CHALMERS
UNIVERSITY OF TECHNOLOGY

Human Image Transfer: From Natural Settings to Controlled Studios

Master's thesis in Complex Adaptive Systems

ISAC NORDIN

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Human Image Transfer: From Natural Settings to Controlled Studios

ISAC NORDIN

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2023

Human Image Transfer: From Natural Settings to Controlled Studios
ISAC NORDIN

Supervisor: Fredrik Vedig, WeOn Company representative
Examiner: Giovanni Volpe, Department of Physics

Master's Thesis 2023
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: Transfer of a person in an image between two different backgrounds. Original images come from Viton HD dataset [1]

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2023

Human Image Transfer: From Natural Settings to Controlled Studios
ISAC NORDIN
Department of Physics
Chalmers University of Technology

Abstract

Machine learning (ML) and artificial intelligence (AI) are two concepts that have driven advancements in various fields, including the fashion industry. Virtual try-on, the technology which allows a user to change a person's clothes in an image, has gained more attention. However, their potential cannot be fully utilized when applied to photos captured in real-world environments (in-the-wild). Therefore, this project aims to investigate methods for realistically transferring in-the-wild photos to in-studio photos, focusing on domain transfer models.

To do this, four CNN-based models were used, trained to fix augmented in-studio images to their original state. The augmentations attempted to simulate in-the-wild images by swapping the background and making the image brighter and blurrier among others. To help the model discern the background of the person a segmentation extractor was used and evaluated. To get the final model, various experiments were done. The model demonstrated its ability to remove lighting, fix sharpened images, and remove noise, but failed at removing shadows among other things. The model showcased better performance at transferring in-the-wild images to in-studio images than copy-pasting the person into a studio background. The segmentation played an important role, ignoring to include body parts that were not inside the segmentation. The evaluation method showcased inconsistencies and needs further research.

Keywords: Deep learning, Domain transfer, Virtual Try on

Acknowledgements

I would like to express my gratitude to my supervisor Fredrik Vedig, and all others at WeOn for their guidance and for giving me access to their computers and resources. This thesis would not have been possible without their help. I am also thankful to my examiner Giovanni Volpe for his input during the project.

Isac Nordin, Gothenburg, June 2023

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

ANN: Artificial Neural Network
CNN: Convolutional Neural Network
FID: Fréchet Inception Distance
GAN: Generative Adversarial Network
HDR: High Dynamic Range
LPIPS: Learned Perceptual Image Patch Similarity
LR: Learning Rate
MAE: Mean Absolute Error
MEG: Multiple Encoder Generator
MSE: Mean Squared Error
PSNR: Peak Signal-to-Noise Ratio
SEG: Single Encoder Generator
SSIM: Structural Similarity Index
VGG: Visual Geometry Group

Contents

List of Figures

List of Tables

1
Introduction

1.1 Background

Machine learning (ML) and artificial intelligence (AI) which have been present for numerous years had driven advancements in various fields, from the initially small models to the larger and more complex models using deep learning. These accomplishments have transformed several industries and fields such as computer vision, and content generation, among others. One of the fields which have been undergoing rapid change due to the surge in deep learning is the fashion industry. Application of deep learning techniques has led to the possibility of changing a person's clothes in an image, or so-called virtual try-on, which enables e-commerce companies to cut photography costs and illustrate garments in novel ways [1][2][3].

The customizability of where the virtual try-on is performed is a fundamental aspect, where users are allowed to choose the specific context or background in which the try-on is performed. For instance, a user might provide an outdoor image of themselves and desire to be transferred to a different background, such as a professional photography studio. However, a simplistic approach like copying and pasting the person into the target background would lack the necessary photorealism required for commercial use. Consequently, there is a demand for high-resolution inpainting methods for transferring individuals captured in real-world environments (in-the-wild) to specific target backgrounds such as the in-studio background.

Another reason for developing high-resolution inpainting methods is that almost all state-of-the-art try-on networks are trained using professional high-resolution images taken in studios with carefully controlled backgrounds and lighting conditions [1][2][3]. However, individual users might not possess similar photography skills or access to comparable equipment, such as cameras, light sources, or backgrounds, resulting in significant differences between their own images and the training dataset, which in turn causes the generated images to have a lower quality.

Although the customers could still utilize virtual try-on by testing the clothes on a specific model in a studio setting. It is much more desirable for customers to personally try on the clothes themselves. Therefore domain transfer techniques are needed where information from one domain is transferred to another domain, where a domain can be seen as a collection of features. In this case, the domain refers to the combinations of studio background, the lighting expected from the studio,

and the camera quality expected from in-studio images. The person would then get transferred from e.g. an in-the-wild domain to an in-studio domain, containing all these features. A key point with these domain transfer techniques unlike regular ML techniques is that they effectively allow models to be applied on a dataset that is similar yet quite different in certain aspects than the dataset it was trained with.

1.2 Problem statement

The project aims to explore various aspects of deep learning, including model architecture and dataset-augmented approaches to realistically transfer in-the-wild photos to in-studio photos. Additionally, the project aims to assess the importance of feature extractors.

The limitations that existed included a GPU of 12 GB memory for training and the usage of public datasets.

1.3 Thesis outline

The report is divided into six different chapters, Introduction, Theory, Method, Results, Discussion, and Conclusion. The first chapter, Introduction, introduce and motivate the subject of domain transfer, the goals of the project, and related work. The second chapter, Theory has the purpose to bring up relevant theory to help the reader to better understand the thesis. The third chapter, Method, express the steps taken to accomplish the goals in section 1.2 and how they were argued for. The fourth chapter, Results, presents the different results of evaluating trained models. The fifth chapter, Discussion, evaluates the method process and results to relate how well the project has succeeded according to the goals in section 1.2. The sixth and last chapter, Conclusion summarizes the results and the discussion of how the goals were accomplished.

1.4 Related work

In recent years there have been several different projects demonstrating the potential of domain Transfer models. These include the generation of shadows for composite images where an object has been placed in a background image, the removal of shadows from images, and converting images into 3d models among others. [4][5][6][7]. Some of the more specific researched areas relevant to this project are Relighting and Style transfer.

1.4.1 Relighting

An image's lighting condition is quite important to present an impression on the person watching it, where e.g. a very dark image may elicit a sense of creepiness,

while a bright image can appear as colorful and lively. However, a simple adjustment to brightness does not suffice, since the dark and bright parts would still have a big contrast. In this regard, many different techniques are presented for changing the lighting in an image. Some have researched the usage of reference images to relight an image [8][9]. Additionally, researchers have also tackled the challenge of recovering details from images that have lost information due to being either too dark or too bright. One approach involves transforming these images into high dynamic range (HDR) representations, which can capture a broader range of light intensity compared to conventional images [10]. Another approach was to combine images with a lighting reference and feature extractors to make certain areas brighter and restore details according to human perception [11][12]. Moreover the concept of "image harmonization" have been explored where e.g. an image of a human is moved into a random background to create a composite image of two different lighting styles. The lighting on the human is then changed to match the background lighting, creating a harmonized image [13][14].

In the various papers, the architecture structure varies but in general, most papers use an encoder to extract useful information, and a decoder to use the information to reconstruct the image. This is used as a base where some intricacies and complex layers are added to the model in different ways.

1.4.2 Style transfer

The core of Style transfer is to take an image and keep the content and global structures but change local structures to be more similar to a specific target style. There are many variations in style transfer but there have been three important milestones. The first being from Gatys et al which showcased the possibility of turning photos into a painting made by e.g. Van Gogh [15]. Some other notable papers involve the introduction to CycleGAN which allowed for style transfer without paired training data and the AdaIN technique which allows for precise control over style [16][17]. Certain papers have explored the transfer of images depicting environments, such as buildings from daylight to nighttime, simulating building windows to shining light. Similarly, efforts to transform an image depicting a spring day to look like a winter day have been made [18][19]. Note that style transfer and relighting are closely intertwined, with some relighting papers utilizing style transfer techniques to achieve their goal [18]. The combined usage of both fields has expanded upon the possibilities for image manipulation and scene transformation.

2
Theory

2.1 Machine learning

Artificial Intelligence (AI) is a broad concept used to encompass systems that are capable of tasks that are normally thought of as requiring human intelligence. A field of this is Machine learning, which shares the same requirements as AI, but additionally requires the developed system to be able to learn without explicit coding.

2.2 Deep learning

Deep learning shares the same requirements as machine learning. However, it also requires that the algorithm structure attempts to replicate certain aspects of the brain's structure, often called Artificial Neural Networks (ANN).

An artificial neural network at its core attempts to simulate the brain's structure, consisting of many interconnected nodes, with the simplest form being nodes organized into layers from left to right. Each node is only interconnected to nodes in nearby layers, as depicted in figure 2.1. The connections between nodes possess weights, and when calculating the value of a target node, such as a node in the hidden layer, the values of the nodes in the input layer are multiplied by the corresponding weights connected to the target node. These weighted values are then summed up to get the value for the target node.

The training of this neural network is called a feed-forward network since the nodes' values are calculated one layer at a time until the output has been given. The output can then be compared to some target output the model was supposed to generate. Using a loss function that does the comparison, backpropagation is used to update the weights for the connections between nodes, causing the ANN to learn.

2.2.1 Backpropagation

Backpropagation is the process of propagating the changes to the node connection weights in an ANN. It is essentially about optimizing a function output, via gradient descent. Gradient descent is the mathematical concept where a point follows a mathematical surface to reach the closest minimum point.

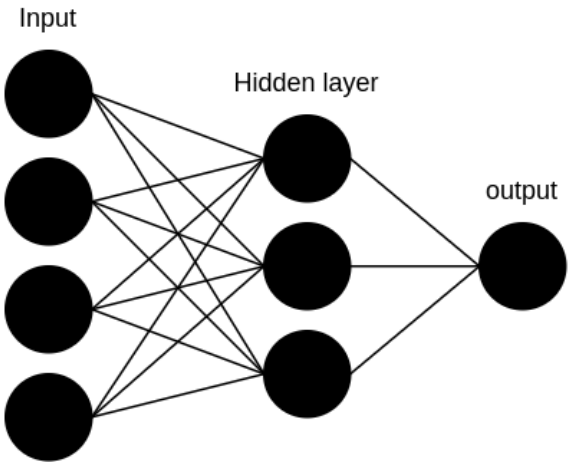

Figure 2.1: Structure of very simplistic ANN, showing different layers, and how the connections between nodes are.

