Synthetic Data Generation Techniques for Automotive Machine Learning

Master thesis in Mathematical sciences

Rasmus Durgé
Jonny Fredriksson

Department of Mathematical Sciences
CHALMERS UNIVERSITY OF TECHNOLOGY
VOLVO CARS
Gothenburg, Sweden 2023

Master thesis 2023

Synthetic Data Generation Techniques for Automotive Machine Learning

Rasmus Durgé
Jonny Fredriksson

Department of Mathematical Sciences
Chalmers University of Technology
Volvo Cars
Gothenburg, Sweden 2023

Synthetic Data Generation Techniques for Automotive Machine Learning

Rasmus Durgé, Jonny Fredriksson

Supervisors: András Bálint, Mladen Gibanica
Examiner: Serik Sagitov

Master Thesis 2023
Department of Mathematical Sciences
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in LaTeX
Gothenburg, Sweden 2023

Synthetic Data Generation Techniques for Automotive Machine Learning
Rasmus Durgé, Jonny Fredriksson
Department of Mathematical Sciences
Chalmers University of Technology

Abstract

Seat belts drastically reduce the risk of injury or death, given that one is wearing them correctly. This thesis emanates from Volvo Cars' aspiration to tackle this risk, using the growing potential of machine learning. The foundation of this work stems from another thesis at Volvo Cars, where a *semantic segmentation model* was developed, for identifying and segmenting the seat belt in an image of a car occupant. To apply this segmentation model approach, the tedious and costly process of collecting and annotating data is fundamental. The thesis explores the concept of using *synthetic data*, i.e., data that is made by software and annotated in silico, as a substitute for, and a complement to, previously collected real-world data.

Specifically, the thesis explores different methods on how to apply and generate synthetic data and what aspects improve its quality, regarding the prediction accuracy of the segmentation model. As a measure of prediction accuracy, the mean *intersection over union* (IoU) over a test set consisting of real-world images is used. Several segmentation models, with different architectures, are evaluated to find the best-performing network. The thesis also explores the concept of *domain randomization*, which aims to narrow the domain gap between the synthetic and real data, as well as multiple label annotations to investigate whether identifying other objects improves segmentation of the seat belt, and guided backpropagation to explain predictions made by the segmentation model.

This thesis shows that, although the choice of network architecture is shown to have a relatively small effect on performance, the top performing network is found to be a *Unet++* decoder and a *ResNet 34* encoder. The thesis also suggests that when there is a scarcity of real-world data, introducing synthetic data can improve prediction accuracy, both by training the model on a mix of real and synthetic data, and by pre-training the model on synthetic data before training it on real data. The results also suggest that when the model is trained to also identify objects which often interact with the seat belt, e.g., the occupant's shirt, its prediction accuracy on the seat belt can improve.

The thesis has identified ways to make synthetic data more appropriate for training the seat belt segmentation model. This thesis successfully demonstrates that there is a lot of potential to further develop the application of synthetic data in the future. One obvious approach would be to use a more powerful graphics engine, making the synthetic data even more realistic.

Keywords: seat belt, car occupant, semantic segmentation, neural networks, augmentations, synthetic data, domain gap

Acknowledgements

First of all, we would like to thank our industry supervisors András Bálint and Mladen Gibanica, for their valuable guidance and dedication throughout the work on this thesis. Working with you has been truly inspiring.

We would like to thank Filip Johansson and Sebastian Linde, for their tireless work in providing us with amazing synthetic data, always accommodating our requests. You have been absolutely fundamental in this thesis.

We would also like to thank our former examiner Johan Jonasson, for his valuable, insightful input in the early stages of this thesis, and our examiner Serik Sagitov, for taking on our thesis at such a late stage.

The colleagues at Volvo Cars have been an invaluable source of feedback and insights into this domain. Their curiosity and interest have given us continuous boosts of energy.

Finally, we would like to express our gratitude to our friends and family who have supported us with endless encouragement.

Contents

List of Figures

List of Tables

Glossary and Abbreviations

Synthetic data is artificially generated data. The synthetic data in this thesis is graphically rendered images of car occupants.

Annotation assigns pixels within an image to a certain label. Different pixels can belong to different labels.

Real data is data that is collected in the real world. In our case, this data consists of images of real car occupants.

Encoder refers to a part of the neural network that transforms input data into a compact representation while preserving important information.

Decoder works opposite the encoder, it decodes the compact information and tries to reconstruct the original input data.

Weights are the components in the neural network that get updated as the model learns.

Semantic value refers to the interpretation associated with visual content. Higher semantic value allows for a higher-level understanding of visual objects rather than just the raw visual data.

Spatial resolution refers to the granularity in an image. Higher spatial resolution implies more pixels used per unit area and thus a more detailed image.

FCNN - Fully convolutional neural network; a type of artificial neural network which is state-of-the-art in semantic seggmentation tasks.

IoU - Intersection over union; the evaluation metric commonly used in segmentation tasks.

DR - Domain randomization; a novel augmentation approach that replaces background, translates objects and implements distracting objects.

GBP - Guided backpropagation; a method to visualize which features in the input image the segmentation model extracts.

1
Introduction

Machine learning (ML) techniques, such as *artificial neural networks* (ANNs) [2], are under continuous development and have emerged as powerful tools applicable across many domains, such as the automotive industry and medicine. Successfully integrated ML can relieve the workload of personnel, who could instead focus on development and inference. However, ML also has drawbacks, one often being its inherent data requirement. Each model's performance relies heavily on the data's quality and quantity [3]. This thesis considers data consisting of images. One issue with image data is that gathering and annotating is a time-consuming and costly process [4].

Synthetic images, however, can be generated with automatic annotation, on a very large scale in silico. Producing good quality annotation such as a bounding box could take up to 35 seconds per object [5]. Since there could be many objects of interest in an image, the annotation could take several thousand hours. The cost of this process for data sets containing millions of pictures cannot be overlooked. Generating annotated data artificially takes a fraction of the time and cost compared to acquiring it in the real world. This highlights the main motivation behind this research. This thesis aims to investigate whether the performance of a model trained on synthetic data could get close to, or even exceed, a model trained on real-world data. It also investigates methods of using synthetic data as a complement to real-world data.

Volvo Cars are looking at several different use cases where the introduction of synthetic data could be useful. Some examples of such scenarios are car trajectories in crashes and car trajectories in everyday situations like roundabouts and intersections. In this thesis, the case of images of belted and unbelted vehicle occupants will be investigated. A semantic segmentation model is applied, which tries to identify the seat belt. Given this data, Volvo Cars could then evaluate whether or not a person is correctly using their seat belt.

Looking at Figure 1.1 we see a sample of the synthetic data that will be used. This synthetic data is created with the graphical engine Unity. Note that the synthetic images were created by the Internal Perception team at Volvo Cars.

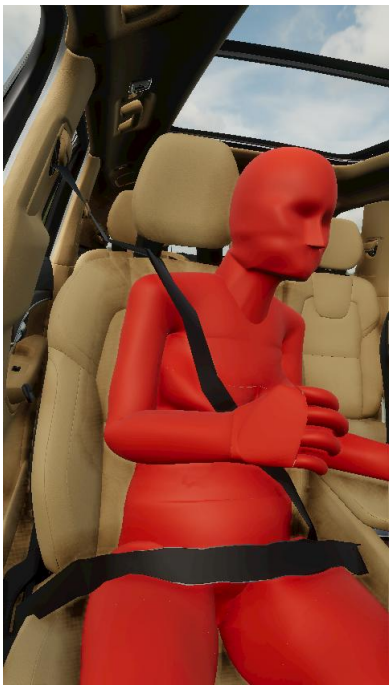


Figure 1.1: Generated image of an occupant in the front seat

1.1 Purpose

The purpose of this thesis is to measure the performance of machine learning models trained with synthetic data. The specific use case uses a segmentation model to detect pixels belonging to the seat belt class. This information could later be utilized to detect whether a car occupant is wearing a seat belt, and if so, wearing it correctly. However, this thesis will only focus on object segmentation. It is investigated how reliable models trained on synthetic data are compared to models trained on real world data or a model trained on a mixture of data. Augmentation techniques [6] that alter the images to get better test results will also be identified and implemented. Good test results in this case mean that a model trained on synthetic data achieves similar or better performance than when trained only on real data. There are numerous works to be found related to the use of synthetic data in object segmentation models [7, 8, 9, 10]. However, most of them revolve around 3D images of urban scenes, where the intended implementation is related to autonomous driving. None of them seems to use a similar type of data as the one investigated in this work.

2
Theory

This chapter will introduce the theoretical knowledge necessary for the remainder of the thesis. It contains descriptions of the different neural networks used to identify the seat belts, followed by an explanation of the technique behind distinguishing different objects in images and the architecture of the underlying neural network. The synthetic data is also described, as well as methods to improve image segmentation models. The chapter concludes by describing a method to interpret the models' results.

2.1 Image segmentation

In computer vision and image processing image segmentation has become a key topic [11]. Semantic segmentation labels an image on a pixel-wise basis and categorizes pixels into different classes. Figure 2.1 shows an image that has been segmented, via DeepLabV3, to distinguish and label different pixels belonging to certain objects with different colors. The DeepLabV3 network will be presented in Section 2.3.5. Note that the original image has been produced by the authors, using a diffusion model which is later described in Section 6.1.

Figure 2.1: Semantic segmentation of objects in a living room image

2.1.1 Adam optimizer

For our neural networks a method called Adam optimization, ADAptive Moment estimation, is used. Adam is a stochastic gradient descent method that utilizes

adaptive estimation of first- and second-order moments. The optimizer is a combination of two other popular methods, AdaGrad and RMSProp, consolidating some of their good properties. Adam deploys the adaptive learning rates from AdaGrad that improve performance on sparse gradients tasks, and it also uses per parameter learning rates from RMSProp, improving noisy performance problems [12].

The equations for the first- and second-order estimation (m_t, v_t) can be seen in equations (2.1) and (2.2). The estimates, i.e. their updates $(\hat{m}_t, \hat{v}_t)$ and shown in equations (2.3) and (2.4). New estimates are calculated for every time step t, this variable can also be seen as iterations of the algorithm.

$$m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t \tag{2.1}$$

$$v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \tag{2.2}$$

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t} \tag{2.3}$$

$$\hat{v}_t = \frac{v_t}{1 - \beta_2^t} \tag{2.4}$$

In these equations, the parameters β_1 and β_2 are the exponential decay rates for the first and second-moment estimations and $g_t = \nabla f(\theta)$ represents the gradient of the objective function. The objective function , also known as the loss or cost function is selected by the user and could for instance be the binary cross entropy function if one needs to compute some classification task.

The first-order moment represents the exponential moving average of the gradient vectors encountered during iterations. This estimate provides information about the average direction in which the parameters should be updated.

The second-order moment captures information about the magnitude of the gradients. This assists in adjusting the learning rate for each parameter based on the variance of the gradients. Both these estimates are then used to compute bias-corrected estimates. It is these estimates that are utilized to update the model parameters θ_t. The equation for the parameter update can be seen in Equation (2.5).

$$\theta_t = \theta_{t-1} - \frac{\alpha}{\sqrt{\hat{v}_t} + \epsilon} \hat{m}_t \tag{2.5}$$

The parameter values are updated in every time step t until θ_t converges [12].

2.1.2 Intersection over Union

Intersection over Union (IoU) is a metric that is used to evaluate the accuracy of object segmentation. It does this by evaluating the overlap of the ground truth and the prediction region [13]. The equation for the IoU metric can be seen in (2.6). In this equation, TP, FP, and FN denote true positive, false positive, and false negative. In this case true positive indicates the area of intersection between the segmentation mask and the ground truth. False positive is the predicted area outside the ground

truth and false negative is the number of pixels belonging to the ground truth the model failed to predict [14]. Note that for this thesis the value for a empty set is set to 1 and there is a padding of 0.000001 introduced to the nominator and denominator to avoid division by zero.

$$IoU = \frac{TP}{TP + FP + FN} \tag{2.6}$$

2.1.3 Dice

The dice metric is similar to IoU. The formula can be seen in (2.7). The dice coefficient can be interpreted as two times the intersection divided by the amount of annotated pixels in both images [15]. The rules for the edge cases described for IoU applies for the dice metric as well.

$$Dice = \frac{2TP}{2TP + FP + FN} \tag{2.7}$$

2.1.4 Binary cross entropy with LogitsLoss

The Binary cross entropy (BCE) with LogitsLoss (2.8) is a combination of binary cross entropy and a sigmoid layer (2.9) in one single class. The difference between regular BCE and BCEwithlogits is that the latter uses the raw unnormalized values outputted by the network. These values are more numerically stable than using the actual predictions [16]. In this case, p denotes the raw data or logits and is not a predicted class as in regular BCE.

$$BCEwithLogits = -[y * \log(\sigma(p)) + (1 - y) * (1 - \sigma(p))] \tag{2.8}$$

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.9}$$

The binary cross entropy, along with IoU and Dice are used in the loss function of the network, see Section 3.5.

