framework outlined so far, it is a very simple matter to implement a loop that generates thousands of images. However, one of the benefits of synthetic data is that it is possible to design the distribution of images, i.e., it is possible to control variations in rotation, position, colour, and even deformations of the 3D models. It is essential to apply these small variations to prevent the model from learning that objects are always in the centre of the screen or facing a particular direction. Another consideration is the sampling of these variables, e.g., if most cars are grey by default, the random colour distribution programmed for car generation should better match the real world with a variety of colourful cars.

The following code applies a rotation, translation, and new size to the 3D model

(in this case, we use a fixed value, but it is recommended to randomise the values for each image).

```
model.rotation_euler = (1.5, 2, -1)
model.location = (2, 0.5, 2)
model.scale = (7, 12, 8)
```

Blender also allows to modify the geometry of 3D models. Using "Modifiers," it is possible to deform the vertices that form the model. This is a very useful tool that enables many variations of the same model. The computer vision model benefits greatly from these alterations because real-world instances (e.g., people in a crowd-counting dataset) will not always be in the same position and will not have a similar appearance.

The following code applies the "Cast" transformation to the 3D model and modifies the geometry of the vertices:

```
bpy.context.scene.objects.active = model
bpy.ops.object.modifier_add(type="CAST")
model.modifiers["Cast"].factor = 1
```

In this case the "Cast" is of spherical type, meaning that the 3D model vertices are displaced to resemble the shape of a sphere.

Lastly, it is also essential to alter the camera parameters, such as the field of view, since generally, datasets in the real world are captured with different types of hardware. The following code alters the camera's field of view (see Figure 7 for result).

```
bpy.data.cameras["Camera"].lens = 40
```

A.11. Closing remarks

Blender is a very powerful tool for computer graphics generation that can be used to produce training datasets for computer vision. With the necessary assets (3D models, textures, backgrounds), it is possible to generate annotated datasets for almost

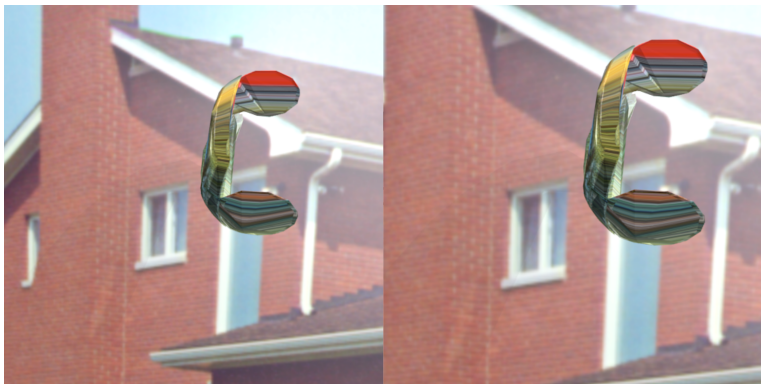


Figure 7: Lens parameter comparison. The left image is set to 40, right image is set to 50.

any task. It is important to find a balance between performance and real-world representation while keeping in mind that the computer vision model will typically learn the most general traits from synthetic images. As mentioned, all the datasets used in this thesis were generated following the principles outlined in this appendix.