Calculation of the gradient gives an indication of the direction and magnitude of the steepest descent which is used to update the weights. Since the ANN has a feed-forward structure, the loss is calculated and propagated layer by layer starting from the output layer. To control the learning process there are two methods. Firstly, the loss function quantifies the differences between generated and target data, with each loss functions capturing different aspects. Secondly, the Learning rate (LR), which is multiplied by the loss function can be modified to change how fast the model learns.

2.2.2 Loss function

A wide range of loss functions are employed to assist the learning for various problems. Simple pixel-wise comparisons and more complex comparisons involving correlation between pixels are commonly used for image losses. Perceptual loss is another type of image loss where generated & target data is fed into a pre-trained model. The output from one of the multiple model layers is chosen and used to compare the two using one of the previously mentioned losses. This approach is useful as the pre-trained model captures high-level features, enabling more perceptually accurate comparisons. Note that a model's layers learn different things and the chosen layer might not yield good results chosen randomly.

2.2.3 Convolutional Neural Network

In contrast to the simplest ANN where each layer is represented as a list of nodes with weights between nodes, a Convolution Neural Network (CNN) employs a different approach. For the CNN each layer has nodes structured as an image. The transition to the next layer involves a convolutional layer, which performs element-wise multiplication between two matrices known as a convolution. The convolutional layer performs convolutions for all image channels and sums the results elementwise. Figure 2.2 showcases this process for a single trainable filter, where the red and yel-

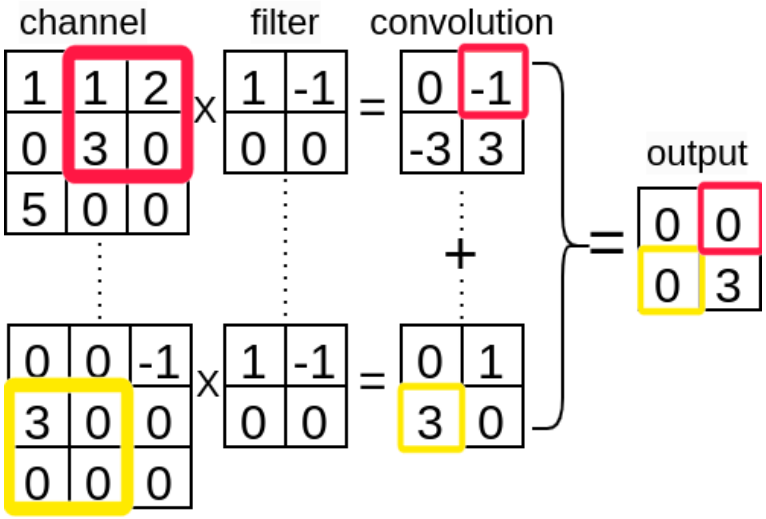

Figure 2.2: The process of a convolution layer given just one trainable filter. The red and yellow squared highlight the stepwise results.

low squares indicate certain values throughout the process.

Instead of updating the weights of connections like the ANN, a CNN updates the values of the trainable filters. The filters in the first layer learn simple features such as edges, and corners in an image, while subsequent layers learn more complex features. By adding padding to the image and using different shapes of learning filters the convolutional layer is able to decide the "image" width and height at each step, from downscaling to upscaling, however in the case of upscaling an image it is usually referred to as Transposed convolution layer.

The advantage of using a CNN instead of a simpler ANN where the image is squished to a list has to do with correlation. By squishing an image into a list, it is hard to find the correlation between what was once nearby pixels, and different channels.

2.2.4 Pooling layer

Pooling layers are layers where information is compressed. In exchange for slightly reduced information, the image size is decreased. This helps reduce model parameters and training time.

2.2.5 Activation function

An activation function is a crucial layer that takes all of the layers' nodes and applies an elementwise transformation to it. Activation layers serve two primary purposes. Firstly, the usage of a nonlinear transformation allows the model to better learn complex and nonlinear features. Secondly, activation functions are used to restrict the output growth which could otherwise make the output grow exponentially layer by layer.

2.2.6 Encoders

An encoder in its essence is the process of "condensing" the information of its input, by extracting relevant features. The implementation of an encoder in practice depends on the type of ANN being used. In regard to images, encoders typically employ convolutional layers to downscale the image size and increase the number of channels.

If the compression is excessively strong resulting in a significant reduction in image size and channels, the condensed information may fail to reflect the original input. Note that the condensed information is not necessarily a low-res version of the input. The encoder instead captures significant features that characterize the image, although these features might not be understandable by a human.

2.2.7 Decoders

Decoders are structured like encoders but in reverse order. While encoders' aim is to condense information, decoders aim to use condensed information as building blocks to construct something, often closely resembling the input. By using transposed convolutional, decoders effectively reverse the downsampling.

2.2.8 GAN

A Generative Adversarial Network (GAN) is a concept of having two interconnected networks, a generator, and a discriminator. These networks compete with each other to improve their own performances. As can be seen in figure 2.3 The generator takes random noise as input and aims to generate an image, similar to a decoder. The discriminator takes an image and aims to decipher if the image is true and is similar to an encoder but where the final step compresses the information into one value, zero, or one. As seen in figure 2.3 the discriminator loss is dependent on its ability to discern real from generated images. The generator loss instead depends solely on if the discriminator is fooled by the generated images. The goal of the GAN is to generate information that is indistinguishable from the training data.

During the training process, the generator's loss typically increases while the discriminator's loss decreases, and vice versa. The training thus leads to cycles where each model takes turns to outperform the other.

The negatives with GANs is that they can be unstable and hard to converge to an optimal solution and needs careful tuning of hyperparameters to address the issue. The positive thing is that a GAN can produce amazing results if trained stable and can be trained without any target image.

2.2.9 Learning rate scheduler

Training models with a constant learning rate throughout the training yields satisfactory results. However, it is often more beneficial to adjust the learning rate

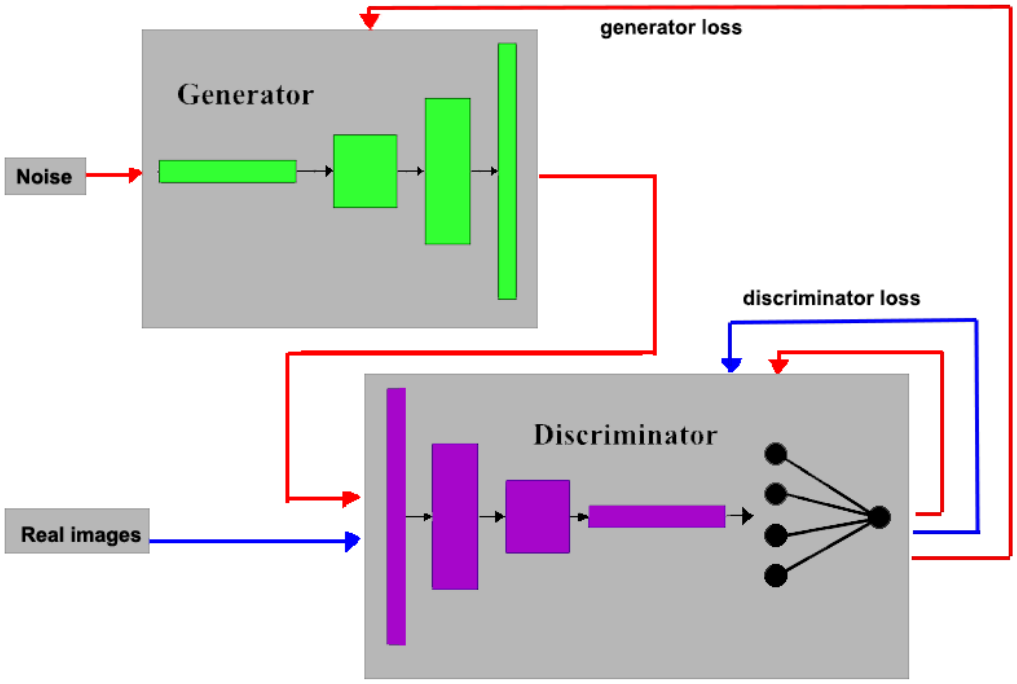

Figure 2.3: A GANs' structure and how it is trained. Red arrows come from generated images and blue arrows from real images.

during the training process.

A higher learning rate enables models to explore a broader range of options and escape from local minima. This can lead to faster convergence faster in the initial training epochs. On the other hand, using a lower learning rate instead allows for finer adjustments of model parameters. It helps the model to converge to more optimal solutions in later training epochs. A scheduler is thus used, to control how the learning rate changes commonly. Most schedulers either strictly decrease the learning rate or change in a cycle.

2.2.10 VGG19

Visual Geometry Group (VGG) is a well-known group of deep CNN models commonly used for predicting what is inside an image. Among the group, VGG19 is the model which has 19 convolutional layers.

The VGG models have been trained on the famous dataset ImageNet-1k consisting of a broad range of labeled images. These models are frequently used as benchmarks and used for transfer learning purposes. Transfer learning helps overcome the limitations of small datasets by using a VGG model that has learned many valuable features, which may be challenging to learn with a small dataset.

2.3 Dataset

2.3.1 Data augmentation

Data augmentation is commonly used in ML projects, primarily for the purpose of diversifying and artificially expanding the training data. With limited training data, expansion of the dataset is very important as the model may suffer from overfitting or insufficient training for satisfactory results. Overfitting occurs when a model learns the training data too well and fails to generalize to new unseen input data. The augmentation of data by e.g. adding shadows or making images brighter also diversifies the data allowing models to generalize better, and handle unseen data.