2.2 Model architecture

The architecture used for the semantic segmentation is taken from [17], which is a master thesis where the author developed a model that tracked the posture of a car occupant and shoulder belt position. The choice of architecture used in [17] is based on a survey [11] where several state-of-the-art architectures have been tested. According to that survey, no architecture outperforms all others on all data sets for semantic segmentation but they all use ResNet 101 as an encoder. On top of the ResNet 101 encoder, a feature pyramid network was implemented, both of these concepts will be explained in Section 2.3 and 2.3.2.

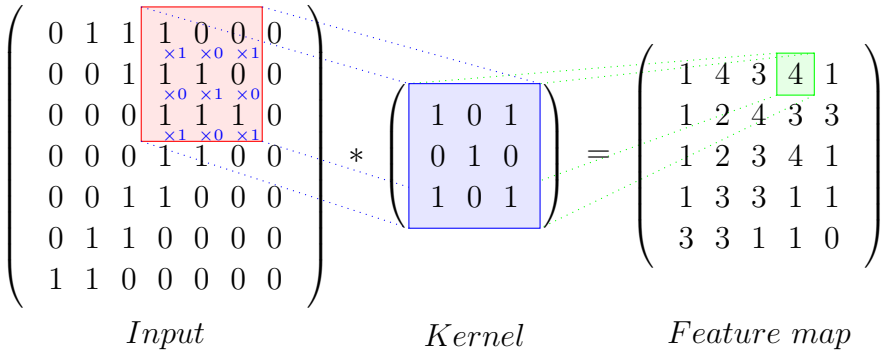


Figure 2.2: Mathematical schematic of a convolution expression

2.3 Convolutional neural networks, Encoders and Decoders

Convolutional neural networks (CNN) are deep learning architectures that have shown very successful results and are commonly used in computer vision tasks. They typically consist of three types of layers: convolutional layers, pooling layers, and fully-connected layers [11].

The *convolutional layers* are the key pieces in a CNN, it is here that the operation called convolution is performed. The convolution can be described as a linear operation where a set of weights are multiplied with an input, in this case, the RGB values of an image. These sets of weights are also referred to as a kernel. The kernels are often designed to detect specific objects like corners or edges. The output of the multiplication between the kernel and the input is mapped to a feature map. In Figure 2.3 a schematic showing a convolution operation followed by a pooling operation can be seen. Note that the kernel is much smaller than the input, which allows for sliding the kernel across the entire image, detecting any instances of the object that is searched for. Furthermore, these convolutional layers can be stacked on top of each other. The purpose of stacking layers is that each layer has a different objective, like detecting edges and corners, and also its own kernel.

When traversing deeper within the network, the low-level features, such as edges and lines, are used to extract features of higher order to express shapes [18]. This process continues until objects such as faces or animals can be extracted. Note that each value in a feature map is also passed through an activation function such as the sigmoid function, see Equation (2.9). Once the image has propagated through the network the results are evaluated using a loss function such as BCEwithLogits. Then the weights in each kernel get updated through *backpropagation* [19], optimizing the network to detect the features of interest.

Next is the *pooling layer*, which reduces the spatial dimensions of the input and thus reduces the number of parameters. Like the convolutional layer, the pooling layer slides a kernel across the input but instead of computing weights, it uses an aggregate

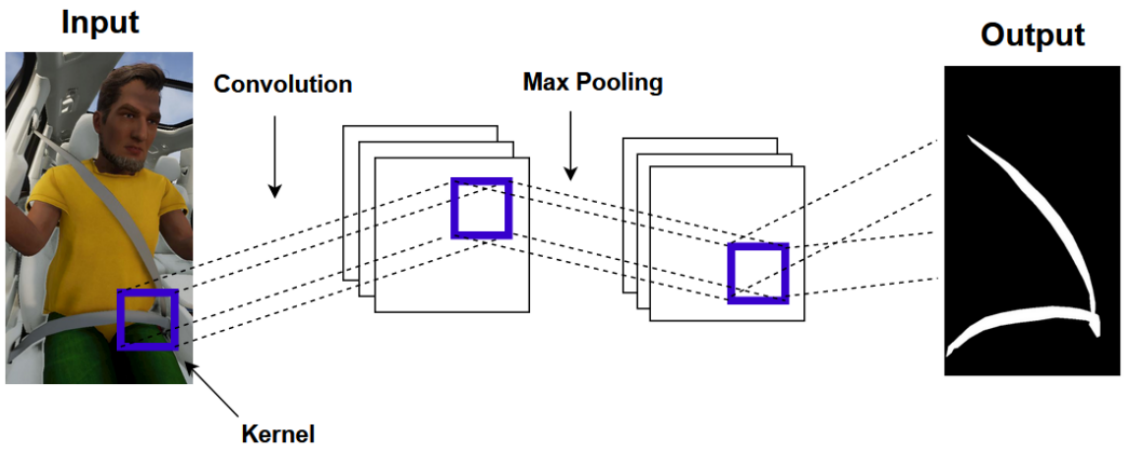


Figure 2.3: Image processed by a convolution operation and then followed by a pooling operation. The output of the image is a semantic segmentation

function such as average or max value. Pooling layers are used to reduce complexity, prevent overfitting, and extract the most prominent features in the input.

In fully-connected layers, each node in the output layer is connected to a node in the previous layer. This layer is used for classification based on the previous layers [20]. In Figure (2.2) a mathematical representation of a convolution is shown. Note that the feature map is the scalar product between the input and the kernel. For a pooling layer, the feature map would be represented by the maximum value of the input.

In segmentation tasks an *encoder-decoder* schematic is often used. Both the encoder and the decoder are CNNs. The objective of the encoder is to take an input sequence and convert it into a vector representation. An input sequence in this case could be an RGB image with dimensions *[channels, height, width]* $= [3, 256, 256]$ where each channel represents one color and the height and width are the dimensions of the image. The dimension of the vector could in this case be represented by a $\mathbb{R}^{512}$ dimensional vector where all color channels and each grid has been concatenated into one vector. More specifically, the color dimensions are reduced to one dimension and the width and height are compressed. The vector representation is also known as a hidden or latent representation of the input and is in a more condensed form. This way the network still keeps the information about key features while ignoring redundant information [21]. The encoders that are used consist of several versions of the ResNet architecture.

The decoder reads the compressed input from the latent vector and produces an output which is compared with the target and information is sent to the network for computing loss. In computer vision, the decoder can also be a tool for reconstructing the input images based on the hidden space [21].

This thesis is focused on semantic segmentation performed on a pixel-wise level, i.e.,

each pixel is classified rather than classifying the image as a whole. In these cases, it is useful to deploy a fully convolutional neural network. What differentiates these networks from the regular CNN is that there are no fully-connected layers. Instead of the dense layers that have a fixed output dimension, fully convolutional networks use a $1x1$ convolutional layer at the end so that the output dimension can match that of the input [22]. As can be seen in Figure 2.3, the output is an image representing the semantic segmentation rather than a label or a class.

2.3.1 ResNet

ResNet or residual network is a certain type of neural network that is mostly used for the task of image recognition [23]. The ResNet networks usually come with a number suffix that indicates the number of layers, for example, the ResNet101. What is iconic for the ResNet is the *residual block*, also known as a *skip connection*. The residual blocks aim to eliminate the vanishing/exploding gradient problem that can otherwise occur in deep neural networks. This block provides an alternate route for the gradient, giving it the option to bypass some hidden layers ensuring that the next layer will perform at least as well as the previous layer [23]. Figure 2.4 shows the schema of a residual block. Note the identity mapping x added on top of the outputs of the layers, preventing the possibility to pass on extremely low or high gradient values stemming from inside the hidden layers [24].

The skip connection first introduced by the ResNet architecture solves a common problem in deep learning. A naive approach would be that if the depth of the network is increased, the model should be more robust and yield stronger performance, for example adding more layers for more features. However, it turns out that the error rate actually saturates and then increases if the network gets too deep [24].

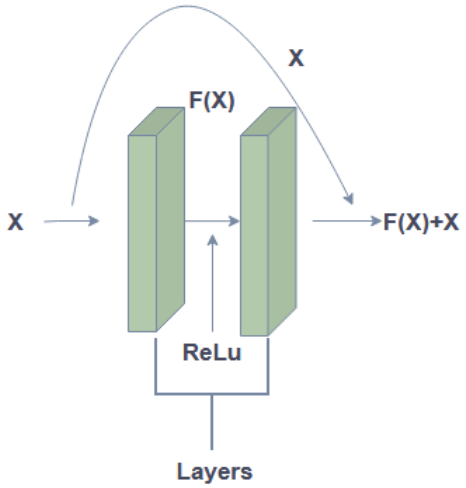


Figure 2.4: Example of a residual block. The curved arrow shows the shortcut that skips the intermediate layers [1]

2.3.2 Feature pyramid network

A *feature pyramid network* (FPN) is a neural network that is used for object detection. This model particularly excels at detecting objects at different scales [25]. For example, in an image, a human hand can be of different sizes depending on where it is located in a depth perspective [26] and so it can be difficult for a network to correctly identify the hand in such scenarios.

The FPN addresses this issue by creating a pyramid of feature maps with different spatial resolutions. This allows objects of different scales to be detected. To achieve this the architecture consists of two main pathways, the top-down pathway and the bottom-up pathway [27].

The bottom-up pathway is a feed-forward network that generates feature maps of different resolutions from the input image. When going up, the spatial resolution will decrease and the *semantic value* will increase. This process is also known as *downsampling*. Semantic value and spatial resolution are two concepts revolving around the interpretation of visual content. Semantic value implies the recognition and understanding of objects and scenes, whereas spatial resolution is the granularity present in an image, i.e., the number of pixels used per unit area. The top-down pathway increases the spatial resolution of the feature maps provided by the bottom-up pathway. This process is known as *upsampling*. The FPN also uses lateral connections, sending information between the bottom-up and top-down pathways without passing it through each layer of the pathways. These connections blend the upsampled feature maps with the corresponding downsampled feature maps, combining global context and fine-grained details to better capture objects at different scales [28].

In Figure 2.5 a schematic of the feature pyramid network can be seen. Note the small squares in the top-down pathway indicating the predictions made for each level, this shows the predictions made for different scales and highlights the main property of the FPN network.

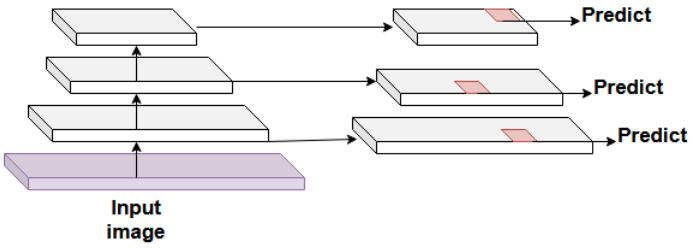


Figure 2.5: A schema of the feature pyramid with the bottom-up pathway to the left and the top-down pathway to the right

2.3.3 Unet

The Unet architecture was originally developed for biomedical image segmentation. The network consists of two paths, *the contraction part* and *the symmetric expanding part* [29]. The contracting part captures the context of the image by using a series of fully-connected convolutional and max pooling layers. In this process, the spatial information is reduced and the feature channels are increased, abstracting the input image to capture the most relevant information [30]. Spatial information refers to the arrangement and relationship between objects, and increasing the feature channels means that the network can learn more complex features and patterns. Detecting edges is an example of one feature. The expanding part decodes the data from the contracting part to reconstruct the input image, similar to an encoder-decoder structure. Furthermore, it uses skip connections to localize where in the images objects are located [31]. In Figure 2.6 an example of the Unet architecture can be seen. Note the U-shape which has given the architecture its name.

The Unet++ is an architecture that was built to improve the existing Unet architecture. While it retains much of the architecture it has one key difference. Instead of only implementing skip connections between the corresponding encoder-decoder layers it also implements *nested skip connection* [32]. These connections make it possible for the network to connect several different levels. The advantage of this is that the network can access features from different scales and different levels of abstraction. For a Unet++ the arrows in Figure 2.6 can access all levels and not only the layers on the same level.

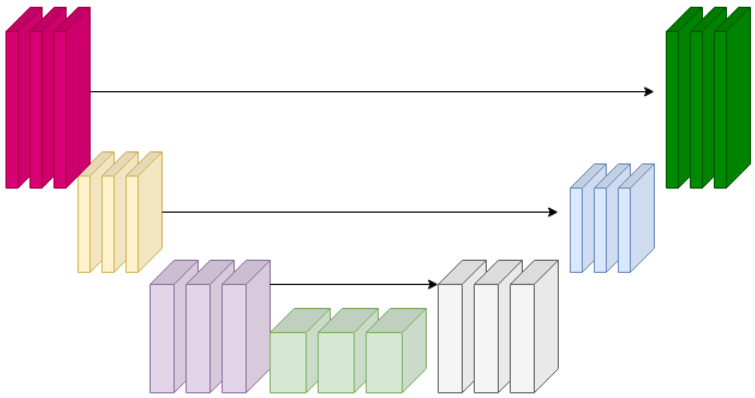


Figure 2.6: Neural network architecture of Unet. In the picture each layer is marked and it also shows the contracting part which points down and the expanding part going upwards in the figure. These two pathways constitute the U shape giving the architecture its name. Also, note the skip connection between the layers

2.3.4 LinkNet

The LinkNet uses an encoder-decoder schema similar to the Unet, but there is also a connection between each encoder-decoder block. The objective of this connection is to recover spatial information that could otherwise be lost in the downsampling of the encoder. Furthermore, LinkNet uses batch normalization between each convolutional layer [33]. Batch normalization is a technique that standardizes the input to a layer for each *mini-batch*. In this context, a mini-batch is a small portion of the training data. Normalizing the output from an activation function in a previous layer means that the spread and distribution of subsequent layers will, potentially, not be as dramatic. This increases training speed and also reduces generalization error [34]. In Figure 2.7 the architecture for the LinkNet can be seen.

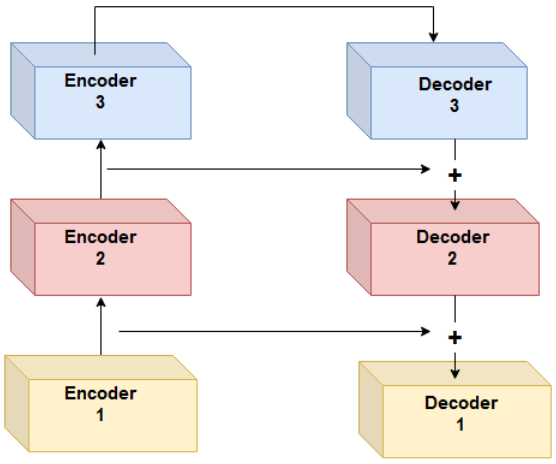


Figure 2.7: The schematic of the LinkNet architecture

2.3.5 DeepLabV3

The DeepLabV3 is a fully convolutional neural network model introduced by Google for the task of image segmentation. The main characteristics of the network are the implementation of *atrous convolutions* and *atrous Spatial pyramid pooling*. These techniques are designed to tackle two common pitfalls of image segmentation respectively. The term "*atrous*" comes from the French word "atrous," which means "with holes" or "with gaps". Atrous convolutions are also known as dilated convolutions.

The DeepLabV3 architecture attempts to solve two issues in image segmentation. The first issue stems from downsampling data through the encoder. While this process retains essential features with smaller data requirements, spatial information can be lost. This makes the network efficient at predicting the labels of objects but the information of where objects are is reduced. The second issue can arise in images that have objects of the same label but with different resolutions. Since the receptive field increases when traversing deeper into the network, larger objects get detected easier, whereas smaller objects become harder to recognize [35]. The receptive field is the part of an input that after convolution results in a feature [36].
The Atrous convolution can be seen in Figure 2.8. The idea with this convolution filter is to dilate the kernel by inserting gaps. With this, the receptive field gets increased and the amount of parameters remains fixed. This allows for control over the field of view and thus the resolution at which features are detected [37].

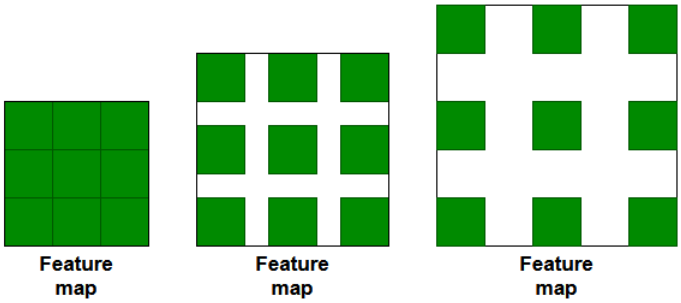


Figure 2.8: Different atrous convolution filters with a kernel size of 3×3. Moving from left to right the dilation of the filter is increased

The atrous spatial pyramid pooling uses atrous convolutions and applies this to an input feature map in parallel. This is an effective way to classify regions at arbitrary scales. The technique resamples features at different *atrous rates*, i.e., how much the kernel is diluted. It then uses image pooling and batch normalization [37].

Like Unet, DeepLabV3 also has a successor, this one is called DeepLabV3+. The DeepLabV3+ extends the DeepLabV3 architecture by adding a decoder module to improve segmentation along object boundaries. The decoder adds skip connections, 1x1 convolutions and upsampling [38].

2.4 Synthetic data

Synthetic data is data that is not collected in the real world but instead generated artificially in some way [39]. The synthetic data used in this thesis is mainly images created in the rendering software Unity (version 2021.3.2f1). The data is produced by the Internal Perception team at Volvo Cars which have worked closely together with the authors throughout the thesis to create the datasets. To gain more diverse datasets, the production of the data has been steered via regular feedback to include such things as different light conditions, shading, and placement of the occupants. These changes could also be isolated so that only one variable changes at a time while keeping others fixed, resulting in a high level of scrutiny for variables in each image.

In Section 3.2.2 the different prompts and requests that have been made to Internal Perception are described in more detail, and how the data evolved during the thesis is also shown. As the images were rendered in the software, each pixel was also annotated. This is what highlights the advantage of synthetic data. Creating a bounding box for an object in an image could potentially take around 35 seconds per object [5] and a pixel-wise annotation potentially takes 10 times as long [40]. If there are

multiple objects per image and the data set is very large, the annotation will be a very tedious and costly process. Furthermore, since the RBG values of each pixel are determined by the software, this also means that annotation will be extremely precise on a pixel-wise basis, something that would be very hard to obtain with human annotation.

The one drawback encountered while creating the synthetic data was the interplay between the seat belts and the models. To align the seat belt along the body of the model the Internal Perception team had to manually adjust the settings of physical collisions, also known as colliders in Unity. In short, the colliders describes the interplay for physical collisions between different objects. The collider component is invisible and requires precision to work well [41]. The necessity for correct colliders meant that only produce a few different synthetic human models could be produced, but for those models, an arbitrary amount of data could still be produced.

2.4.1 Data augmentation

Data augmentation is a technique that artificially inflates a data set while also preserving the labels of the original data [6]. Transforming the input image helps the model to reduce generalization errors by forcing it to learn invariant representations. Data augmentation has been shown to lead to great improvements in deep learning models, both for performance and generalization. It is particularly useful when training data are scarce and the model fails to grasp the entire variance of the data distribution from the training set [42]. While there are several different ways to augment the training data, this thesis has focused mainly on *photometric* and *geometric augmentations*.

Photometric augmentations are transformations that alter the RGB channels of the image. It does this by changing each pixel value to a new value that is defined by the augmentation technique. Geometric augmentations map individual pixels to new destinations and thus change the geometry of the image [6]. The augmentations are implemented using the `albumentations` Python library, which has a plethora of different techniques to use [43].

2.4.2 Bridging the domain gap

One important issue that arises when using synthetic data is minimizing the *domain gap* between synthetic and real world images [10], i.e., the visual difference. This could mean generating synthetic data which is more photorealistic. In this specific work, the photorealism is partly limited by the graphics engine, which is reason to believe that the usability of synthetic images will improve as the engines which produce them improves. There are other methods, such as *domain randomization* and augmentations, as described in Section 2.4.3 and Section 2.4.1 respectively, which also work towards bridging the domain gap.

2.4.3 Domain randomization

The main idea of domain randomization is, instead of training the model in a single environment, it is exposed to a myriad of different environments. With enough

variability in the training images, the real-world testing images will appear as just another variation of the training [44].

Inspired by [45] new images were rendered, with an atypical background and also with *flying distractors*. In other words, the typical car interior could be replaced with another background. The flying distractors are objects such as circles, rectangles, and also seat belts. Correct annotation was added for flying distractor seat belts. The objective is to increase variability and also make the model more robust to identifying the relevant object regardless of the rest of the image, making the image, or more specifically the seat belt, invariant to its environment. In Section 3.3.2 some image examples of the domain randomization are shown.

2.5 Interpretability

Machine learning models are often so-called *black boxes*. This could mean that they are too complex for human understanding, e.g., deep neural networks, or simply that they are hidden for proprietary reasons. The models investigated in this thesis falls under the first category. Understanding a model's predictions can be very important for further development, e.g., by understanding and correcting the reasons for bad predictions. In the case of *image classification models*, the common methods of explanation focus on the model's ability to identify relevant features in the input images, e.g., given an input image of a cat, the model predicted the correct class "cat" by identifying the features belonging to the cat. A common approach to visualize this explanation method is through *saliency maps* [46], which evaluates each pixel's impact on the probability of a specific class and visualizes it as a heat map. This approach is intuitive when explaining classification models. In the case of segmentation models, however, this explanation method is not as intuitive, since the prediction itself shows the highlighted features.

2.5.1 Guided Backpropagation

By implementing *Guided Backpropagation* (GBP) [47], a segmentation model can be interpreted by visualizing what it has learned. While in the original paper the implementation of GBP was performed on a classification model, it has since been successfully adapted to a segmentation model [48]. The method is an extension of the standard *backpropagation algorithm*. As a first step a *forward pass* is performed, i.e. an input image is propagated through a trained network up until the last convolutional layer. For each layer, the ReLU activation function is applied to the feature maps, such that all features with negative values are set to zero. Let f_i^l be the i:th feature in the l:th layer feature map, then the forward pass process can be described as

$$f_i^{l+1} = \text{ReLU}(f_i^l) = \max(f_i^l, 0) \tag{2.10}$$

Then, starting from the last convolutional layer, a backpropagation is performed, where the gradients with respect to the activations are calculated and then propagated backward using the chain rule. Again the ReLU function is applied, only allowing

positive gradients to propagate backward. Let b_i^l be the i:th gradient in the l:th layer, then the modified backpropagation can be described as

$$b_i^l = (f_i^l > 0)(b_i^{l+1} > 0)b_i^{l+1} \tag{2.11}$$

where the expressions in parenthesis are logical operators, yielding 1 if true and 0 if false. The resulting gradients at the input layer will represent the importance of each pixel in relation to the output [46]. If negative gradients were included in the visualization, they can result in noisy images [47]. This was the motivation behind the use of GBP.

3
Method

This chapter explains the different methods used to evaluate the usability of the synthetic data, which includes both evaluating the synthetic data on its own, as well as using it as a complement to real-world data. It explains the different datasets, both containing synthetic and real-world data, that are being evaluated. The implementations of augmentations and DR are also described.

3.1 Choosing the model

All models tested in this work are imported from the PyTorch software library `segmentation_models_pytorch` [49], where numerous encoder/decoder setups are implemented. A list of all setups used in the project can be seen in Table 3.1. Note that each decoder is compatible with each encoder, and all combinations are explored. To get a reference for which network was used and how the architecture was implemented, see Section 2.2.

Encoders	**Decoders**
ResNet18	FPN
ResNet34	DeepLabV3
ResNet50	DeepLabV3+
ResNet101	Unet
ResNet152	Unet++
	LinkNet

Table 3.1: List of all encoders and decoders used

Regardless of which architecture a segmentation model has, it takes an image of a car occupant as input and predicts which pixels, if any, contains the seat belt. The outputted *prediction* has the format defined below.

Definition: Given an input image I with dimensions (m, n), the *prediction* outputted

by the segmentation model is a matrix A of size $m \times n$ with elements a_{ij} such that

$$a_{ij} = \begin{cases} 1, & \text{if pixel at (i,j) in } I \text{ is predicted as seat belt} \\ 0, & \text{else} \end{cases} \tag{3.1}$$

Note that Definition 3.1 regards binary predictions, e.g. when only the seat belt is segmented. The case where multiple classes are predicted, which is also explored, is explained in Section 3.3.4.

3.2 Data

The available data consist of a vast amount of synthetic data (as described in Section 3.2.2) and a limited amount of real world data (Section 3.2.1). The synthetic data was provided by Internal Perception at Volvo Cars, who were given continuous feedback by the authors, attempting to decrease the domain gap (Section 2.4.2). The used real data is provided by Volvo Cars. It was originally collected for the master's thesis [17] which, in regards to seat belt segmentation, laid the foundation for this work.

3.2.1 Real data

The real data was collected by placing a video camera aimed at the front passenger seat. The data consist of a total of eleven test persons, where 40-60 frames of each person have been collected from the respective video recordings. Attempts have been made to generate a natural variation in the data, i.e. recording during different light conditions, with the passengers both belted and unbelted, wearing the belt correctly and incorrectly, as well as recording empty passenger seats.