2.3.2 Flipping

Flipping an image, whether horizontally or vertically, might not seem like a significant augmentation from a human perspective. For a machine or ML model, the impact is significant which expands and diversifies the dataset.

2.3.3 Rotation

Rotating an image is another example that might not seem significant at first glance. However, it plays a crucial role in aligning training and testing data. For instance, if the training data consists of humans standing straight up, and the testing data consists of humans in various poses, rotating the straight images may better align the images. This ensures the model learns features that are more representative of unseen data.

2.3.4 Morphological operations

Morphological operations are image processing techniques commonly applied to black and white images, although they can also be applied to grey-scaled images. Two essential morphological operations are Dilation and Erosion.

2.3.4.1 Erosion

Erosion is an operation involving a structural element (matrix) containing ones and zeros. The center of the matrix is moved across the entire image and element-wise multiplies with the image. If the resulting sum equals the number of ones in the structural element, then the pixel at the center of the structural element becomes one, otherwise, it becomes zero. Figure 2.4 illustrates the effects of erosion alongside dilation, where black represents a value of zero, and white represents a value of one. Erosion is used to shrink the white parts while dilation expands the white parts in a black-and-white image, often referred to as a mask. The operations can thus be used to reduce noise of a specific color, and are closely related to each other. To go from e.g. erosion to dilation or vice versa one could invert the image apply the operation and invert the image once more.

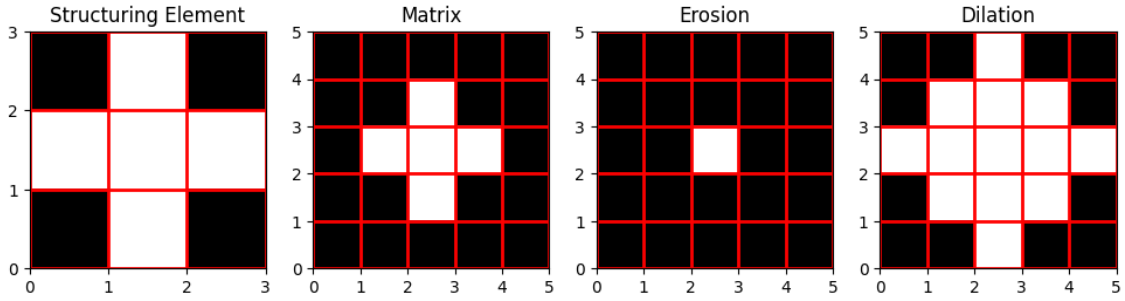

Figure 2.4: Effects of dilation and erosion given a structure element. Black represents zero, and white represents one.

2.3.5 Advanced augmentations

There are several advanced augmentations that can be useful given the models and problems addressed. In the context of this thesis, some relevant augmentations include adding shadows to a person in an image, changing the contrast and brightness, simulating a sun glare, sharpening, and defocusing the image, and making it look blurry.

2.3.6 Albumentations

"Albumentation" is a Python library providing a broad range of image augmentations, including previously mentioned augmentations (excluding the morphological operations) [25]. The augmentations provided can be particularly valuable when working with different variations of photographs as they can account for variations in camera qualities.

2.4 Data Evaluation

For the case of ANNs aimed at label prediction, the evaluation is straightforward as the accuracy of label predictions can be measured. However, for generated images, no simple reference point like a label exists to compare. Therefore, various quantitative metrics have been developed to compare and evaluate the similarity between images. Some of the metrics that are common in image generation papers are the PSNR, SSIM, LPIPS, and FID scores.

2.4.1 PSNR

The first score used to asses image quality is the Peak signal-to-noise ratio (PSNR). The PSNR score is commonly used to compare the quality of reconstructed images after compression, which tends to look more pixelated. The PSNR score can be defined as in equation 2.1 where MAX is the maximal pixel value of the image, and Mean Square Error (MSE). Note that The MSE is also called the L2-loss, and if each term wasn't squared then it would be the Mean Absolute Error (MAE), also known as L1-loss.

$$\text{PSNR} = 10 \log_{10} \left(\frac{\text{MAX}^2}{\text{MSE}} \right) \tag{2.1}$$

While the PSNR score is commonly used to evaluate image quality, it has some flaws. In comparison to some other metrics, the PSNR score doesn't align with human perception of how well-generated images look. The higher the PSNR score is, the better the evaluated image is.

2.4.2 SSIM

The Structural Similarity Index (SSIM) is a metric that evaluates the quality of an image by comparing luminance, contrast, and structural information. In comparison to the PSNR score which uses The MSE and treats each pixel separately, the SSIM score cares about the correlation between pixels and tends to align well with human perception. The score goes from negative one to positive one, where the higher the score, the better.

2.4.3 LPIPS

The Learned Perceptual Image Patch Similarity (LPIPS) is a metric that evaluates the perceptual similarity between two images. Unlike more traditional metrics like PSNR, LPIPS employs an ANN for its evaluation, where both generated and target images are passed through a pre-trained model. The output from one of the model's layers is chosen to be compared using a non-perceptual score like PSNR. Perceptual scores utilize the pre-trained models' learned high-level features to better evaluate images. The LPIPS score has been deemed to correspond to human perception, and a lower score means better image quality. Figure 2.5 showcases how the PSNR, alongside the SSIM, LPIPS score is affected by comparing different images to a target image. A1 and A2 are augmented versions of the target, B is another in-studio image with another person, and C is an in-the-wild image.

2.4.4 FID

The last metric to mention is the Fréchet Inception Distance (FID) score which is a metric commonly used to evaluate generative images created by a GAN. Unlike the other metrics, the FID compares two datasets of images. Each image is passed through a pre-trained model such as the Inception-v3 to extract features from a specific layer. The FID score is then calculated mathematically as in equation 2.2, where x and y represent real and generated data.

$$\text{FID} = \|\vec{\mu}_x - \vec{\mu}_y\|_2^2 + \text{Tr}(\Sigma_x + \Sigma_y - 2(\Sigma_x \Sigma_y)^{1/2}) \tag{2.2}$$

The equation states that the FID score compares the mean $(\vec{\mu}_x, \vec{\mu}_y)$ and covariances (Σ_x, Σ_y) of the features extracted for two datasets, one with real images and one with generated images. For reliable results, it is recommended to have a minimum of 5000 images for the dataset. It is also recommended to use a similar amount of

	target	A1	A2	B	C
PSNR:	inf	23.89	20.49	7.50	4.53
SSIM :	1.00	0.87	0.68	0.53	0.16
LPIPS:	0.00	0.27	0.44	0.47	0.71

Figure 2.5: How LPIPS, SSIM, and PSNR score is affected by comparing different images to the target image. A1, and A2 are augmented versions of the target, B is another in-studio image with another person, and C is an in-the-wild image. The original images come from Viton HD dataset, and Fashion550K [1][21].

images for both datasets as the FID score is sensitive to the number of images used. The metric aligns well with human perception and the lower the score the better the result.

3
Methods

This section describes the process, thinking, and experiments that were done to answer the research questions in section 1.2

3.1 Dataset

To achieve the goal of transferring a person from an in-the-wild background to an in-studio background, it would have been ideal to have a dataset that contained images of people in both backgrounds for supervised training. However, creating or finding such a dataset is not feasible due to the difficulty in replicating micro-positions such as body posture or hair position when taking a photo in two backgrounds. Therefore only one dataset could be used for supervised training or usage of both datasets for unsupervised training. This project used an in-studio dataset with supervised training as it allowed for more precise control and guidance in the training. To make sure the models translated in-the-wild images to in-studio images these in-studio images were first augmented to resemble in-the-wild images before being passed into the model.

There were two public datasets that included over ten thousand model images, the first being the Viton HD, or in short Viton dataset, and the Deepfashion dataset [20][1]. Although the Deepfashion dataset had roughly five times as many images, the Viton dataset was ultimately chosen to train the models. This was due to the Viton dataset being much more consistent in various aspects like the human poses, background color, and always using images containing the entire person. The file names and folders were also much easier to handle for the Viton dataset.

For validation purposes, two datasets were instead used. The Viton dataset was used to evaluate the transfer of an augmented in-studio image into an un-augmented in-studio image. On the other hand, the Fashion550k (F550k) dataset, which contains approximately "550" thousand images of random people in random backgrounds environments was used to evaluate the transfer of in-the-wild images to in-studio images [21].

To help guide the model to separate the background from the person, a segmentation extractor from the GitHub repository "Self-Correction-Human-Parsing" was used [22]. The repository proved effective in nearly achieving perfect segmentations of in-studio images and performed relatively well for most in-the-wild images. Note

that there are recent advancements in image matting, which provides a higher quality segmentation that contains opacity information (e.g. distinguish transparent glasses) [23]. For this study, a segmentation extractor was instead employed as it provided higher stability, which is especially important for in-the-wild images.

3.1.1 Data augmentations

To train the different models using the in-studio dataset as both input and target, it was necessary to introduce augmentations to the image before passing it into the models. These augmentations were focused on making the input image resemble an in-the-wild image, which would also allow it to perform better when introduced to actual in-the-wild images.

The first augmentation was done by using the segmentation extractor mentioned in section 3.1 to locate the human as well as the background. The background was then swapped with a random background from the BG-20K dataset which contains over ten thousand background images.

To clarify the differences between in-the-wild images and in-studio images there are many differences such as the person's pose, where the lighting is directed, its intensity, color, camera focus, and quality. Some in-the-wild images contain sun glares and might also have lost details due to the image being too bright while other in-the-wild images have weird shadows created from light interacting with nearby objects. For in-the-wild images, all of these factors are quite varying, while for in-studio images all images are consistent. To view images of both in-studio and in-the-wild images used, please refer to figure A.5 and figure A.6

Due to these differences, further augmentation of the input images to resemble in-the-wild images was needed. Simpler augmentations such as rotation, as well as horizontal and vertical flipping were used. These augmentations tried to align the pose better with in-studio images by rotating the person. More advanced augmentations coming from the Python library "Albumentation"[25] was also chosen to be used as the library had a broad range of augmentations, and was simple to use and modify.

The process of choosing which specific advanced augmentations to be utilized was quite subjective. Many advanced augmentations were tested and subjectively picked if they simulated in-the-wild images. These augmentations included ISONoise, Defocus, Downscale, Sharpen, RandomShadow, RandomSunFlare, and RandomBrightnessContrast, and can be seen alongside the background swapping augmentation in figure 3.1[25].

The ISONoise, Defocus, Downscale, and Sharpen were augmentations that attempted to change the image quality in hopes of replicating the impact of poorly captured in-the-wild images. The RandomShadow and RandomSunFlare augmentations added

geometric shapes with transparency to mimic real sun glares and shadows, although the resulting images may not appear realistic. These two augmentations aimed to eliminate darker areas (shadows) and to recover details from images that have lost details from being too bright. Lastly, the RandomBrightnessContrast was used to simulate the varying degrees of light that would take place in in-the-wild images.

During the training process, the augmentation parameters were evaluated by subjectively looking at some in-studio images, the augmented version, and the generated results. If the augmentations had too much impact they were adjusted to be less effective to better simulate realistic in-the-wild images. Augmentations that hindered learning despite adjustments, such as RandomSunFlare and RandomShadow were removed. The final parameters for the augmentations can be seen in table 3.1, where the written parameters represent deviations from the standard ones.

Table 3.1: Augmentation information for training. The parameters written only represent deviations from the standard parameters

Augmentation	**Probability**	**Parameters**
Flip vertically	0.25	
Flip horizontally	0.5	
ISONoise	0.5	
RandomBrightnessContrast	0.5	brightness & contrast limit=0.2
Downscale	0.25	scale=[0.75,0.8], interpolation=Nearest
Defocus	0.25	radius=(1,2)
Sharpen	0.25	alpha=(0.3, 0.3)

3.2 Architecture

There are various forms of model architectures available for image-to-image tasks, with some using Vision transformers, diffusion-based models, or simple models like regular CNNs. In this project, a CNN-based model was chosen for various reasons. Although a Vision transformer with its attention heads and a diffusion-based model have both shown great results they tend to be larger than CNNs. To ensure limitations in section 1.2 were upheld these models were not chosen to be used. CNNs were also chosen due to their simplicity and flexibility in design, making them suitable for customization.

3.2.1 MEG

The first model, referred to as Multiple Encoders Generator (MEG) was used for the project and can be seen in figure 3.2. As seen, it uses two encoders, one for the input image, and one for the segmentation mask. The encoded outputs are concatenated and passed to a shared decoder which attempts at generating an in-studio image with the person. To help the learning skip connections were used between the input image encoder and the shared decoder. The model architecture was used

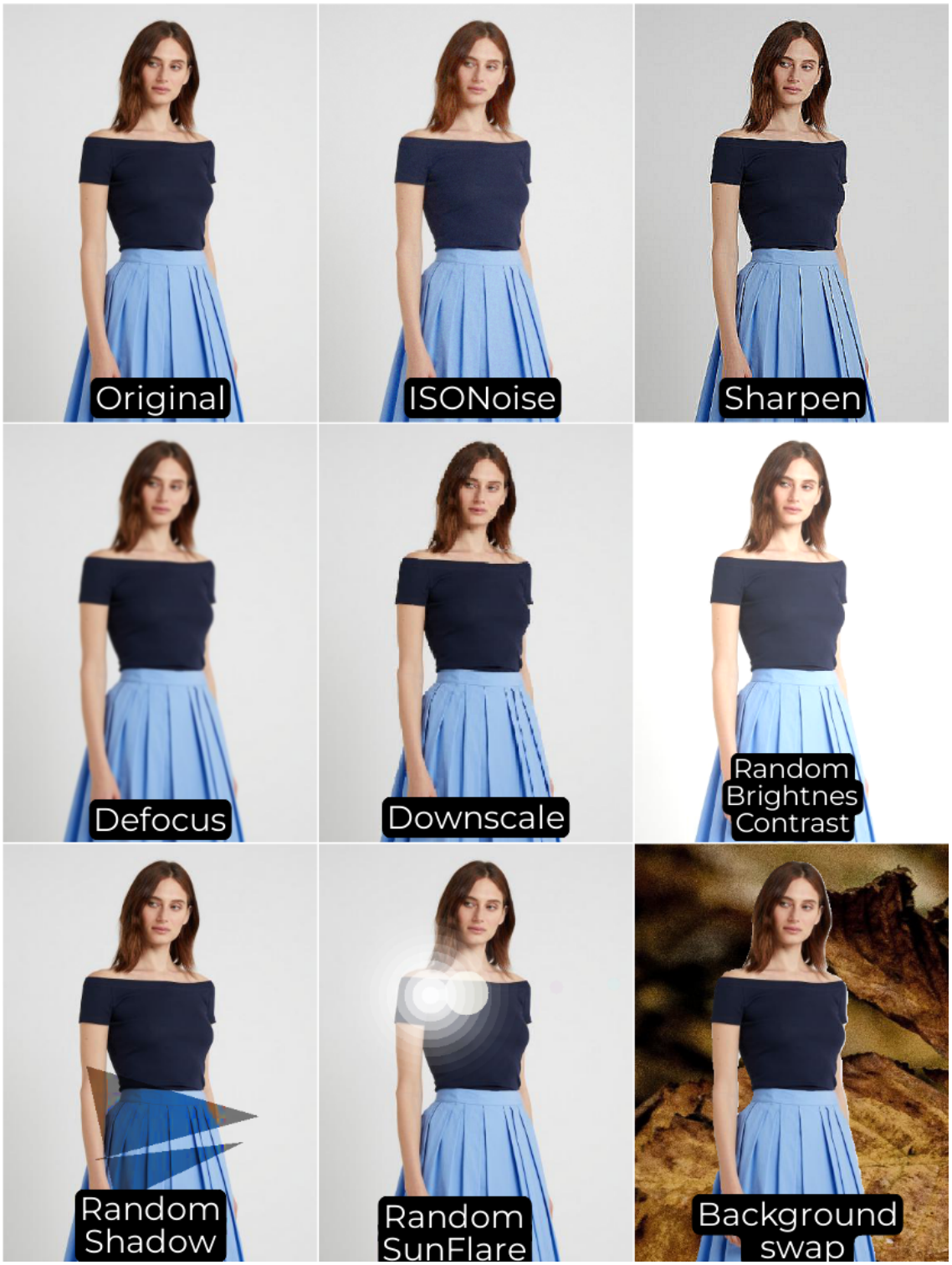

Figure 3.1: Showcase of advanced augmentations used to resemble in-the-wild images. The original images come from Viton HD dataset [1].

as similar architectures have been used in many research papers, some related to domain transfer.

In figure 3.2, each layer represented a sequence of layers including convolutional layers with parameters Conv2d($in_{channels}$ =in,$out_{channels}$ =out kernel=3x3, stride=1, padding=1) or in short Conv2d(in, out, 3, 1, 1), Relu-activation layers, normalization layers, and a max-pooling layer with kernel=2x2. The layers seen in the list below show the sequence of layers for each encoder. To obtain the decoder, the seventh layer is removed and the first layer is changed to Conv2dTranspose(X·in, out,2,2,0) where X is two if there was a skip connection, and otherwise one.

- 1) Conv2d(in, out, 3, 1, 1)
- 2) BatchNorm
- 3) ReLu
- 4) Conv2d(out, out, 2, 2, 0)
- 5) BatchNorm
- 6) ReLu
- 7) maxpool2d(kernel=2)

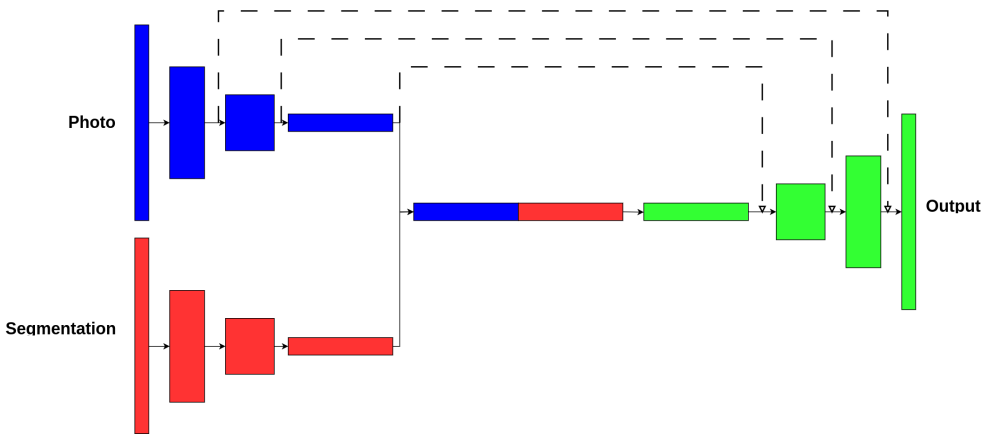

Figure 3.2: Image showcasing the structure of the MEG model, containing two encoders, one decoder, and skip connections represented with dashed lines.

3.2.2 SEG

The second model used was the Single Encoders Generator (SEG). The SEG uses the same encoder and decoder structure as the MEG however only uses one encoder. Instead, the encoder takes the input image concatenated with the segmentation as input.

Its usage was motivated as convolutional layers excel in processing the correlation between concatenated images such as the input image and segmentation. The model also used one less encoder which allowed for a bigger model which could learn more complex features to be used.

3.2.3 SEG-GAN

The third model that was tested is called SEG-GAN. It was inspired by GANs, containing the SEG acting as the generator. The discriminator was structured the same as the encoder part of the SEG, but where the output is fed into a classifier model. The Classifier downscales the number of channels and image size to (64x1) via Conv2d(X, 64, (image size), 1, 0), where X represents the encoders' output channels. The output is then flattened, passed into a linear layer, a sigmoid function, and rounded to the nearest integer for predicting either zero or one for binary classification.

The model was used as GANs' have commonly been used in generative models and have shown great results. It also allowed the model to be trained using both in-the-wild and in-studio datasets with the GAN's unsupervised learning.