The data is divided into two subsets (see table 3.2), one consisting of images of four test persons which is used as a test set, and another one consisting of images of the remaining seven test persons which is used for integration in training, as described in Section 3.3.3. This split was made to ensure that the test set only consists of, for the model, previously unseen images.

Use	# images	# test persons
Testing	222	4
Training	341	7

Table 3.2: Division of available real data

3.2.2 Synthetic data

The synthetic data consists of synthetically rendered images, produced in the Unity engine by Internal Perception at Volvo Cars. The synthetic data has evolved during

the work on this thesis. These steps of evolution will hereon be referred to as *generations*, of which six have been introduced (see Figure 3.1 below). This evolution is a result of continuous feedback from the authors. In part, the feedback has been inspired by poor predictions, e.g. increasing the portion of images of non-belted occupants to improve the predictions in such scenarios. Feedback has also been results of trying to bridge the domain gap (see Section 2.4.2) between the synthetic and real data, such that each new generation aims to add variation inspired by real-world natural variation. These natural variations regard the color of clothing, variations of light (such as angle of incidence and intensity), car interior color, seat belt color, seat belt placement (such as over and under shoulder), occupant's placement (such as different rotations in seat and arm positions which obstruct the visibility of the seat belts). Some of these variations are visualized in Figure 3.2 below.

The sixth and last generation is a result of the development team obtaining a new software license, leading to greatly improved details on the occupants, adding faces, gender, hair, and actual clothing objects. By adding labels to more features in the images, such as shirts, pants, and skin, the impact of context is explored, as described in Section 3.3.4. The details of specific generations are summarized below.

First generation - Include color variations in car interior, seat belts, and occupants. The occupants are single-colored dummies.

Second generation - Improved seat belt colliders, i.e. the way the seat belts interact with the occupants made more realistic.

Third generation - More car interior colors, especially colors similar to the seat belt. Includes images with unbelted occupants, as well as images without occupants. Added multiple colors on occupants, simulating clothing and skin.

Fourth generation - Added light variations and shadows to simulate sunlight.

Fifth generation - Added more seat belt colors.

Sixth generation - Greatly improved the appearance of occupants, adding clothing, faces, and hair. Also added multiple labels as explained in Section 3.3.4

3.3 Training

The applicability of the synthetic data is explored with several different approaches. Using a fixed synthetic data set during training, the network, i.e. encoder/decoder combination as described in Section 3.1, with the highest performance when tested on the real data test set is explored.

Training is performed on all generations of synthetic data separately, as well as on data sets with mixed generations attempting to further increase variation. Different

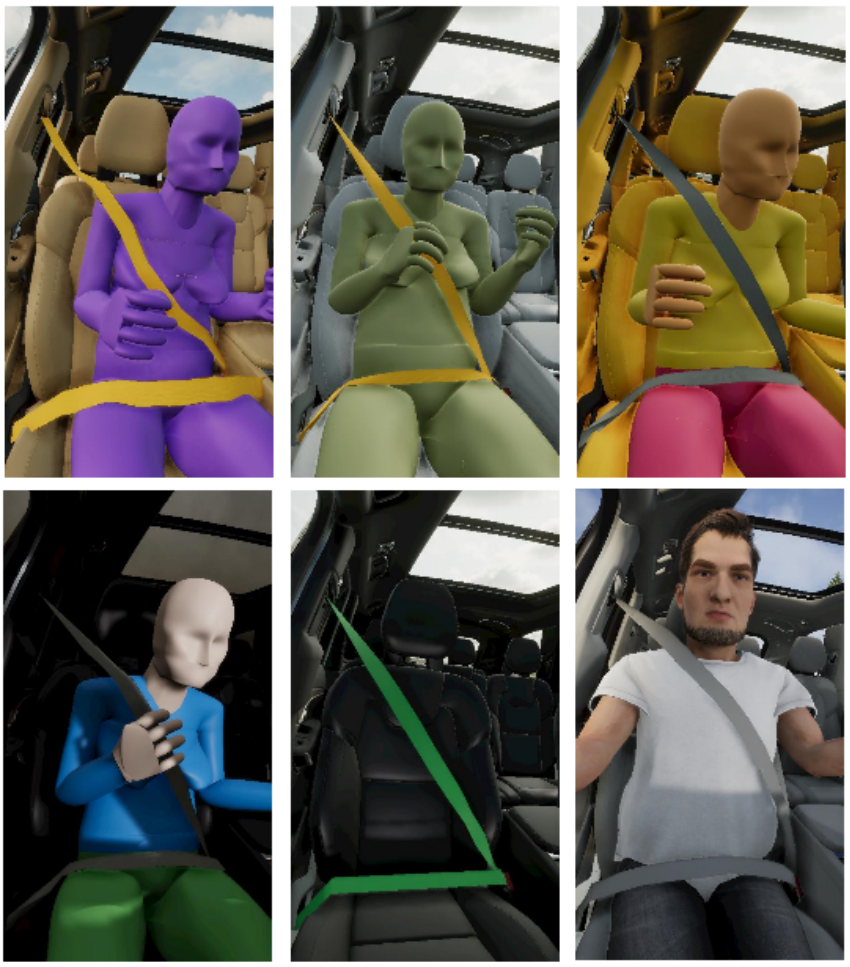


Figure 3.1: Six generations of the synthetic data in chronological order, going from left to right, starting at the top. The fifth generation's belt color addition is here visualized with an empty seat, which was added in the third generation

(a) Angle of incidence of light.

(b) Temperature of light.

(c) Intensity of light.

(d) Occupant's rotation on seat.

Figure 3.2: Visualization of four naturally inspired variations, implemented on the sixth generation data

Name	value	Name	Value
batch size	4	decoder learning rate	0.00025
encoder learning rate	0.0002	epochs	50
threads	20	early stopping patience	50
layer weight decay	0.00003	optim factor	0.5
optim patience	6	decoder dropout	0
encoder depth	5	adam weight decay	0.0003

Table 3.3: Hyperparameters used for training the neural networks

combinations of augmentations are explored, as described in Section 3.3.1, as well as the concept of domain randomization, as described in Section 2.4.3 and Section 3.3.2. The integration of real data in the training process is explored with two different approaches; by mixed data sets of real and synthetic data, and by pre-training the model on synthetic data before training it on the real data, as described in Section 3.3.3.

For training with synthetic data a random $80 - 20$ split is used, with 80% used for training and 20% used for validation. When training with real data a $70 - 30$ split is used instead. This follows from splitting by test persons, such that the training set consists of images of five test persons and the validation set of images of the remaining two test persons. The purpose of this split is to increase the model's ability to generalize towards previously unseen occupants.

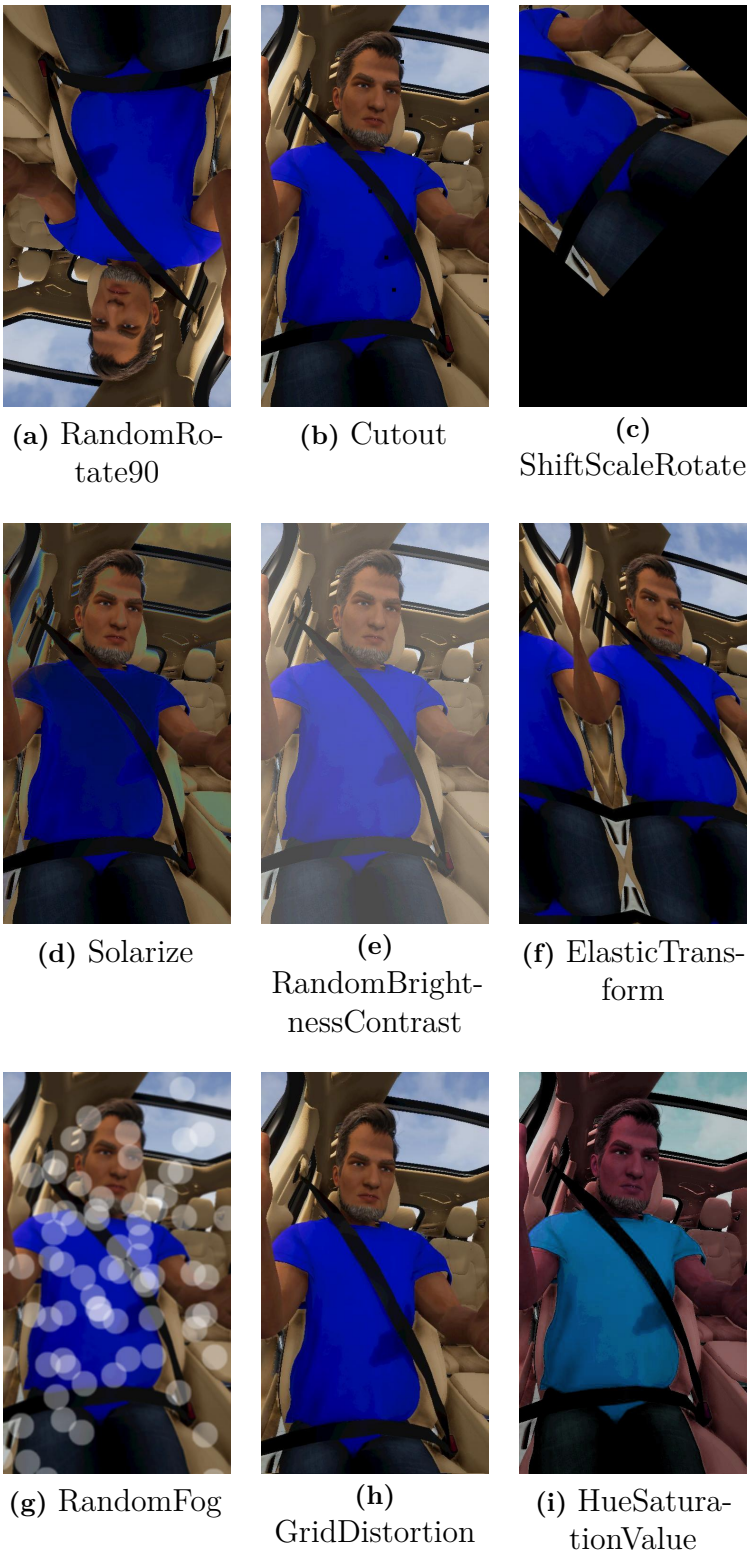
The hyperparameters used in training can be seen in Table 3.3. These parameters are largely inherited from a previous implementation [17], except for the encoder and decoder learning rates which are decreased by a factor of 10. This was inspired by the intuition that higher learning rates decrease the impact of using pre-trained models.

3.3.1 Augmentations

Experiments are done with numerous different augmentations (Section 2.4.1), and combinations thereof, to improve the model's accuracy. All augmentations are loaded from the python library `albumentations`. The different augmentations are listed below in Table 3.4 and are visualized in Figure 3.3.

Augmentation	Explanation
RandomRotate90	Rotates the image by a random integer multiple of 90 degrees.
RandomCrop	Crops out a portion of the image of specified dimensions.
Cutout	Cuts out a specified number of squared holes of specified size from the image, i.e., turning them black.
ShiftScaleRotate	Shifts the image, rescales it, and rotates it, following specified parameter values.
Solarize	Inverts all pixel values above a specified threshold, i.e., the RGB-value x becomes $255 - x$ given that x is above the threshold.
RandomBrightnessContrast	Randomly changes the brightness and contrast of the image.
ElasticTransform	Elastically deforms the image as described in the paper from P.Y. Simard et al. [50].
RandomFog	Simulates circles of fog in the image.
GridDistortion	Spatially distorts the image.
HueSaturationValue	Randomly changes hue, saturation, and value of the image. Also known as the three components of color [51].

Table 3.4: All augmentations that were applied together with explanations



(a) RandomRotate90 (b) Cutout (c) ShiftScaleRotate

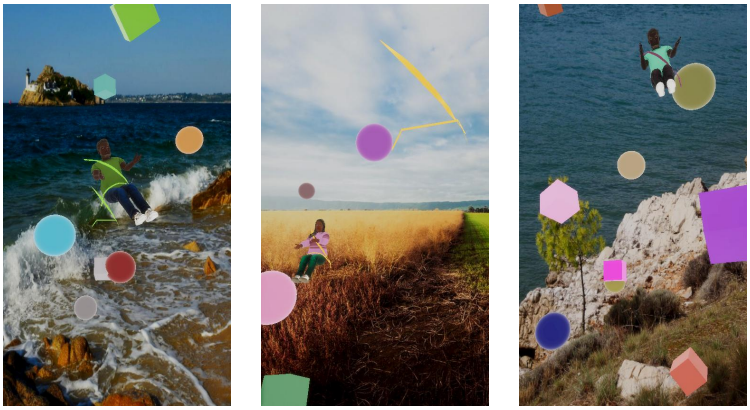
(d) Solarize (e) RandomBrightnessContrast (f) ElasticTransform

(g) RandomFog (h) GridDistortion (i) HueSaturationValue

Figure 3.3: The different augmentations implemented in training

3.3.2 Domain randomization

Following the concept of domain randomization, as explained in Section 2.4.3, new images, where the person as well as the seat belt was cut out of the original image, were created. Next, the person within the image was shrunk and translated. Different background was also introduced, a as well as some flying distractors, i.e., geometric objects such as circles and rectangles, and some are even seat belts (see Figure 3.4). When there are flying distractors depicting seat belts, such as in example **b)** in Figure 3.4, they are also annotated as seat belts.