3.2.4 VGG-SEG

The fourth tested model was the VGG-SEG. It uses the same decoder part as the SEG, but the encoder has been replaced with some layers from a pre-trained VGG19 model. The pre-trained VGG19 model comes from the torch-vision library, and the encoder is structured as follows, according to Python code:

- Conv2d(4,64,3,1,1) & vgg19.features[1-5]
- vgg19.features[5-10]
- vgg19.features[10-19]
- vgg19.features[19-28]

Each new encoder layer was followed by a skip connection to the decoder, and the pre-trained layers were frozen and not further trained during the project. The model also uses Conv2d(4,64,3,1,1) instead of vgg19.features[0] as the model expects four channels instead of three. The motivation for testing the pre-trained model was the potentially useful and complex features the model had learned from a much broader dataset.

3.3 Training parameters

Many parameters and loss functions were tested, and unless specifically mentioned, these were the parameters used.

All models except VGG-SEG which were unchangeable used four layers with $out_{channels}$ with $out_{channels}$ =(64,128,256,512) for the encoder. The decoder used the same amount of layers and used the same $out_{channels}$ but in reversed order. To see a schematic of all layers for a trained MEG go to figure A.4. The models were trained for eight epochs, with a constant learning rate of 0,001, and were trained with a modified SSIM loss function (see section 3.5.1 for details of modified SSIM loss).

As the SEG-GAN introduced additional losses it will be described some more. The generator was trained with both SSIM loss in the same way as the other models.

Both discriminator and generator were however trained with Binary Cross Entropy (BCE) as the GAN loss and can be seen in equation 3.1. The adjustable parameter C was used to balance the SSIM and GAN loss, while the upper score text indicated the usage of real or generated images and the dataset the images originated from. The model used $C = 1/1000$ and was only trained with The Viton dataset as an experiment including the F550k dataset (blue text) yielded worse results.

$$\begin{aligned} \lambda_{Generator} &= \lambda_{SSIM} + C \cdot \lambda_{BCE}^{fake,Viton} + \frac{C}{3} \cdot \lambda_{BCE}^{fake,F550k} \\ \lambda_{Discriminator} &= C \cdot \lambda_{BCE}^{real,Viton} + C \cdot \lambda_{BCE}^{fake,Viton} + \frac{C}{3} \cdot \lambda_{BCE}^{fake,F550k} \end{aligned} \tag{3.1}$$

3.4 Evaluation method

The evaluation of the models involved the SSIM, PSNR, LPIPS, and FID metrics, all computed with the Torchmetrics library. The FID score was calculated by comparing 5000 in-studio images with translated in-the-wild images. Commonly the SSIM, PSNR, and LPIPS scores are used when comparing an image to its augmented counterpart. Instead, in this project a translated in-the-wild image was compared with an in-studio image and created a mean score from 500 comparisons. 500 in-studio images were then augmented, translated, and compared to 500 other actual in-studio images. This was done as no target images existed for the in-the-wild dataset and using different evaluation approaches would cause the scores to be incomparable.

To reduce randomness and make the results reproducible, the same images were always used. The augmented images were also saved so all evaluations used the same augmentations.

3.5 Experiments

A multitude of tests was conducted to assess and improve the different architecture models, with each test changing one parameter to isolate its impact.

First, the impact of different model parameters and learning rates was tested. The translated image was also passed into the models once again to test if the model could be used several times to slowly enhance the translated image.

3.5.1 Loss-functions

Testing if the loss function impacted the model performance was tested. First, the SSIM, LPIPS, and PSNR scores, calculated with the Python library "TorchMetrics", were used [26]. These scores were modified to $\lambda_{SSIM} = (1 - SSIM)$, $\lambda_{LPIPS} = LPIPS$, $\lambda_{PSNR} = 1/(1 + PSNR)$ to ensure the scores decreased the more similar the generated image and target image was. The MSE and L1 loss was then tested.

Some perceptual loss functions were also obtained by using a pre-trained VGG19 model and tested. All layers up to the third, eighth, thirteenth, seventeenth, and twenty-fourth layers were used for perceptual loss. The images were passed through these layers, and later compared using the modified SSIM loss to give the perceptual loss.

3.5.2 Segmentation

There were three different tests to verify the segmentation extractor's importance. The first compared the SEG when trained with and without a segmentation. The second test was to train the model with an inaccurate segmentation, caused by dilation and erosion. The last test was done by using an inaccurate segmentation to swap backgrounds but accurate segmentation as input, as seen in figure 3.3. This was done to see the effects of not completely replacing the in-studio background during training.

Figure 3.3: Image showcasing in-studio image after swapping the background. On the left, the original segmentation was used to swap the background. On the right, the segmentation got dilated before swapping the background. The original images come from Viton HD dataset [1].

3.5.3 GAN

Various tests to make the GAN model work properly were made. The parameter C in equation 3.1 was modified to be between 1/1500 and 1. The discriminator was also tested by changing how often it got trained in relation to the generator. The SEG-GAN model was trained with only the Viton dataset, except for one test where it was trained on both Viton and F550k datasets. Lastly, the SEG-GAN was analyzed from both metrics and how the learning rate for the generator and discriminator changed during the training.

3.5.4 Final models

The SEG and SEG-GAN had the best performance and were used as the final models. These models were trained for a longer time than the other tests (25, 40 epochs each), and used the Pytorch learning rate scheduler "ExponentialLR" starting at 1E-3. The final SEG model used layers channels (3,170,340,680,1360,2720), with roughly 10Gb worth of GPU memory, and the SEG-GAN model used layer channels (3,170,340,680,1360,2720), with roughly 9Gb worth of GPU-memory. The SEG-GAN model used $C = 1/1500$ and trained the discriminator for each iteration the generator was trained.

To compare the final results, a base model was used. The base model used the segmentation extractor to copy-paste the person onto a general in-studio background image. The background image was created by removing the human from one random in-studio image and filling the gap using Photoshop's Content-Aware Filling.

4
Results

There were many experiments made, thus some lesser important images and tables will be found in the appendix. All tables columns contain one bold marked value noting the best score. The tables' metric scores also contain arrows, where an arrow up indicates that a higher score is better.

4.1 Parameter experiments

First, the learning rate was tested and showcased little difference given the learning rate was smaller than 0,1. If it was bigger the model performed poorly. The architecture structure was then tested with metrics scores in table A.1 and figures illustrating the result in figure A.1. The Experiment showcased that skip connections were quite important to get non-blurry images, and having a model with more layers and smaller $out_{channels}$ (and fewer model parameters) performed better than one with bigger $out_{channels}$ and fewer layers. The SEG model seemed to perform slightly better than the MEG model, and VGG-SEG model in metrics, seen in table 4.1. From a subjective evaluation, the generated images looked identical.

The different Loss functions were tested and can be seen in table 4.2, which showcases SSIM-loss performed best for the SSIM score, VGG[13] for the LPIPS score, VGG[3] for the PSNR score and MSE for the FID score. Generated images confirm that the model trained with LPIPS, VGG[3], and VGG[17] in table 4.2 performs badly, and are found in figure A.2.

The SEG-GAN found no optimal value of C from equation 3.1. However, The GAN-loss from the generator and discriminator can be seen for C=5, 1/1500 in figure 4.1, alongside the SSIM loss. For C=5 the GAN-loss seems to be constant, and for C=1/1500, the generator loss is shrinking while the discriminator loss is growing. The model also produced worse results when trained over two datasets rather than just trained on the in-studio dataset, and can be seen in figure A.3.

Table 4.1: SEG model compared to VGG-SEG and MEG Model

version	$ssim_{viton} \uparrow$	$lpips_{viton} \downarrow$	$psnr_{viton} \uparrow$	$ssim_{F550k} \uparrow$	$lpips_{F550k} \downarrow$	$psnr_{F550k} \uparrow$	$fid \downarrow$
MEG	0.5778	0.4112	10.0309	0.6078	0.4337	9.9652	0.9399
SEG	**0.5791**	**0.4027**	**10.0659**	**0.6104**	**0.4001**	**10.0677**	1.0415
VGG-SEG	0.5790	0.4088	9.8130	0.6063	0.4129	10.0183	**0.9306**

Table 4.2: Different loss functions were tested, where VGG[3]-VGG[24] are the perceptual losses described in section 3.5.1.

version	$ssim_{viton} \uparrow$	$lpips_{viton} \downarrow$	$psnr_{viton} \uparrow$	$ssim_{F550k} \uparrow$	$lpips_{F550k} \downarrow$	$psnr_{F550k} \uparrow$	$fid \downarrow$
MSE	0.5758	0.4134	10.0425	0.6060	0.4255	9.9538	**0.8985**
SSIM	**0.5862**	0.4207	10.1318	**0.6134**	0.4133	9.9934	0.9099
LPIPS	0.0060	0.4192	10.1097	0.0042	0.4062	9.8223	1.0492
PSNR	0.5823	0.4025	10.0681	0.6114	0.4026	10.0111	1.0211
L1	0.5797	0.4030	9.5577	0.6106	0.3973	9.7139	1.0101
VGG[3]	0.0081	0.7678	**22.3621**	0.0067	0.7673	**22.1023**	1.1297
VGG[8]	0.5859	0.4020	9.5940	0.6117	0.3915	9.5868	1.1124
VGG[13]	0.5794	**0.3985**	9.8036	0.6022	**0.3912**	9.7254	1.0800
VGG[17]	0.0014	0.7881	16.4730	0.0008	0.8007	16.6907	1.1628
VGG[24]	0.5699	0.4029	9.9406	0.5994	0.3928	10.0173	1.1345

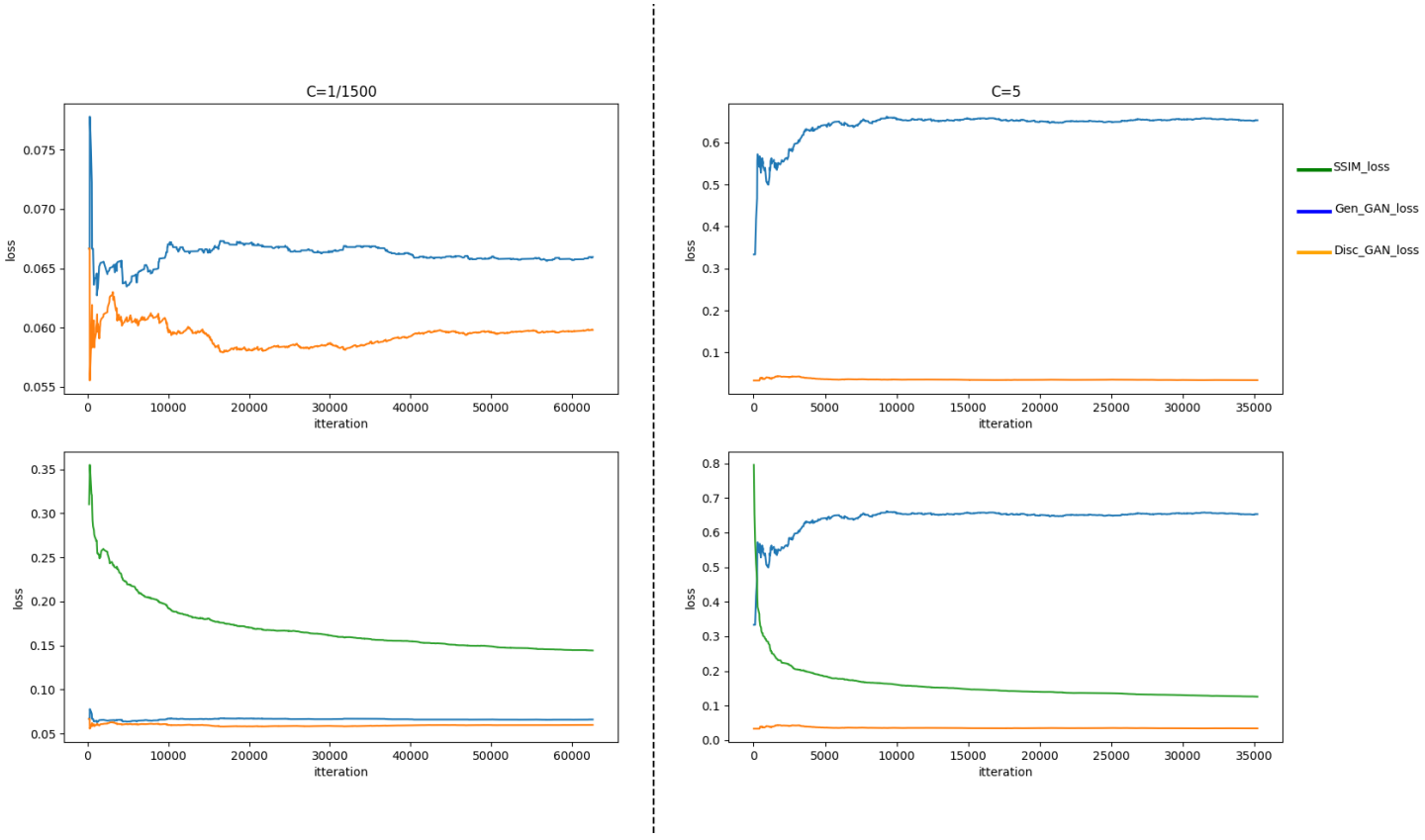

Figure 4.1: How loss function for a trained model with C=1/1500, and C=5, with the green loss is the SSIM loss used by the generator, the blue and orange loss is the GAN loss of the discriminator and generator. The loss is plotted with an Exponential sliding average applied.

4.2 Segmentation quality

The importance of the segmentation extractor was tested and metrics can be seen in table 4.3. Model A was trained using the segmentation without any augmentation. Model B and C both applied erosion and dilation to the segmentation but in different ways. Model C augmented the segmentation that was concatenated with the input image, while model B instead augmented the segmentation used to swap the input image's background. Model D did not concatenate any segmentation to the input image. The base model is the same as described in section 3.5.4. According to the table 4.3 C and D got the best results. However, the model's performance on a random image can be seen in figure 4.2, where the A performed the best.

Figure 4.2: How changing the segmentation affects generated images, here A contains no dilation to segmentation or background, B dilates the background with a kernel of size 10, C dilates the segmentation by 10, and D does not use any segmentation. The original images come from Fashion550k [21].

Table 4.3: Metrics when trained in three different ways. A) Using original segmentation, B) dilates segmentation when swapping background, C) dilates segmentation when concatenating with the input image, and D) no segmentation used at all. The base model refers to copy-pasting the input image into a studio background.

version	$ssim_{viton} \uparrow$	$lpips_{viton} \downarrow$	$psnr_{viton} \uparrow$	$ssim_{F550k} \uparrow$	$lpips_{F550k} \downarrow$	$psnr_{F550k} \uparrow$	$fid \downarrow$
BASE	0.5752	0.4136	9.8579	0.6071	0.4064	9.6452	**0.8719**
A	0.5789	0.4063	10.1390	0.6100	**0.4049**	10.0401	1.1253
B	0.5691	0.4126	9.8885	0.5908	0.4267	10.2472	1.1097
C	0.5799	0.4066	9.9320	**0.6136**	0.4073	10.0393	1.0916
D	**0.5850**	**0.4017**	**10.1830**	0.5750	0.4202	**10.3438**	1.2083

4.3 Final models

The final models' evaluation scores can be seen in table 4.4 with an illustration of the final models seen in figure 4.3. The figure shows random augmented images where A1, A2, and A3 represent the segmentation, the augmented image, and the original image. B and C were generated images by the final SEG, and SEG-GAN models. In figure 4.4 specific images and augmentations were instead chosen to highlight what

the model has learned. On the left, the whole image is showcased, and on the right specific parts have been cropped and enlarged.

Table 4.4: Quantitative metrics of final models.

version	$ssim_{viton}\uparrow$	$lpips_{viton}\downarrow$	$psnr_{viton}\uparrow$	$ssim_{F550k}\uparrow$	$lpips_{F550k}\downarrow$	$psnr_{F550k}\uparrow$	$fid\downarrow$
BASE	0.5752	0.4136	9.8579	0.6071	0.4064	9.6452	**0.8719**
SEG	0.5979	0.3995	10.0442	0.6034	0.4005	9.4234	1.0872
SEG-GAN	**0.6007**	**0.3975**	**10.1599**	**0.6044**	**0.3998**	**9.6153**	1.0729

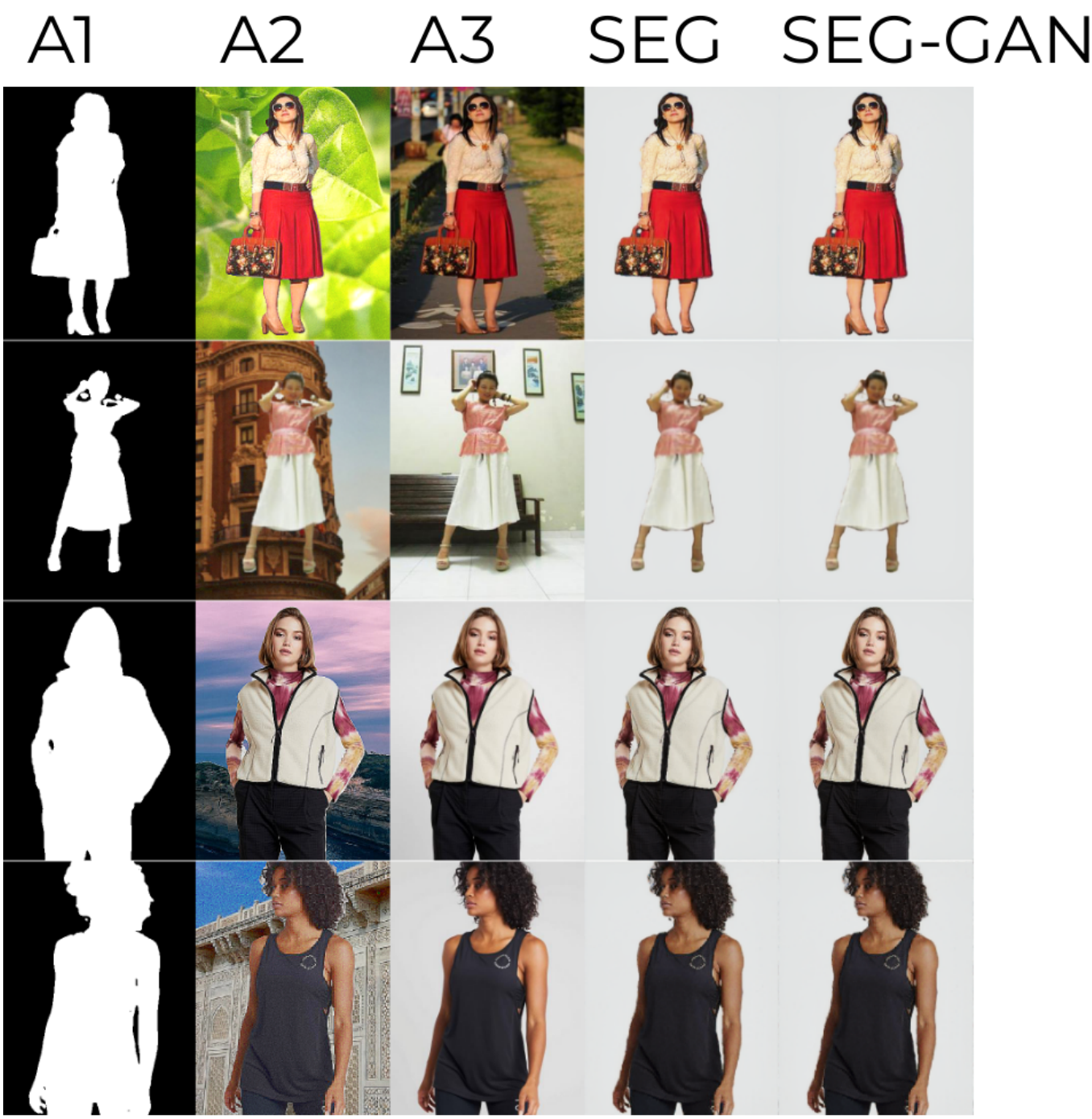

Figure 4.3: A1, A2, and A3 represent the segmentation, the augmented image, and the original image. The generated images by the two final models are showcased on the right. The top two images are using the F550k dataset and the bottom two are with the Viton dataset [1][21].