(a) **(b)** **(c)**

Figure 3.4: Images with domain randomization

3.3.3 Pre-training

The Python software library `segmentation_models_pytorch` allows for using encoders that are pre-trained on *ImageNet*, a large image database designed for use in visual object recognition software research, as opposed to randomly initialized weights. Both these approaches are explored. Similarly, when training a model with real data, both using a model which is pre-trained on synthetic data, and one which only has an encoder pre-trained on ImageNet, are explored.

3.3.4 Inclusion of context

The idea is mainly inspired by the intuition that a model should be better at detecting the seat belt if it also learns to detect the objects which interact with the seat belt. It is also inspired by previous work [52, 53], where the inclusion of context has proven to increase performance. There is a fundamental difference in which role the context has in this work, compared to the previous work mentioned, namely that there is very little contextual variation in the data explored in this thesis. The presence of a certain class does not influence the probability of another certain class being present.

To evaluate whether context can be added to improve prediction accuracy, the model is both trained using only binary labels, i.e. 1 for pixels containing a seat belt and 0 for all other pixels, as well as using multiple labels (see Figure 3.5). Training

with multiple labels utilizes one of the most important advantages of synthetic data images; that multiple classes can be annotated simultaneously automatically.

When training with multiple labels the annotations used instead have the dimensions (k, m, n) where (m, n) are the image dimensions and k is the label dimension. So, given $k = 13$ labels, each of the 13 channels is in itself a binary annotation, annotating a different object. The loss function used during training is simply the mean of the losses for each label.

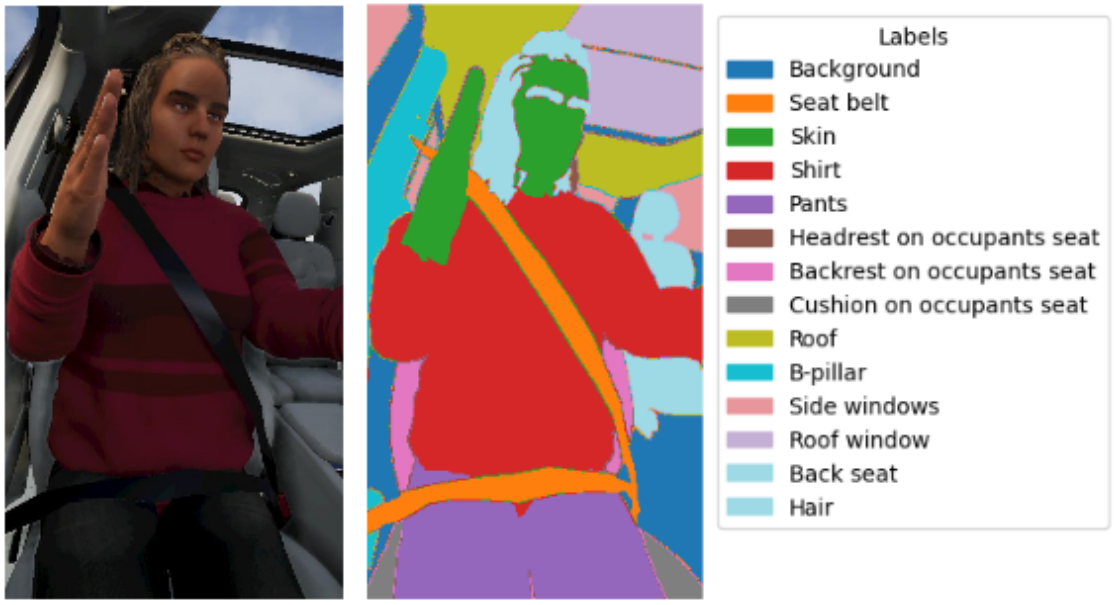


Figure 3.5: Synthetic image with more labels added. Each label is marked with a color code

3.3.5 Mixed data sets

In contrast to the pre-training explained in Section 3.3.3, where a model is trained on only synthetic data first and then trained on only real data, the idea of training on mixed data sets which contain both real-world images and synthetic images is explored. The result from training with the real training set mentioned in Section 3.2.1 is used as a benchmark. The benchmark is then compared to the results from training on various mixed data sets, in each of which synthetic data has been added to the real data set, creating different proportions. The proportions are listed in Table 3.5.

Note that the real data is constant, i.e. the property that separates the data sets is the amount of synthetic data.

Dataset	Synthetic data [%]	Real data [%]	# images
1	50	50	682
2	40	60	853
3	30	70	1137
4	20	80	1705
5	10	90	3410

Table 3.5: The proportions of synthetic and real data in the mixed datasets. Each dataset contains 341 real-world images

3.4 Explainability of the segmentation models

As a step towards understanding the mechanics of a segmentation model, one can evaluate which features of the input image the model finds most important when it is making its prediction. To this end, two different methods are explored; *Simplistic saliency map* (Section 3.4.1) as formulated by the authors, and GPB (Sections 2.5.1 and 3.4.2).

3.4.1 Simplistic saliency map

Given that the model accurately segments the seat belt, one wants to understand whether it does so by actually detecting the seat belt, or just by learning where the seat belt is likely to be positioned. To this end, a simplistic method has been implemented, heavily inspired by saliency maps, as mentioned in Section 2.5. A single image of a belted occupant is chosen as a base image and fed to the model, resulting in a prediction that is later used as a benchmark. A new data set is generated, consisting of perturbed versions of the base image. Each perturbation entails the removal of a different 8×8 pixel square. The perturbed data set is fed to the model and the distance D between the resulting predictions and the benchmark is calculated.

Definition: Let A and B be two predictions (3.1) with dimensions (m, n) and elements a_{ij}, b_{ij}. Let A_i denote the i:th row of pixels in A, i.e. $A_i = [a_{i1}, ..., a_{in}]$. The *distance between A and B* is defined using the L_1 vector norm.

$$D_{AB} = \sum_{i=0}^{m} ||A_i - B_i||_1 \tag{3.2}$$

The images used in this work have dimensions $(848, 480)$. If split into a grid of 8×8 pixel squares there are $848/8 \cdot 480/8 = 106 \cdot 60 = 6360$ squares

So with perturbations of size 8×8 pixels, one base image results in 6360 unique perturbed images. The result is visualized with a heat map representing the resulting 106×60 grid, i.e., the color in each square represents the distance between the benchmark and the prediction of the related perturbed image. This heat map will

indicate which areas of the base image have the largest effect on the prediction, which in turn indicates whether the model focuses on the object of interest or a surrogate such as the typical geometrical position.

3.4.2 Guided Backpropagation

To visualize the results of the GBP, as described in Section 2.5.1, the PyTorch library `medcam` [54] is used. The gradients are derived from the last convolutional layer in the network, resulting in a *gradient map*, mapping gradients to related pixels, after backpropagation to the input layer. This gradient map is visualized by scaling the values, converting them to a color map, and overlaying it on the input image. The regions with higher importance values will appear more prominent in the visualization.

3.5 Evaluation

Every model is evaluated based on its predictions on the same real-world dataset, i.e. a test set consisting of 222 images (see Table 3.2). The evaluation metric, or the accuracy, of a single prediction, is the IoU between the input image's annotation mask and the prediction. Each model is evaluated with the mean over IoUs related to the predictions of all images in the test set.

All possible encoder/decoder pairs are evaluated and compared when trained on a fixed synthetic dataset, consisting of images from all generations except the first generation. The mixed-generations dataset was chosen since it gave a high relative prediction accuracy. The different generations of synthetic data, as well as combinations of them, are evaluated by training separate models, with a fixed network architecture, and comparing each model's prediction accuracy.

When evaluating the impact of pre-training, only the six top-performing network architectures are chosen to avoid cluttered results. When evaluating ImageNet versus random initialization the models are trained on a fixed synthetic dataset, comparing their respective prediction accuracy. When evaluating the use of synthetic data in pre-training, three separate evaluations are performed and compared; the models when trained on only a fixed synthetic dataset, when trained on only the real-world dataset, and when trained on first the synthetic dataset and then on the real-world dataset.

Regarding the explainability of the segmentation models, both the simplistic saliency map and GBP are implemented, as described in Sections 3.4.1 and 3.4.2 respectively. These methods are also used as a further evaluation of the synthetic data; by training a model on only synthetic data and exploring whether it learns to extract the relevant features from the real data.

4
Results

This chapter presents the results of the thesis. The results mainly consist of mean IoU values, see Section 2.1.2, that range between $[0, 1]$ where 1 indicates a 100% prediction accuracy. The improvements in the network performance have been incremental and the results from different methods will be presented separately.

4.1 Neural network

The effect of altering the network architectures has been investigated. A range of different encoders and decoders was used to define the architectures, as listed in Table 3.1. Note that throughout this process the dataset and hyperparameters were fixed. In Figure 4.1 the performance of each separate model composition has been plotted. As the results suggest, there is no noteworthy performance to be gained by choosing one network architecture over another. It has also been observed that the choice of synthetic dataset affects which architecture performs best. The best-performing model found during the work on this thesis is a Unet++ decoder with a ResNet 34 encoder, trained on a synthetic dataset containing generations 2-6, as can be seen in Figures 4.4 and 4.5. From hereon all results, if nothing else is stated, use this architecture.

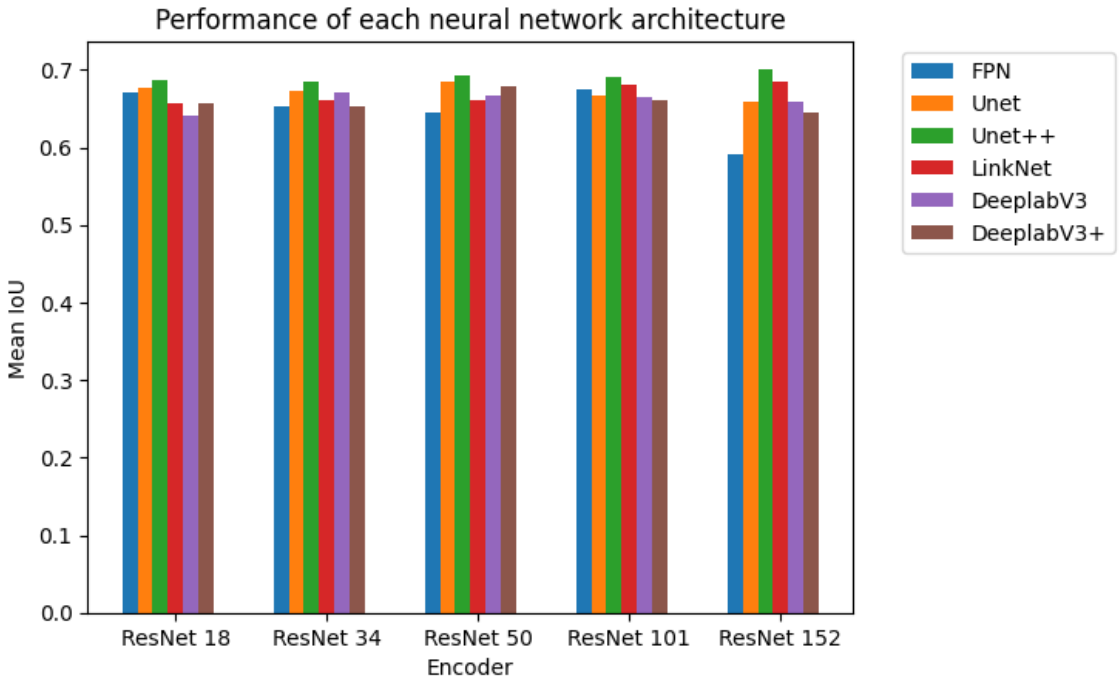


Figure 4.1: Mean IoU performance for each neural network architecture. Each different color represents a decoder that has been coupled with an encoder listed on the abscissa. The dataset used for this experiment contains generations 3, 4, and 5

4.2 Data sets

For the datasets, both stand-alone generations, as well as mixtures of several generations, are evaluated. The different generations are described in Section 3.1. In Figure 4.2 it can be seen how well a model performs when trained on different synthetic datasets. In the graph, it is shown that the datasets which contain multiple generations result in higher performance than any of the generations on their own. It is also noticeable that each generation performed significantly better when augmentations, as described in Section 3.3.1, are applied. Furthermore, with augmentations, generations 3 and onward performs on par with or better than the real data without augmentations. The best performance is obtained when training on datasets with mixed generation.