Figure 4.4: Showcase of how the SEG-GAN transfers chosen images with certain augmentations with the whole image on the left and cropped parts on the right. The original non-augmented images come from and with Zalando's permission.

5
Discussion

5.1 Evaluation

The evaluation metrics used in the study displayed some peculiar behavior that needs to be addressed. While the metrics largely seem capable to evaluate the images, there were some exceptions and inconsistencies observed.

When looking at the PSNR score for VGG[3], and VGG[17] in table 4.2, it can be noticed that the PSNR score is unexpectedly high while the SSIM and LPIPS scores are low, indicating that the PSNR score should not be used as an evaluation score. This seems reasonable given that the score is said to not align well with human perception. More discrepancies were also noticed as training a model without using a segmentation gave better metrics in table 4.3. This contradicts the seen results in figure 4.2 where the model performs much worse than the model trained with a segmentation.

In table 4.2 the SSIM score is much worse than other scores for the model trained with LPIPS loss, which corresponds with figure A.2. The SSIM score was however too low given the generated image and is better reflected in the other scores. This showcases that multiple scores help evaluate a model. However, it is still hard to solely rely on the scores or to know which score should hold the most weight for the evaluation.

The root cause behind the weird behavior is not clear, but it is likely attributed to the evaluation approach for the SSIM, PSNR, and LPIPS scores. The scores have been shown to be utilized when comparing images to either augmented counterparts or different images that contain many similarities. The images compared in this study may have been too different for the score to be reliable. To address this issue some changes to the evaluation approach will be considered.

One approach is to not evaluate the in-the-wild images and only evaluate in-studio images. The translated in-studio image could then be compared with the original in-studio image giving a more robust score.

Another approach is to compile a new dataset by augmenting and passing the in-studio dataset into some ANNs like a style transfer network, or image harmonization network, among many others. By utilizing many ANNs alongside some augmenta-

tions, the goal is to further make the images resemble natural in-the-wild images. Although this new dataset might not resemble in-the-wild images completely, it will still provide a more robust and accurate evaluation of in-the-wild images.

The last approach although expensive suggests creating a new dataset containing two different images of the same person in the same clothes. While fine-grained details like hair position will not be captured, it should mitigate discrepancies that arise from comparing unrelated images and provide a more controlled and robust evaluation.

5.2 Importance of feature extractor

As can be observed in figure 4.2, the SEG model appears to have learned to in-paint the human based on the segmentation, even disregarding parts of the leg that were not included in the segmentation. This behavior affects the transfer of areas such as hair, where the remnants of the original background still remain. This shows that the segmentation extractor played quite an important role. The reasons behind the in-painting can thus be likely attributed to the high percentage of segmentation overlapping with the target image's person.

It is important to state that the results for the segmentation extractor can not be extended to all feature extractors. However, if an extractor provides important information that spatially correlates with the image, similarly to the segmentation extractor, it is reasonable to assume that similar results apply. The results are also only conclusive for simpler models, whereas, this might not be the case for models with more complex operations or layers(not deeper models). Lastly note that the MEG model might have performed better if more and various types of feature extractors were used, but is inconclusive from the results.

To improve the model's performance, one suggested improvement is to instead in-paint the model onto a general background before passing it into a model. This would shift the model's focus from in-painting to changing lighting, applying augmentations, and other transformations. This approach also aligns more closely with how relighting and style transfer problems are handled which have been more studied.

Another potential improvement would be to use an Image Matting extractor instead of a segmentation extractor which would help transfer e.g. hair and glasses more accurately. Lastly, to ensure that the model can translate in-the-wild images some restrictions might be necessary, even if it might limit the try-on for customers. These limitations could be images with a certain resolution or only using model images.

5.3 Models performance

It is observed that the MEG and VGG-SEG Models performed slightly worse than the SEG model in the different metrics. However, the differences were not easily noticeable when looking at generated images. As the VGG-SEG model performed equally well to the SEG it indicates that the training set is not too small to learn useful representations. The final model's metrics show that the SEG-GAN performed slightly better in comparison to SEG but is hard to tell from generated images.

In figure 4.4, it is observed that the SEG-GAN is capable of handling sharp images, and noise to make them more smooth. However, it struggles more with unblurring blurry images. Similarly, the model demonstrates some ability to adjust the lighting and recover slight detail from bright images. The bottom and top images in figure 4.3 further showcase the model's ability to fix lighting with e.g. the red skirt, and black tank top. The model also has areas it fails completely, like the removal of shadows, sun glares, or fixing the lighting caused by e.g. a red lamp.

One can see in the zoomed-in base model for figure 4.4 that the person does not blend in with the background. Instead for the un-augmented and translated zoomed-in image the person blends in much better, although some blurriness can also be seen on the pants. According to the goal in section 1.2, This thesis achieved transferring a human from the in-the-wild domain to an in-studio domain. However, some effects like unblurring images, removal of shadows, and other augmentations were more minimal than hoped for.

It is very likely that the models used in this study did not use optimized training parameters. Although various parameters were explored, the results did not significantly change. It suggests that focusing on a different model architecture such as a diffusion model, or using different training data might have been more beneficial for achieving improved results. Parameter C in equation 3.1 should however be varied more. Figure A.3 illustrates that the loss for the final model's ($C = 1/1500$) generator decreased when the discriminator's loss increased. This is to be expected from a well-trained GAN. However, the two losses did not oscillate back and forth as one would also expect. This could potentially be attributed to training with both GAN and SSIM losses which were unbalanced. Another reason could be that parameter C was the same for both the generator and discriminator, causing the generator to learn much faster. To address these issues, experimenting with more variations to balance the losses could be made. Additionally training the model in stages, where first only trained using SSIM loss, and later only using GAN loss, could potentially stabilize the training more.

The usage of using GAN loss with both in-studio images, and also generated in-the-wild images was thought to bring improvements to the model. The results instead showcase that the model trained with both performed worse. Given that experiment only trained the model for eight epochs, and the SSIM and GAN loss was still unbalanced, it is difficult to evaluate if the method of using both datasets were a

bad idea. In short, the SEG-GAN showed the best performance in metrics but also shows the most potential for improvement when the losses have been balanced.

5.4 Augmentations

Originally chosen augmentations such as random shadows and sun flares, were expected to help the model to remove shadows or sun flares. The augmentations did neither. As the model had a small effect on other augmentations these augmentations could have been too extreme for the model to remove. Another reason could be that the augmentations did not look realistic and were unable to generalize the removal of real shadows.

Since the final models struggled to unblur images and remove shadows or strong lighting, but did slightly blur and remove noise from the images, some augmentations could be used to reduce the problem. For the lighting, there are some HDR-related projects which try to fix areas of an image that are too bright or dark. To remove the shadows there is no current augmentation that adds or removes shadows from the human body realistically. Instead, the usage of different Photoshop tools allows for the quick addition of somewhat realistic shadows on the human body to create a new augmented version of the Viton dataset. Lastly, the input image could be passed into a trained super-resolution model to help unblur the image and improve the overall image quality. Improving the image quality could potentially improve the segmentation extractor's performance, which affected the used models.

Another approach could have been to use multiple models, each focusing on handling a single augmentation. The models could then either be applied sequentially or be used as input to a model that pieces together relevant augmentations.

6
Conclusion

In conclusion, the evaluation metrics used in the study exhibited peculiar behavior. While the metrics generally provided a means to compare images and reflected the model's learning performance, there were some exceptions. The most likely cause is the evaluation approach. To address this, the in-the-wild dataset should only be evaluated using the FID score, while other scores only compare in-studio and translated in-studio images. Evaluation of in-the-wild images is still desired, and two potential methods are suggested. The first is the augmentation of the in-studio dataset with background swapping and style transfer and the second is to gather two different images of the same person wearing the same clothes.

The feature extractor greatly influenced image quality and would likely be the case for other extractors that spatially correlate with the target image. The model in-painted the person unto a background based on the segmentation, leading to difficulties in transferring areas like hair. One improvement would be to in-paint the human onto a general in-studio background before inputting it into a model. Using an Image Matting extractor could enhance accuracy for features like hair and glasses. Restriction on images used may be necessary for better segmentation.

The MEG and VGG-SEG models slightly underperformed compared to SEG, while the SEG-GAN model showed potential, it required further examination as hyperparameters were unoptimized. The final models handled sharpness, lighting, and noise but had minimal impact on blurred images or images containing shadows among others. Potential solutions included HDR-related projects for lighting, photoshop tools for realistic shadows, and a super-resolution model for image refinement.

This thesis successfully achieved the goal of transferring humans from the in-the-wild domain to an in-studio domain and demonstrated better effects than copy-pasting the person into a studio background.

Bibliography

[1] Choi, Seunghwan, et al. "VITON-HD: High-Resolution Virtual Try-On via Misalignment-Aware Normalization" Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR), 2021.

[2] Ge, Chongjian et al. "Disentangled Cycle Consistency for Highly-realistic Virtual Try-On." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 16928–16937, 2021.

[3] Ge, Yuying, et al. "Parser-free virtual try-on via distilling appearance flows." Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2021.

[4] Liu, Daquan, et al. "Arshadowgan: Shadow generative adversarial network for augmented reality in single light scenes." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[5] Cun, Xiaodong, et al. "Towards ghost-free shadow removal via dual hierarchical aggregation network and shadow matting GAN." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 07. 2020.

[6] Ramesh, Aditya, et al. "Hierarchical text-conditional image generation with clip latents." arXiv preprint arXiv:2204.06125 (2022).