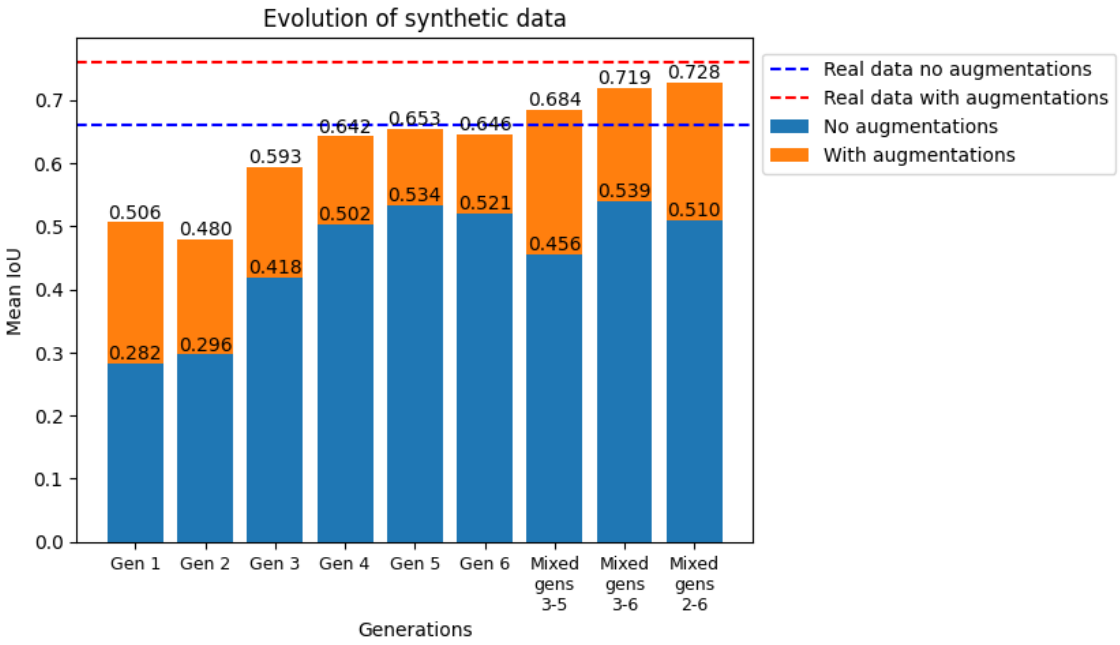


Figure 4.2: Performance of a model when trained on different synthetic datasets. All six synthetic generations are evaluated separately, as well as three mixed sets. The performance of the model when trained on real data is used as a benchmark. The impact of using augmentations is also incorporated in the graph

The results of training a model on datasets that contain both synthetic and real data are visualized in Figure 4.3. The results suggest that adding synthetic data to the real-world training set increases the performance of the model. When using an encoder with randomly initialized weights, the inclusion of synthetic data increases performance dramatically, while the difference is much smaller when using pre-training with ImageNet.

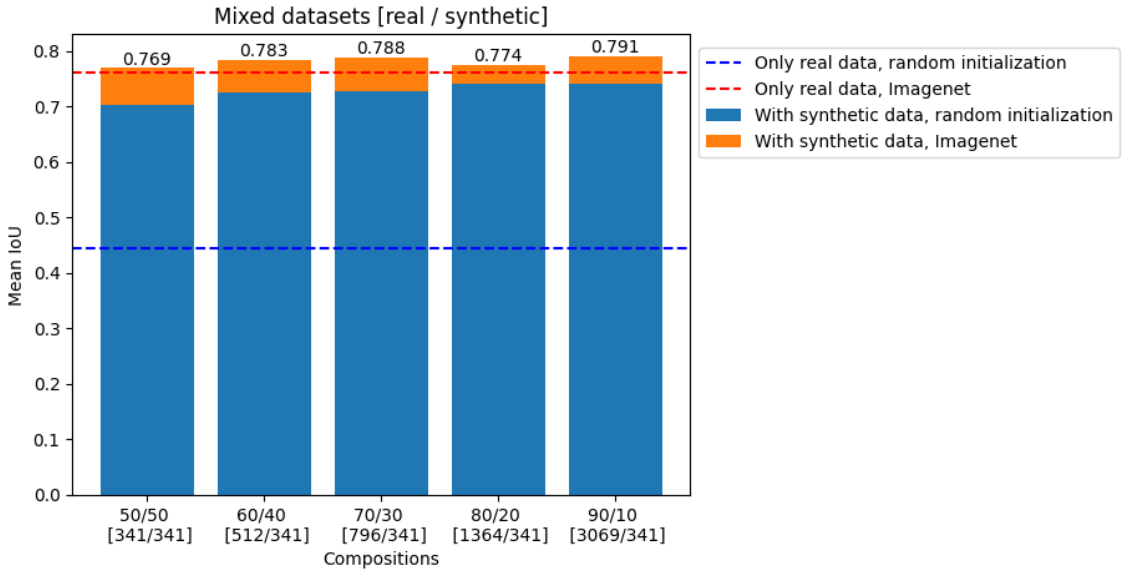


Figure 4.3: The graph shows the performance of the model when trained on different portions of synthetic and real data, both with pre-trained and randomly initialized encoder weights. The labels of the abscissa indicate both proportions and sample sizes. The synthetic data used is from generation 5. The cases when only trained on real data, with prediction accuracy 0.444 and 0.760 respectively, are used as benchmarks

4.3 Domain randomization

The domain randomization technique (DR) is implemented on generation six. See Section 3.3.2 for examples of how the DR is designed. The synthetic data is combined with the DR images and the model's performance is measured, both with and without the added images. Note that trials with and without augmentations are also explored. In Table 4.1 some results from the DR trials are provided. It can be seen that incorporating DR in this case actually reduces the model's performance.

Augmentations	With DR	Without DR
Off	0.3892	0.4722
On	0.6278	0.6848

Table 4.1: Mean IoU values for the model when DR is included in the dataset

4.4 Pre-training

In Figure 4.4 the effect of using synthetic data to pre-train the network weights has been evaluated. To avoid cluttered plots, the top three decoders with their respective top two encoders, with respect to prediction accuracy, are visualized. The results

suggest that pre-training the model on synthetic data before training it on real-world data yields a higher prediction accuracy than when only training it on real-world data.

In Figure 4.5 the impact of using encoder weights pre-trained on ImageNet, as opposed to using randomly initialized weights, is shown. The improvement is especially noteworthy for the Unet++ decoder.

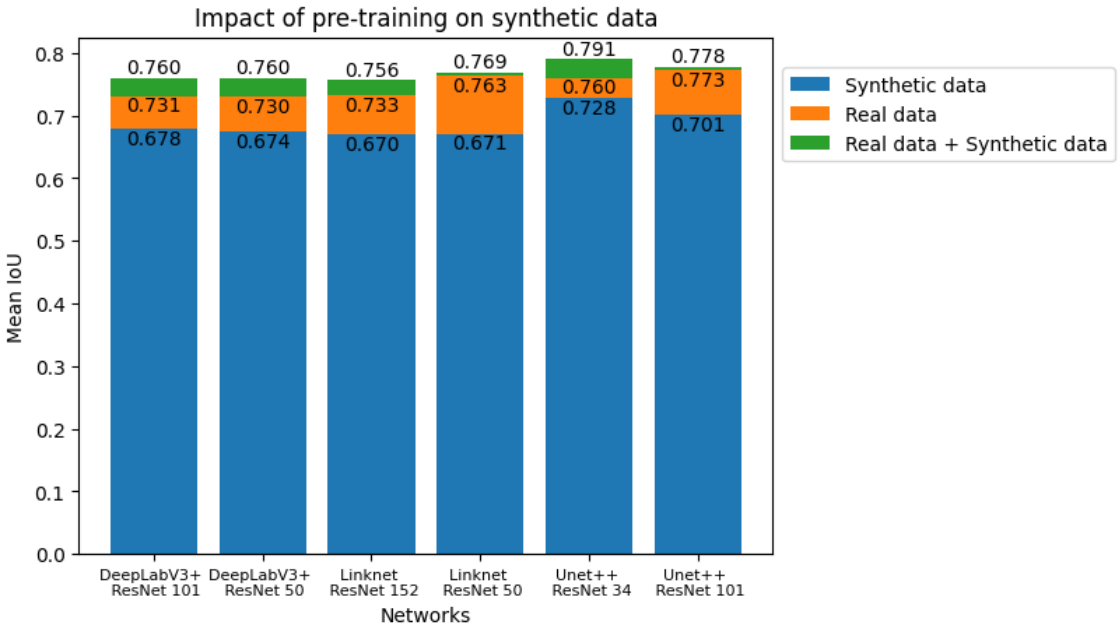


Figure 4.4: The plot visualizes the prediction accuracy of models trained on only synthetic data, only real data, and models first trained on synthetic data and then trained on real data. Initially, each model uses an encoder that is pre-trained on ImageNet

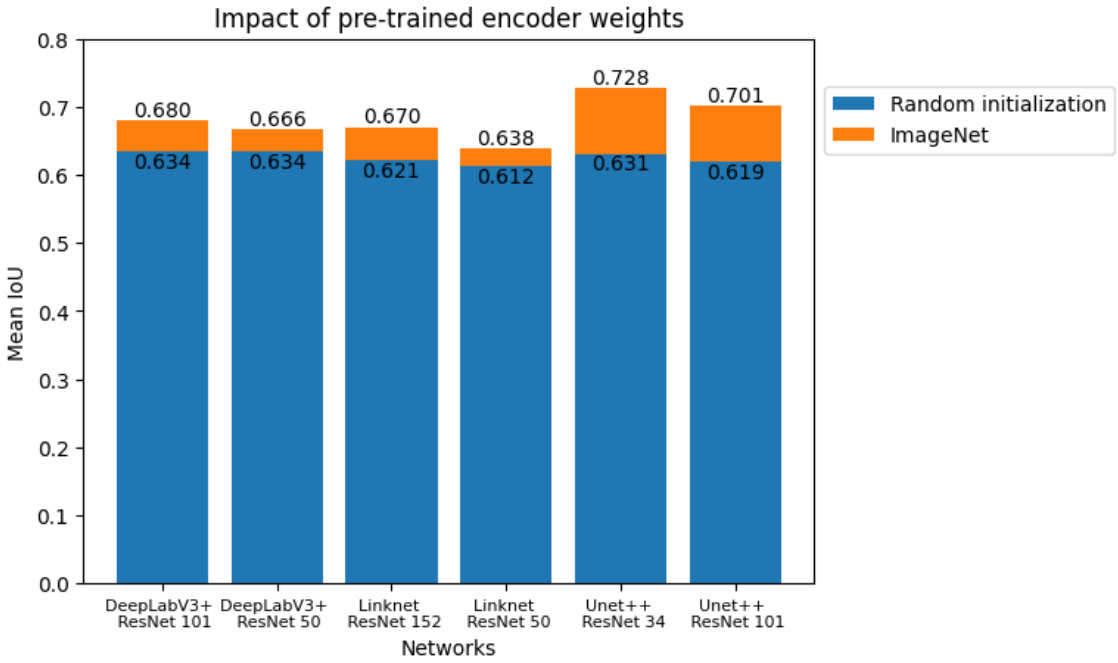


Figure 4.5: Visualization of the prediction accuracy gained from using encoders that are pre-trained on ImageNet

4.5 Inclusion of context

If the model learns to identify and segment other objects in the image, especially objects that interact with the seat belt, does its ability to identify and segment the seat belt improve? The results gathered in Figure 4.6 indicate that this could be the case. Small improvements can be observed when the model is trained on segmenting the objects which border the seat belt. The model is trained on the sixth generation of synthetic data, and tested on the real-world test set.

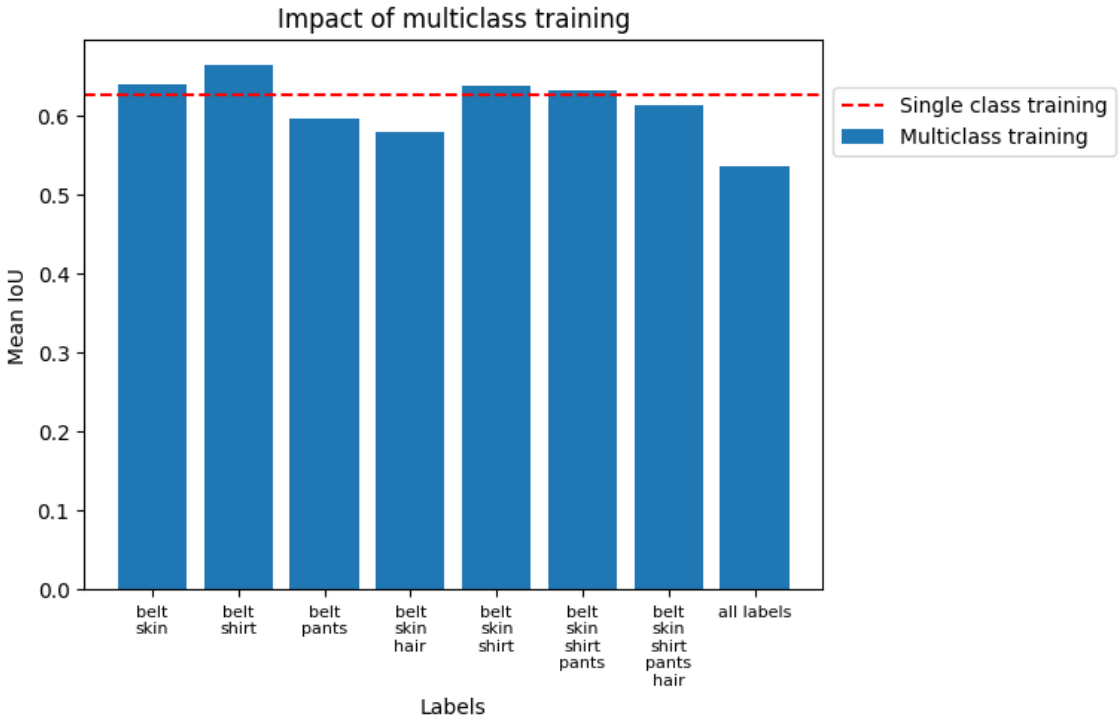
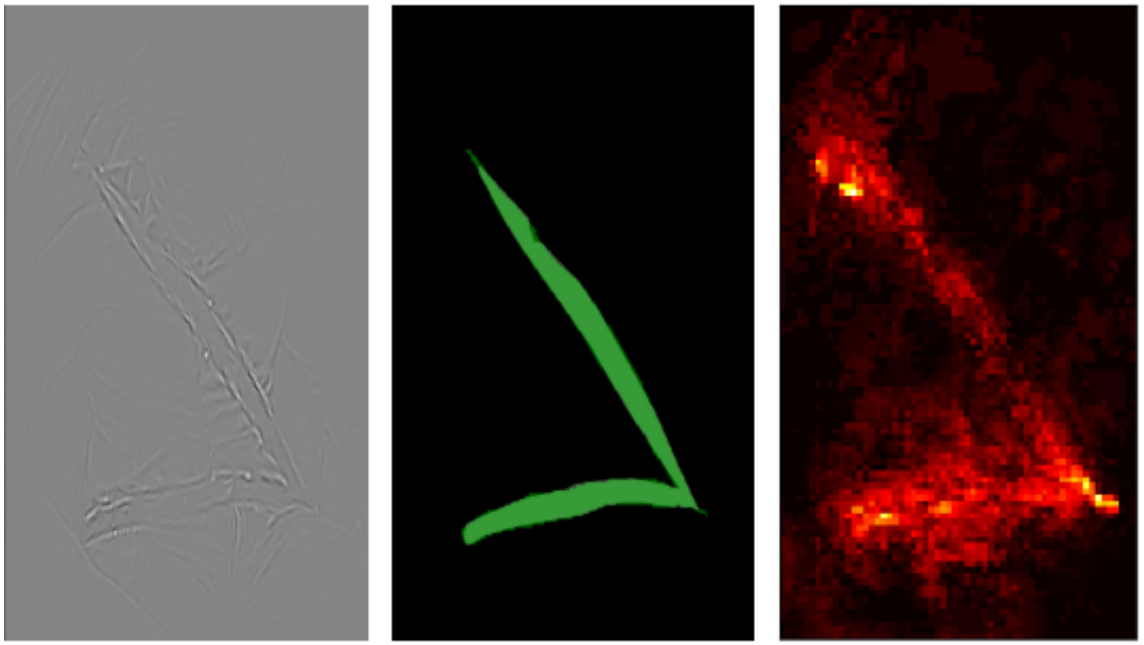


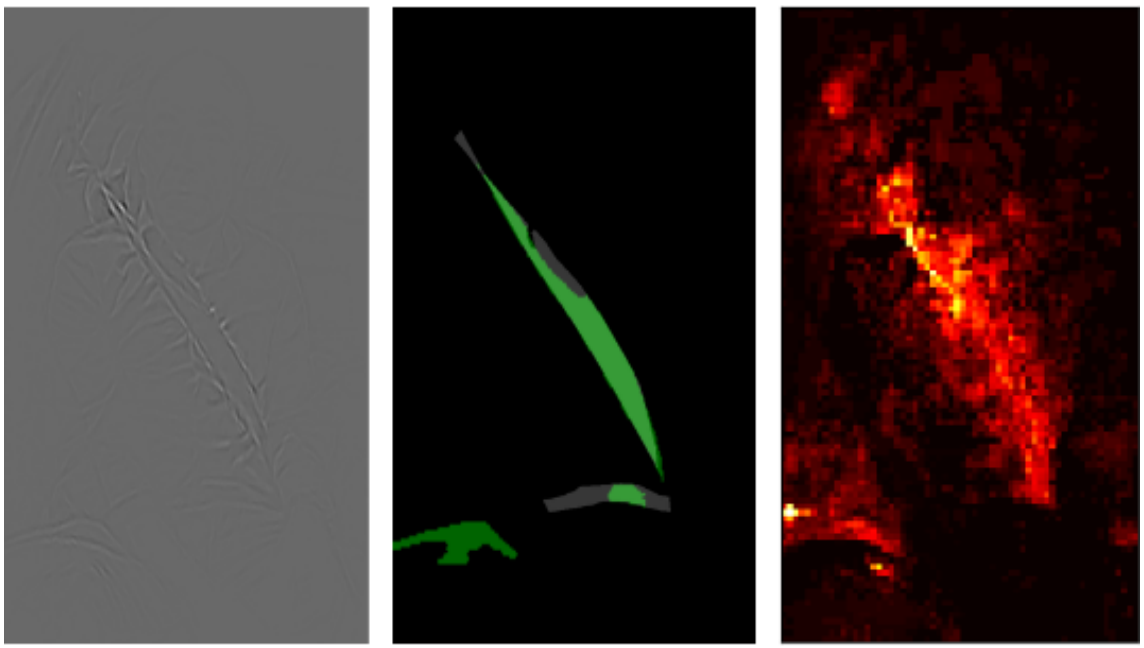
Figure 4.6: Results from the inclusion of context. Each bar shows the prediction accuracy of models trained with various combinations of classes/labels. The models are evaluated solely on the task of segmenting seat belts

4.6 Explainability of the segmentation models

The methods described in Sections 3.4.1 and 3.4.2 are implemented on a model which is trained solely on synthetic data. The results of GBP presented in Figure 4.7 indicate that the model learns to identify the important features of the real-world images from training on synthetic data. For images where the prediction accuracy is high (Figure 4.7a) the model successfully identifies all necessary features to correctly identify the seat belt. For images where the prediction accuracy is low (Figure 4.7b) the model fails to identify certain parts of the seat belt, and it's easy to see why it fails in its segmentation. The results from the simplistic saliency map method serve to highlight that the model bases its predictions on a pixel level, not just identifying the border features and filling in the gaps, i.e. removing pixels (turning them white) in the middle of the seat belt affects the prediction in that region.



(a) IoU= 0.883



(b) IoU= 0.498

Figure 4.7: The leftmost images in each triplet visualize the gradient map obtained from GBP. The images in the center visualize the predictions (in green) overlaying the annotation masks (in grey). The rightmost images visualize the results from the simplistic saliency map method. The figures visualize the best (4.7a) and worst (4.7b) segmented image in the test set. The network is trained on a synthetic dataset consisting of generations 2-6

5
Discussion

In this chapter the specific results gathered throughout the work on this thesis are discussed in more detail, highlighting the main takeaways from the plots in Chapter 4. It also addresses why the experiment regarding DR did not prove fruitful.

5.1 Changing the architecture

One of the first points of interest in this thesis was to change the architecture to find if other model compositions could yield a better result than the FPN/ResNet 101 network that was inherited from an earlier thesis work at Volvo Cars. Even though the results show that the choice of architecture has a relatively small impact on the prediction accuracy for the specific task considered in this thesis, Unet++/ResNet34 gave the best results (Figure 4.1). Note that according to [11] it is hard to evaluate whether any architecture is objectively better than others in any general sense. The result depends strongly on the task at hand, and on the type of data.

Another interesting, and somewhat counter-intuitive, takeaway from these results is that the depth of the encoder had little to no improvement over more shallow encoders. The complexity, i.e., the number of parameters, in a deep neural network does not indicate its performance. One train of thought on this matter is that, since the task of segmenting seat belts in environments with very little variation is a simple task, relatively few parameters are needed to extract all necessary features [55].

5.2 Improving the results with augmentations

Augmentations is a widely applied concept for improving results in image classification or segmentation tasks [6]. Augmentation could also be seen as synthetic data in itself since it artificially expands the data set. In this work, the goal of the augmentations was simply to introduce a larger variation within the data set, to improve the model's ability to generalize. In this thesis augmentations, albeit stochastic, were always shown to be beneficial.

Since the `albumentations` library is very large, only a few of the augmentations were applied, and the selection process was subjective. Every augmentation that was added, however, showed an increase in performance. As indicated in Figure 4.2, augmentations had a big impact on the model's performance, and given more time, it would have been interesting to investigate which augmentations, and combinations

thereof, were optimal for this use case. However, investigating each combination of augmentations would require an extreme amount of iterations. Since the probability of occurrence is set for each augmentation, there is a stochastic ingredient, demanding even more iterations to collect trustworthy results.

5.3 Evolution of the synthetic data

The result from adding variation to the synthetic data can be seen in Figure 4.2. It is notable that the model when trained on the sixth generation, which looks the most realistic, performs slightly worse than when trained on the fifth generation. This is likely due to a decrease in variation in the dataset. Since the sixth generation only has four different "persons", it likely becomes biased and decreases its ability to generalize. This is in line with mixed generations datasets resulting in the best performance, since variation is added. Another point of interest that emerged, that inspires future work, is the portion of different variations, such as black belts versus blonde belts, light car interior versus dark, or the number of images with empty seats and un-belted occupants.

5.4 Incorporating synthetic data

The main aim of the thesis was to investigate the usability of synthetic data in training a seat belt segmentation model. As indicated by the results presented in Figures 4.3 and 4.4, models, where synthetic data was incorporated in training, outperformed models trained solely on the limited amount of real data. In Figure 4.3, by comparing the results of randomly initialized weights, with and without the inclusion of synthetic data, it is suggested that the inclusion of synthetic data can accelerate convergence during training. When synthetic data is included, the model reaches a mean IoU above 0.70, while training the same network, with solely real data, only reaches a mean IoU of 0.44 in the same number of epochs. It is nearly on par with a model which uses an encoder pre-trained on ImageNet and then trained on real data.

5.5 Inclusion of context

The result from the multiple class training (Figure 4.6) was promising, and was enabled by one of the key advantages of synthetic data; the resources saved on annotations. The results indicate that training the model to identify the occupant's shirt and/or skin improves the prediction accuracy of the seat belt. This is in line with the intuition which fueled this investigation, since they are, arguably, the objects which interact the most with the seat belt. The results also suggest that adding other labels, such as pants and hair, gave a worse performance. This could stem from erroneous annotations due to graphical bugs, which has been observed in some synthetic images. For example, it has been observed that in some image where the occupant's hair is obscuring parts of the seat belt, the annotations indicate the

opposite, i.e., that the hair is under the seat belt. The experiments with the inclusion of context were performed late in the thesis process, since only the last generation of synthetic data has the multiple labeled annotations applied.

5.6 Domain randomization

Inspired by the results from [45] an attempt was made to generate domain randomization (DR) images. For DR there exist no libraries such as `albumentations`. Instead, Internal Perception was tasked with creating these images in Unity. The results from these attempts indicated that DR reduced the model's performance, contradictory to the results in [45]. There is still a belief that there is potential to the approach, though, but that the creation of the data needs more consideration. The implementation in this thesis was likely too extreme, i.e., the poor results were mainly due to the created DR images being too different to the rest of the data.

5.7 Explainability

As mentioned in Section 2.5, the idea of explaining predictions is not as intuitive for segmentation tasks as it is for classification tasks. When looking at the GBP output in Figure 4.7a from a human perspective, the features that are accentuated are enough to successfully segment most of the seat belt. In comparison to the poor prediction equivalent in Figure 4.7b, the features that are accentuated are not enough to properly segment the seat belt. This indicates that the poor prediction is a result of the model failing to identify the key features, not only incorrectly predicting the wrong features as seat belt. Overall the horizontal part of the seat belt is harder for the model to identify. This is likely because it is often obscured by hands or clothing. Including more such scenarios in the training set would likely improve the performance. This is something that could easily be done when using synthetic data.

6
Conclusions

Through a comprehensive analysis of object segmentation models with synthetic data, different techniques on how to produce and alter the data has been explored. Changing the architecture of the neural network to improve performance has also been explored. The following key findings and insights have emerged from this thesis.

The choice of network architecture had a relatively small impact on prediction accuracy. For the specific task, a Unet++ decoder coupled with a ResNet 34 encoder was found to be the overall top performer. Notably, a deeper encoder adds little to no improvement, likely due to performance saturation.

It was observed that for the specific use case considered in this thesis, synthetic data can be used to improve the prediction accuracy of the segmentation model when real data is scarce, either by pre-training the weights on the synthetic data (Figure 4.4) or by adding the synthetic data to the real-world dataset (Figure 4.3). The attempts made to make the synthetic data more realistic, i.e., narrowing the domain gap via prompts of adding variation, proved fruitful. Together with the positive results from the inclusion of context, this motivates the further development of the synthetic data application and inspires curiosity toward what could be achieved with a more powerful graphics engine.

It can also be concluded, that given the current state of the synthetic data, a model which is trained on only synthetic data can not outperform a model trained on real-world data.

6.1 Future work

This thesis also considered other approaches for generating synthetic data instead of (or in addition to) using the images created by the Internal Perception team. Inspired by the stroke edit work in guided image synthesis [56] an attempt to create new images via a diffusion model was explored. Diffusion models can be derived from non-equilibrium thermodynamics. Our initial results, described below, indicate that generating images with diffusion models may be a promising direction for future work. However, since synthetic images based on diffusion models were not used directly in the thesis, only a very brief explanation, that covers the essential steps of the model, will be provided.

The models consists of two Markov processes, a forward and a reverse process. The forward process can be described as taking a sample (which in this case would be an image) from a data distribution which is labeled $q(x^{(0)})$. Then, the data distribution is converted into a more tractable distribution, e.g., by applying Gaussian noise until it is no longer recognisable. Equation (6.1) shows the forward process for the diffusion model.

$$q(x^{(0..T)}) = q(x^{(0)}) \prod_{t=1}^{T} q(x^{(t)}|x^{(t-1)}) \tag{6.1}$$

The reverse process is where the model trains. This process is tasked with de-noising the image to once again make it coherent, i.e. it needs to reverse the Markov process to make it come back to the original data manifold (the training images). The reverse process is described in Equation (6.2). Different time steps in the forward process are associated with different noise levels and the model can learn to undo these individually [57].

$$p(x^{(0..T)}) = p(x^{(0)}) \prod_{t=1}^{T} p(x^{(t-1)}|x^{(t)}) \tag{6.2}$$

In Figure 6.1 a result of a *text-to-image* image, made by the diffusion model, can be seen. While some aspects of the image are more photorealistic than the synthetic data, there are apparent precision issues. This issue is most likely due to a lack of training data, specific to the use case, in the underlying diffusion model. To obtain more reliable and useful images like those presented in [58, 56] a more sophisticated dataset, such as the *LSUN* dataset, would be needed. For an example of a image create from a good model, see Figure 2.1.

Figure 6.1: Image depicting a car passenger created by a diffusion model

Bibliography

[1] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[2] A. J. Al-mahasneh, S. Anavatti, and M. Garratt, "The development of neural networks applications from perceptron to deep learning," 10 2017.

[3] H. Smolic, "How much data is needed for machine learning?." `https://graphite-note.com/how-much-data-is-needed-for-machine-learning`, 2022. Accessed: 2023-04-02.

[4] S. Pokhrel, "Image data labelling and annotation — everything you need to know." `https://towardsdatascience.com/image-data-labelling-and-annotation-everything-you-need-to-know-86ede6c684b1`, 2020. Accessed: 2023-04-02.

[5] A. Jain, K. Lee, G. Swaminathan, H. Yang, B. Schiele, A. Ravichandran, and O. Dabeer, "Completr: Reducing the cost of annotations for object detection in dense scenes with vision transformers," 2022.

[6] L. Taylor and G. Nitschke, "Improving deep learning using generic data augmentation," 2017.

[7] H. Mikami, K. Fukumizu, S. Murai, S. Suzuki, Y. Kikuchi, T. Suzuki, S.-i. Maeda, and K. Hayashi, "A scaling law for synthetic-to-real transfer: How much is your pre-training effective?," *arXiv preprint arXiv:2108.11018*, 2021.

[8] G. Ros, L. Sellart, J. Materzynska, D. Vazquez, and A. 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*, pp. 3234–3243, 2016.

[9] F. Poucin, A. Kraus, and M. Simon, "Boosting instance segmentation with synthetic data: A study to overcome the limits of real world data sets," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 945–953, 2021.

[10] S. Sankaranarayanan, Y. Balaji, A. Jain, S. N. Lim, and R. Chellappa, "Learning from synthetic data: Addressing domain shift for semantic segmentation," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3752–3761, 2018.

[11] S. Minaee, Y. Boykov, F. Porikli, A. Plaza, N. Kehtarnavaz, and D. Terzopoulos, "Image segmentation using deep learning: A survey," 2020.

[12] J. L. B. Diederik P. Kingma, "Adam: A method for stochastic optimization," 2017.

[13] A. Rosebrock, "Intersection over union (iou) for object detection." https://pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/, 2016. Accessed: 2023-03-28.

[14] Kukil, "Intersection over union (iou) in object detection segmentation." https://learnopencv.com/intersection-over-union-iou-in-object-detection-and-segmentation/, 2022. Accessed: 2023-05-29.

[15] E. Tiu, "Metrics to evaluate your semantic segmentation model." https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2, 2019. Accessed: 2023-05-25.

[16] , "Bcewithlogitsloss." https://pytorch.org/docs/stable/generated/torch.nn.BCEWithLogitsLoss.html. Accessed: 2023-05-25.

[17] F. Hartleitner, "Relation of objects with neural networks in three-dimensional space," *Bulletin of Mathematical Biology*, vol. 80, pp. 45-46, 2021.

[18] J. Brownlee, "How do convolutional layers work in deep learning neural networks?." https://machinelearningmastery.com/convolutional-layers-for-deep-learning-neural-networks/, 2017. Accessed: 2023-05-30.

[19] J. Teuwen and N. Moriakov, "Chapter 20 - convolutional neural networks," in *Handbook of Medical Image Computing and Computer Assisted Intervention* (S. K. Zhou, D. Rueckert, and G. Fichtinger, eds.), The Elsevier and MICCAI Society Book Series, pp. 481–501, Academic Press, 2020.

[20] "What are convolutional neural networks?." https://www.ibm.com/topics/convolutional-neural-networks, n.d.

[21] A. Kumar, "Demystifying encoder decoder architecture neural network." https://vitalflux.com/encoder-decoder-architecture-neural-network/, 2023. Accessed: 2023-05-04.

[22] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," 2016.

[23] G. L. Team, "Introduction to resnet or residual network." https://www.mygreatlearning.com/blog/resnet/, 2022. Accessed: 2023-04-06.

[24] V. Sharma, "Resnets: Why do they perform better than classic convnets? (conseptual analysis)." https://towardsdatascience.com/resnets-why-do-they-perform-better-than-classic-convnets-conceptual-analysis-6a9c82e06e53, 2021. Accessed: 2023-04-11.

[25] F. Hasainia, "What is feature pyramid network (fpn)?." https://www.fastpath.one/blog/feature-pyramid-network-fpn, 2022. Accessed: 2023-04-12.

[26] W. Yang, S. Li, W. Ouyang, H. Li, and X. Wang, "Learning feature pyramids for human pose estimation," 2017.

[27] J. Hui, "Understanding feature pyramid networks for object detection (fpn)." https://jonathan-hui.medium.com/understanding-feature-pyramid-networks-for-object-detection-fpn-45b227b9106c, 2018. Accessed: 2023-04-12.

[28] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie, "Feature pyramid networks for object detection," 2017.

[29] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," *arXiv 1505.04597*, 2015.

[30] H. Lambda, "Understanding semantic segmentation with unet." https://towardsdatascience.com/understanding-semantic-segmentation-with-unet-6be4f42d4b47, 2019. Accessed: 2023-05-05.

[31] C. O'Sullivan, "U-net explained: Understanding its image segmentation architecture." https://towardsdatascience.com/u-net-explained-understanding-its-image-segmentation-architecture-56e4842e313a, 2023. Accessed: 2023-05-05.

[32] Z. Zhou, M. M. R. Siddiquee, N. Tajbakhsh, , and J. Liang, "Unet++: A nested u-net architecture for medical image segmentation," 2018.

[33] A. Chaurasia and E. Culurciello, "Linknet: Exploiting encoder representations for efficient semantic segmentation," 2017.

[34] J. Brownlee, "A gentle introduction to batch normalization for deep neural networks." https://machinelearningmastery.com/batch-normalization-for-training-of-deep-neural-networks/, 2019. Accessed: 2023-05-31.

[35] V. Singh, "The ultimate guide to deeplabv3 – with pytorch inference." https://learnopencv.com/deeplabv3-ultimate-guide/, 2022. Accessed: 2023-05-08.

[36] Baeldung, "How to calculate receptive field size in cnn." https://www.baeldung.com/cs/cnn-receptive-field-size. Accessed: 2023-05-25.

[37] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, "Rethinking atrous convolution for semantic image segmentation," 2017.

[38] L.-C. Chen, Y. Zhu, G. Papandreou, F. Schroff, and H. Adam, "Encoder-decoder with atrous separable convolution for semantic image segmentation," 2018.

[39] J. Jordon, L. Szpruch, F. Houssiau, M. Bottarelli, G. Cherubin, C. Maple, S. N. Cohen, and A. Weller, "Synthetic data – what, why and how?," 2022.

[40] V. Petrosyan, "Why pixel precision is the future of the image annotation." https://superannotate.medium.com/why-pixel-precision-is-the-future-of-the-image-annotation-12a891367f7b, 2019. Accessed: 2023-06-05.

[41] Javatpoint, "Understanding collisions." https://www.javatpoint.com/unity-understanding-collisions. Accessed: 2023-05-31.

[42] S. Zhou, J. Zhang, H. Jiang, T. Lundh, and A. Y. Ng, "Data augmentation with mobius transformations," 2020.

[43] "Do more with less data." https://albumentations.ai/. Accessed: 2023-03-28.

[44] J. Tobin, R. Fong, A. Ray, J. Schneider, W. Zaremba, and P. Abbeel, "Domain randomization for transferring deep neural networks from simulation to the real world," 2017.

[45] J. Tremblay, A. Prakash, D. Acuna, M. Brophy, V. Jampani, C. Anil, T. To, E. Cameracci, S. Boochoon, and S. Birchfield, "Training deep networks with synthetic data: Bridging the reality gap by domain randomization," 2017.

[46] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *arXiv preprint arXiv:1312.6034*, 2013.

[47] J. T. Springenberg, A. Dosovitskiy, T. Brox, and M. Riedmiller, "Striving for simplicity: The all convolutional net," *arXiv preprint arXiv:1412.6806*, 2014.

[48] K. Wickstrøm, M. Kampffmeyer, and R. Jenssen, "Uncertainty modeling and interpretability in convolutional neural networks for polyp segmentation," in *2018 ieee 28th international workshop on machine learning for signal processing (mlsp)*, pp. 1–6, IEEE, 2018.

[49] P. Yakubovskiy, "Segmentation models pytorch," *GitHub repository*, vol. 57, 2020.

[50] P. Y. Simard, D. Steinkraus, J. C. Platt, *et al.*, "Best practices for convolutional neural networks applied to visual document analysis.," in *Icdar*, vol. 3, Edinburgh, 2003.

[51] D. J. A. Makhdoomi, "The three components of color you need to understand," Nov 2021.

[52] G. Lin, C. Shen, A. Van Den Hengel, and I. Reid, "Exploring context with deep structured models for semantic segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 40, no. 6, pp. 1352–1366, 2017.

[53] R. Mottaghi, X. Chen, X. Liu, N.-G. Cho, S.-W. Lee, S. Fidler, R. Urtasun, and A. Yuille, "The role of context for object detection and semantic segmentation in the wild," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 891–898, 2014.

[54] K. Gotkowski, C. Gonzalez, A. Bucher, and A. Mukhopadhyay, "M3d-cam: A pytorch library to generate 3d data attention maps for medical deep learning," *arXiv preprint arXiv:2007.00453*, 2020.

[55] J. C.-H. Lin, "Can a neural network be too deep?," Dec 2022.

[56] C. Meng, Y. He, Y. Song, J. Song, J. Wu, J.-Y. Zhu, and S. Ermon, "Sdedit: Guided image synthesis and editing with stochastic differential equations," 2022.

[57] J. Sohl-Dickstein, E. A. Weiss, N. Maheswaranathan, and S. Ganguli, "Deep unsupervised learning using nonequilibrium thermodynamics," 2015.

[58] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," 2022.