[7] Alidoost, Fatemeh, et al. "2D image-to-3D model: Knowledge-based 3D building reconstruction (3DBR) using single aerial images and convolutional neural networks (CNNs)." Remote Sensing 11.19 (2019): 2219.

[8] Kubiak, Nikolina, et al. "Silt: Self-supervised lighting transfer using implicit image decomposition." arXiv preprint arXiv:2110.12914 (2021).

[9] Afifi, Mahmoud. "Semantic white balance: Semantic color constancy using convolutional neural network." arXiv preprint arXiv:1802.00153 (2018).

[10] Wang, Lin, and Kuk-Jin Yoon. "Deep learning for hdr imaging: State-of-the-art and future trends." IEEE transactions on pattern analysis and machine intelligence 44.12 (2021): 8874-8895.

[11] El Helou, Majed, et al. "NTIRE 2021 depth guided image relighting challenge." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[12] Gafton, Paul, and Erick Maraz. "2d image relighting with image-to-image translation." arXiv preprint arXiv:2006.07816 (2020).

[13] Hang, Yucheng, et al. "Scs-co: Self-consistent style contrastive learning for image harmonization." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.

[14] Chen, Haoxing, et al. "Hierarchical Dynamic Image Harmonization." arXiv preprint arXiv:2211.08639 (2022).

[15] Gatys, Leon A., et al. "A neural algorithm of artistic style." arXiv preprint arXiv:1508.06576 (2015).
[16] Huang, Xun, and Serge Belongie. "Arbitrary style transfer in real-time with adaptive instance normalization." Proceedings of the IEEE international conference on computer vision. 2017.
[17] Zhu, Jun-Yan, et al. "Unpaired image-to-image translation using cycle-consistent adversarial networks." Proceedings of the IEEE international conference on computer vision. 2017.
[18] Luan, Fujun, et al. "Deep photo style transfer." Proceedings of the IEEE conference on computer vision and pattern recognition. 2017.
[19] Li, Yijun, et al. "A closed-form solution to photorealistic image stylization." Proceedings of the European conference on computer vision (ECCV). 2018.
[20] Ziwei, Liu, et al. "DeepFashion: Powering Robust Clothes Recognition and Retrieval with Rich Annotations" Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016.
[21] Inoue, Naoto, et al. "Multi-Label Fashion Image Classification with Minimal Human Supervision" Proceedings of ICCVW, 2017.
[22] Li, Peike, et al. "Self-Correction for Human Parsing" IEEE Transactions on Pattern Analysis and Machine Intelligence, 2020. doi: 10.1109/T-PAMI.2020.3048039
[23] Park, GyuTae, et al. "Matteformer: Transformer-based image matting via prior-tokens." Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022.
[24] Li, Jizhizi, et al. “Bridging Composite and Real: Towards End-to-end Deep Image Matting.” International Journal of Computer Vision 130.2 (2022): 246-266.
[25] Buslaev, Alexander, et al. "Albumentations: fast and flexible image augmentations" ArXiv e-prints, eprint = 1809.06839, 2018.
[26] Detlefsen, Nicki Skafte, et al. "TorchMetrics - Measuring Reproducibility in PyTorch" Journal of Open Source Software, Journal of Open Source Software 7.70 (2022): 4101.
[27] Paszke, Adam, et al. "Automatic differentiation in PyTorch", 2017.

A

Appendix 1

Table A.1: Quantitative results for various architecture changes with the MEG model. A1, A2 used different number of layers and $out_{channels}$ as A1=(64), A2=(32,64,128,256). B1 used no skip connections while B2 did. Lastly, C Represents using the regular MEG model but where the generated images are passed through the model once more.

version	$ssim_{viton} \uparrow$	$lpips_{viton} \downarrow$	$psnr_{viton} \uparrow$	$ssim_{F550k} \uparrow$	$lpips_{F550k} \downarrow$	$psnr_{F550k} \uparrow$	$fid \downarrow$
A1	0.5767	0.4161	9.8607	0.6058	0.4306	9.7274	0.8684
A2	0.5789	0.4063	10.1390	0.6100	0.4049	10.0401	0.8631
B1	0.5744	0.4073	10.1622	0.6071	0.4074	10.0964	**0.8531**
B2	**0.5996**	**0.4038**	9.4479	**0.6247**	**0.3920**	9.2068	0.8676
C	0.5012	0.4448	**12.7379**	0.4185	0.5225	**11.4275**	0.9317

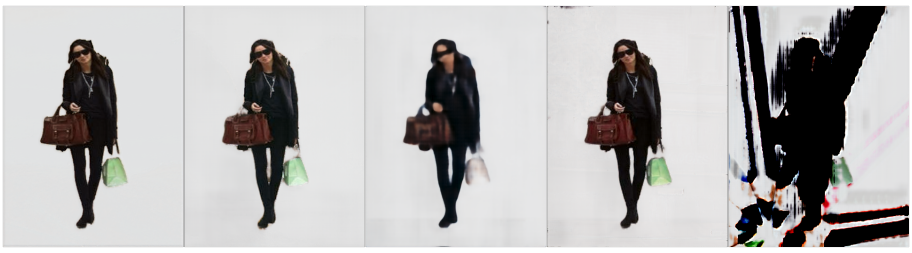

Figure A.1: Generated images for various architecture changes. A1, A2 used different number of layers and $out_{channels}$ as A1=(64), A2=(32,64,128,256). B1 used no skip connections while B2 did. Lastly, C Represents using the regular MEG model but where the generated images are passed through the model once more. The original images come from the Viton HD dataset [1].

Figure A.2: Generated images when the SEG model was trained on different loss functions. From Top left to bottom, right are: MSE, SSIM, LPIPS, PSNR, L1, VGG[3], VGG[8], VGG[13], VGG[17], VGG[24]. The original images come from the Viton HD dataset [1]

Figure A.3: On the left side an image generated by SEG-GAN trained on only the Viton dataset, and on the right trained on both Viton and F550k dataset, where both models were trained for 8 epochs. The original images come from the Viton HD dataset [1]

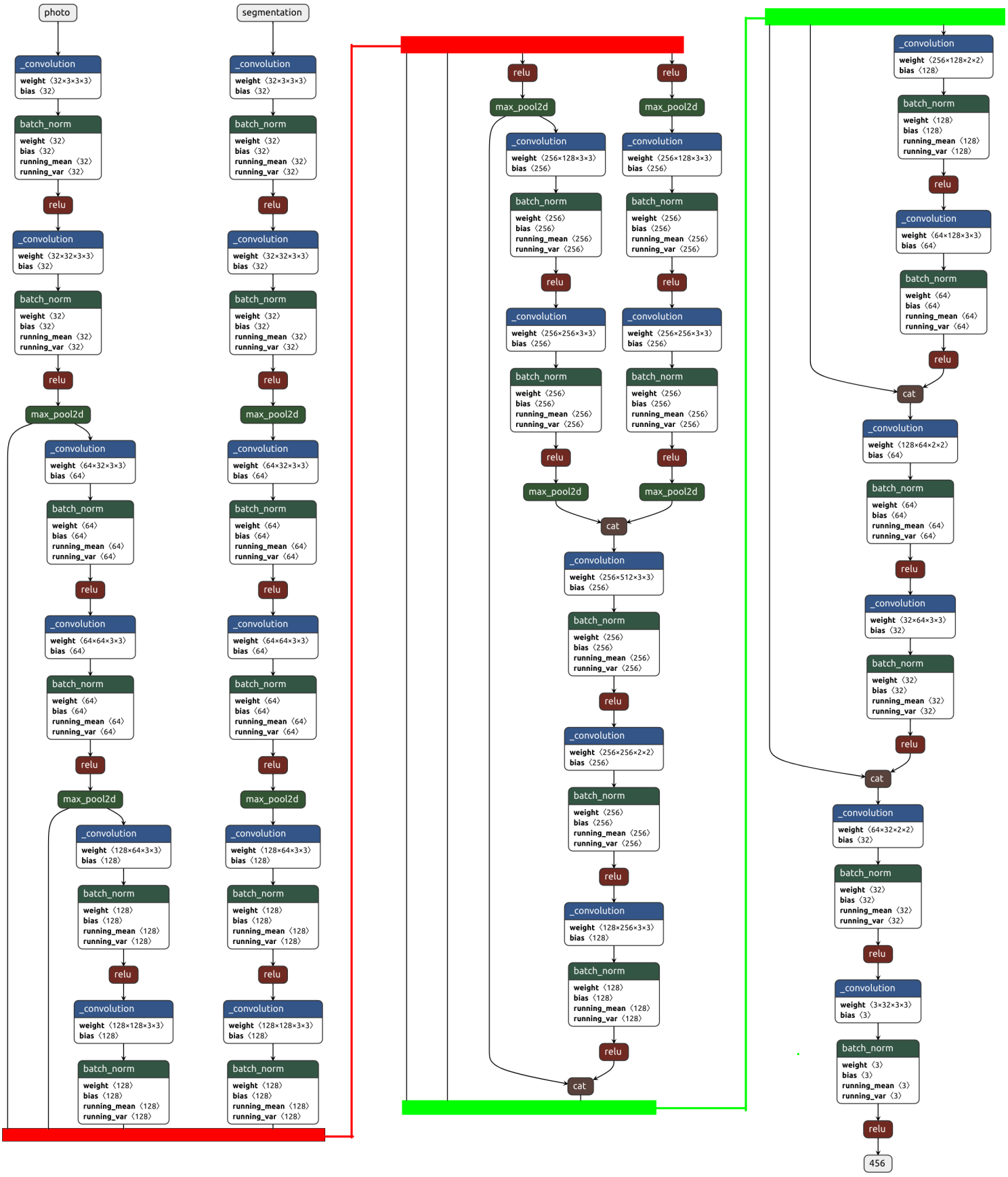

Figure A.4: Schematic of MEG model, where the red and green lines and boxes simply show where the schematic continues

Figure A.5: Showcase of random images from F550k dataset[21].

Figure A.6: Showcase of random images from Viton dataset [1]

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY