High-resolution virtual try-on with garment extraction using generative adversarial networks

Authored by

Daniel J Charters

Supervised by

Stefan S Britz

Co-supervised by

Dino Bernicchi

A minor dissertation in partial fulfilment of the requirements for the degree

Masters of Science - Data Science

in the

Faculty of Science

University of Cape Town

30 April 2024

Abstract

Title:	High-resolution virtual try-on with garment extraction using generative adversarial networks
Student name:	Daniel J Charters
Student number:	CHRDAN015
Supervisor:	Stefan S Britz
Co-supervisor:	Dino Bernicchi

Image-based virtual try-on aims to depict an individual wearing a garment not originally worn by them. While existing literature predominantly focuses on garments from standalone images, this research addresses the use of images where the garment is already being worn by another individual. The study bridges a notable gap as most current systems are tailored for standalone garment images.

The proposed system, given a pair of high-resolution images, extracts the garment from one, refines it using context-aware image inpainting, and subsequently transfers it onto the second image's subject. The methodology incorporates various off-the-shelf models, notably Part Grouping Network (PGN), Densepose, and OpenPose for pre-processing. A state-of-the-art context-aware inpainting model refines the garments, and the final synthesis leverages the HR-VITON architecture, producing images at a resolution of 768×1024. Distinctively, our model processes both standalone and garment-on-person images.

Evaluating the models involves testing on 2 032 high-resolution images under both paired and unpaired conditions. Metrics such as RMSE, Peak Signal-to-Noise Ratio (PSNR), Learned Perceptual Image Patch Similarity (LPIPS), Structural Similarity (SSIM), Inception Score (IS), Fréchet Inception Distance (FID), and Kernel Inception Distance (KID) assessed the model's prowess. Benchmarked against HR-VITON, ACGPN, and CP-VTON, our model slightly trailed HR-VITON but notably surpassed ACGPN and CP-VTON. In realistic, unpaired conditions, the model achieved an IS of 3.152, an FID of 15.3, and a KID of 0.0063. This is compared to an IS of 3.398, an FID of 11.93, and a KID of 0.0034 achieved by HR-VITON on the same data. ACGPN has an FID of 43.29, and a KID of 0.0373, while CP-VTON has an FID of 43.28, while it has a KID of 0.0376. IS is not measured for both ACGPN and CP-VTON. An ablation study underscored the importance of context-aware inpainting in our network.

The findings highlight the model's ability to generate convincing, high-resolution virtual try-on images from garment-on-person extractions, addressing a prevalent gap in the literature and offering tangible applications in high-resolution virtual try-on image generation.

Acknowledgements

I wish to acknowledge those who have played a pivotal role in this research and the broader completion of my master's degree. Special thanks to Dino Bernicchi for advising on my research topic and to Stefan Britz, my supervisor, for consistent guidance over the last two years.

To my parents, Rob and Shan Charters, as well as Charles and Vanessa Cadman: your support enabled my academic pursuit — this achievement would not have been possible without you. Rebecca Cadman deserves special mention for her unwavering support throughout the last two years.

Lastly, I am deeply grateful to the Lord for blessing me with the ability and opportunity to realise a lifelong dream.

To all of you, your efforts and support have been invaluable. Thank you!

Contents

List of Figures

List of Tables

Acronyms

ACGPN	Adaptive Content Generating and Preserving Network
ACM	Attention Computing Module
Adam	The Adaptive Moment Estimation
ALIAS	Alignment-Aware Segment
ASPP	Atrous Spatial Pyramid Pooling
ATM	Attention Transfer Module
CNN	Convolutional Neural Network
CPU	Central Processing Unit
CP-VTON	Characteristic-Preserving Virtual Try-On Network
CRA	Contextual Residual Aggregation
DCGAN	Deep Convolutional Generative Adversarial Network
EDA	Exploratory Data Analysis
ELU	Exponential Linear Unit
ES	Embedding Similarity
FCN	Fully Convolutional Network
FID	Fréchet Inception Distance
FFB	Feature Fusion Block
GAN	Generative Adversarial Network
GC	Gated Convolution
GPU	Graphics Processing Unit
HR-VITON	High-resolution Virtual Try-On Network
IS	Inception Score
KID	Kernel Inception Distance
LPIPS	Learned Perceptual Image Patch Similarity
LWGC	Light Weight Gated Convolution
MG-VTON	Multi-pose Guided Virtual Try-on Network
MSE	Mean Squared Error

MTCNN	Multi-Task Cascaded Convolutional Networks
PGN	Part Grouping Network
PSNR	Peak Signal-to-Noise Ratio
ReLU	Rectified Linear Unit
ResBlk	Residual Block
RMSE	Root Mean Squared Error
SHCP	Self-Correction for Human Parsing
SPADE	Spatially Adaptive Denormalisation
SSIM	Structural Similarity
TPS	Thin Plate Spline
VITON	Virtual Try-On Network
VITON-HD	Virtual Try-On Network with High-definition
VTNFP	Virtual Try-On Network with Feature Preservation

1 Introduction

The online retail industry is worth approximately $6.31 trillion in 2023, accounting for almost a quarter of all retail sales worldwide, and this figure is expected to grow rapidly in the coming years (Cramer-Flood, 2022). Over a third of all online purchases are clothes, which has prompted the online apparel shopping market to create innovative solutions to elevate the customer experience and streamline the purchasing process. Notably, in 2022, the most compelling means to boost conversions emerged from immersive shopping experiences, such as augmented reality and virtual try-on. For instance, Shopify highlighted that when augmented reality is deployed for product visualisation, conversions see a twofold surge (Elgaard, 2023). Such innovations, especially virtual try-on, have captured the keen interest of both researchers and industry experts.

Virtual try-on technologies offer a promising solution to the challenges associated with online apparel shopping by allowing customers to visualise how garments will look on them without the need for a physical fitting room. Various approaches have been proposed to tackle this challenge, such as 3D-based methods that rely on the 3D measurement of garments, and image-based methods that only require a garment and a person image. While 3D-based methods provide realistic clothing simulations, their widespread adoption is limited by the high costs associated with hardware installation and 3D annotated data collection (Han et al., 2017).

Image-based virtual try-on techniques, on the other hand, present a more accessible alternative, as they only require plain images in RGB (red, green, and blue colour channels) format without leveraging any 3D information. Existing image-based methods, such as Virtual Try-On Network (VITON) and its variants, have made significant progress in generating realistic virtual try-on results. However, these methods face limitations in generating detailed visual patterns, accommodating geometric changes, and considering clothing size variations, particularly when the garment is not in a standalone image.

This dissertation will delve into the development of a high-resolution virtual try-on system that addresses the limitations of existing methods while incorporating garment extraction. The proposed system will explore novel techniques and architectures for the seamless transfer of clothing items from one image to the corresponding regions of a clothed person in another image. To lay the foundation for our discussions, we first seek to understand what image-based virtual try-on entails, which is discussed in the next section.

1.1 Image-based virtual try-on overview

Image-based virtual try-on is a method for simulating how clothing items would look on a person using only two 2D images. The two images consist of the clothing item to be tried on and the target person, who will virtually wear the clothing item. This approach relies on computer vision techniques and machine learning algorithms to process and manipulate images, transferring a garment from one image to another. Image-based virtual try-on methods often involve the

extraction of the clothing item from its original image, warping and adjusting it to match the target person's body shape and pose, and finally blending it seamlessly onto the target image. This process is best summarised through images, as seen in Figure 1.1.

(a) Target garment. (b) Target person. (c) Result.

Figure 1.1: Traditional virtual try-on overview showing the target garment in a standalone image, the target person, and the resulting virtual try-on.

The garment, shown in the first image, is virtually tried on by the person in the second image. The result of this try-on is shown in the last image, where the person is wearing the target garment, as opposed to their original shirt. It can be seen that the person's identity (their eyes, skin colour, hair, etc.) was maintained, a crucial part of virtual try-on, as any virtual try-on system needs to adequately represent the person who is trying on the clothes.

The majority of existing virtual try-on methods work like this, where they take a garment in a standalone image and a clothed person image to get the desired result. However, this approach may not be robust enough to handle when the garment is not in a standalone image, meaning the garment first needs to be extracted and then tried on. The virtual try-on systems that allow for garment transfer or extraction first are not as advanced as those that handle standalone garment images, as their results do not have as good photorealism (Han et al., 2017). Again, to understand how this type of try-on system works, see Figure 1.2, where the target garment is now in a clothed person image, as opposed to being standalone (as in Figure 1.1a).

(a) Target garment. (b) Target person. (c) Result.

Figure 1.2: Virtual try-on with garment extraction overview showing the target garment being worn by a person in an image, the target person, and the resulting virtual try-on.

As before, the target garment is placed onto the target person to get the resulting try-on image. Again, the person's identity is maintained, and occluded body parts are synthesised as well (for example, her arms were hidden in the target person image, and have been 'predicted' to

appear as they do in the resulting synthesised image). Occluded areas of the garment are also synthesised in the resulting image. These are all considerations that need to be taken into account when building a robust virtual try-on system.

From these images, we get an understanding of what imaged-based virtual try-on entails, and what the results should look like. The two figures show slight differences around the target garment, Figure 1.1 shows the virtual try-on process when the target garment is standalone, whereas Figure 1.2 shows the virtual try-on process when the target garment first needs to be extracted from an image and then tried on. The latter of the two is the least researched, however, it is deemed to be more practical in its application. However, the methods that can achieve this do not produce high-resolution results. This is the gap in the literature that is identified for this study, and it entails creating a high-resolution virtual try-on system with garment extraction. The specific aims of this study are highlighted next.

1.2 Research objectives

This study aims to build a virtual try-on system that can generalise at test time, meaning perceptually convincing, high-resolution images are synthesised for an arbitrary desired clothing item, whether in a standalone image or not. While the research covers a broad scope, its focus is primarily on female tops as the target garment. The main research objectives of this study are as follows:

1. To design and implement an efficient garment extraction algorithm capable of accurately identifying and extracting clothing items from a variety of images, including those with complex patterns, textures, and colours, and that handles occlusions.
2. To investigate and incorporate pose estimation and body shape modelling techniques for better alignment and adaptation of extracted garments to the target person's body in the virtual try-on process.
3. To develop a robust image-based high-resolution virtual try-on system that accurately and realistically represents garments on target individuals, taking into account various body shapes, sizes, and poses, and that handles occlusions.
4. To evaluate the performance of the proposed high-resolution virtual try-on system through quantitative comparisons with existing state-of-the-art methods.
5. To explore potential applications and extensions of the high-resolution virtual try-on system.

These objectives are researched, and the problem of building a high-resolution virtual try-on system with garment extraction is summarised by the research question which asks:

> *How can the high-resolution photorealistic virtual try-on capabilities of the work of Lee et al. (2022) and Choi et al. (2021) be combined with the robust and practical nature of the work of Ishikawa and Ikenaga (2022) and Lewis et al. (2021) to build a system that can generalise at test time?*

Despite the significant potential and positive impact of this research, it is essential to carefully consider the ethical implications associated with generative models and their applications. The subsequent section briefly addresses these ethical considerations.

1.3 Ethical considerations

The advancements in generative models for image synthesis raise significant ethical concerns. As these models become more sophisticated, it is crucial to address the potential ethical implications they present. One key concern is the misuse of generated content. The ability to fabricate realistic images can lead to counterfeit art, fake news, and deceptive visual content that may harm individuals, organisations, public trust, and credibility.

Generative models can have an inherent bias as a consequence of biased training data, so every care must be taken to ensure representative training data are used. These models are also capable of replicating copyrighted works and challenging notions of ownership and creativity, necessitating clear guidelines to protect original creators and owners.

To address these concerns, collaboration among researchers, policymakers, industry stakeholders, and the public is crucial. Developing responsible guidelines, ensuring transparency, and promoting ethical awareness will help harness the potential of generative models while mitigating their negative impacts. By considering ethical implications throughout development and application, we can promote the responsible and ethical use of generative models for image synthesis.

One method to combat the misuse and protect original content is the incorporation of watermarking, both visible and invisible, and image tagging. Watermarking can help verify the authenticity of images and deter unauthorised replication or dissemination. Image tagging, on the other hand, can provide metadata about the image's origin, purpose, or any other pertinent information, assisting users in determining the credibility and context of an image. However, it is worth noting that whilst these methods are available and beneficial, they have not been incorporated into this study due to the primary focus being on the development and refinement of the generative model's capabilities. The inclusion of watermarking and tagging would introduce additional complexities and could detract from the main objective of achieving high-resolution, convincing image synthesis.

Every care has been taken in conducting this research to adhere to ethical guidelines and best practices regarding generative models and their applications. However, it is important to note that the author, his supervisors, and the University of Cape Town cannot be held responsible for any misuse or unintended consequences resulting from the work in this study. Users and stakeholders should exercise caution and ensure responsible usage, taking into account legal, ethical, and societal considerations when applying generative models in practice.

1.4 Document structure

This chapter has introduced image-based virtual try-on, while also describing the objectives of this study and the research question it seeks to answer. The second chapter is a literature review that investigates Generative Adversarial Networks (GANs), virtual try-on systems, garment extraction, and image inpainting. The third chapter discusses and analyses the data used for this study. The fourth chapter covers the research design and methodology, which explains the model formulation, details, and training. The fifth chapter compares the formulated method with the latest techniques and discusses the results, while the final chapter concludes the research, discusses its practical application and limitations, and makes recommendations for future work.

2 Literature review

This chapter reviews the work that addresses the virtual try-on problem, and similar problems to it. The review looks at how those problems were solved, and how those solutions and methods can be applied to this research. An array of different techniques will be researched, including well-established methodologies such as neural networks, Convolutional Neural Networks (CNNs), image segmentation, and encoder-decoders, among others. Although these will be used throughout the research, they will not be explored in depth in the literature review, however, they will be touched on briefly and explained where needed. This chapter focuses on two main areas: Generative Adversarial Networks (GANs), and imaged-based virtual try-on methods.

GANs are a type of machine learning model that generates new data samples in a feedback loop to improve the quality of the generated data. This has particular relevance for this research as we are aiming to synthesise images for virtual try-on. Virtual try-on, in its broad sense, covers most of the research, as it is a broad topic, however, there will be relevant sections that highlight specific aspects and uses of it. We start by understanding what GANs are and their uses for virtual try-on.

2.1 Generative adversarial networks

GANs are a type of deep learning architecture that are used for generative modelling, which is an unsupervised learning technique where regularities and patterns in data are found and exploited in such a way that new data, that is plausibly drawn from the input sample, is generated. GANs consist of two networks: a generator and a discriminator. In this way, the unsupervised problem is framed as two supervised subtasks, or models, where the generator is responsible for generating new data based on input data, while the discriminator tries to distinguish between real and fake data. The two networks work together in a feedback loop, where the generator creates new data samples until the discriminator can no longer distinguish between real and fake data. The goal is for the generator to produce data that is indistinguishable from the real data. GANs, introduced by Goodfellow et al. (2014), can be used in a variety of applications, including image and video generation, text generation, and even music generation.

GANs can be best described by an analogy; where Goodfellow et al. (2014) state: *"The generative model can be thought of as analogous to a team of counterfeiters, trying to produce fake currency and use it without detection, while the discriminative model is analogous to the police, trying to detect the counterfeit currency. Competition in this game drives both teams to improve their methods until the counterfeits are indistinguishable from the genuine articles."* In their paper, Goodfellow et al. (2014) demonstrate how GANs can generate fake images that resemble real data. They generated fake images of human faces, animals, and numbers that resemble the popular MNIST handwriting dataset (Deng, 2012) that are indistinguishable from the real data. The authors highlight that as long as the data can be curated, GANs can be used to generate fake samples based on it.

Although Goodfellow et al. (2014) first introduced GANs, the majority of its application is based on Deep Convolutional Generative Adversarial Networks (DCGANs), developed by Radford et al. (2016a). Image-generating DCGAN models are generally more stable and produce better results than image-generating GAN models. This is attributed to the fact that DCGANs use convolutional layers instead of fully connected layers, which GANs use, in the generator network. This allows them to capture spatial information and learn hierarchical representations of the input data, which is particularly useful for image generation, where spatial relationships between pixels are important. Another advantage of DCGANs is that they use batch normalisation, which helps to normalise the inputs to each layer and reduce internal covariate shifts. This can help to stabilise the training process and reduce the likelihood of mode collapse or vanishing gradients, making it a good candidate for training virtual try-on systems. Nonetheless, the concept, and high-level architecture of DCGANs are similar to that of GANs.

Delving into the general architecture of GANs, the generator network typically consists of the following:

1. **Input layer**: The generator typically takes a random noise vector, known as the latent space, as input. This is usually sampled from a simple distribution, such as a Gaussian or uniform distribution. The dimensionality of the latent space is chosen depending on the desired complexity of the generated samples.
2. **Hidden layers**: The generator network typically consists of multiple hidden layers, which can be fully connected layers, convolutional layers, or a combination of both, depending on the type of data being generated. For example, when generating images, convolutional layers are commonly used as they are more suited to capturing spatial patterns, which is why DCGANs work well.
3. **Non-linear activation functions**: These are applied to the outputs of the hidden layers to introduce non-linearity in the generator network. Commonly used activation functions include Rectified Linear Unit (ReLU), Leaky ReLU, or tanh. These non-linearities allow the generator to model complex patterns in the data, which are common to images.
4. **Upsampling layers**: In the case of generating images, upsampling layers are often used to increase the spatial dimensions of the generated samples progressively. This is typically achieved using techniques like transposed convolutions or nearest-neighbour upsampling followed by convolutional layers. These layers help in creating a smooth transition from the low-dimensional latent space to the higher-dimensional output space, which is key in image synthesis.
5. **Output layer**: Lastly, the output layer produces the generated data samples, which are structured to match the format of the real data. For example, when generating images, the output layer produces a 3D tensor (a multi-dimensional array) representing the height, width, and colour channels of the image. The activation function in the output layer is chosen based on the nature of the data. For instance, the tanh activation function is often used for images with pixel values in the range $[-1, 1]$, while the sigmoid activation function is used for pixel values in the range $[0, 1]$.[1]

The generator network is trained to produce realistic samples by optimising its parameters through backpropagation. The discriminator network is similar to a standard binary classification network, consisting of one or more layers of convolutional neural networks, followed by a sigmoid function that outputs a probability between 0 and 1. The input to the discriminator network is a sample from the generator network or real data from the original dataset, and the output is a probability that the input is real or fake. The outputted probability represents

[1]These ranges are under the assumption that the pixel values, initially from 0 to 255, are normalised or scaled, respectively.

the chance that the input to the discriminator came from the actual data, as opposed to from the generator. The discriminator network is trained to maximise its ability to correctly classify real and generated samples. It is typically updated using backpropagation with a binary cross-entropy loss function, which penalises the discriminator for misclassifying samples.

Generative modelling constitutes an unsupervised learning challenge; however, a clever characteristic of the GAN architecture is that the training of the generative model is presented as two supervised learning subproblems. The two models, the generator and the discriminator, are trained concurrently, and adversarially. This means that the generator produces a batch of samples, which are then combined with real examples from the domain and provided to the discriminator for classification as either real or fake. The discriminator is updated to improve its ability to discern between real and fake samples. Crucially, the generator's updates are based on the degree to which its generated samples successfully deceived the discriminator. This adversarial training continues until an equilibrium is achieved, where neither the generator nor the discriminator improves on their performance.

The theory of adversarial training of GANs lies in Game Theory (von Neumann and Morgenstern, 1944), where the generator and discriminator are trained simultaneously by competing in a zero-sum minimax game (Goodfellow et al., 2014). Formally, we can model the objective function of this game with the value function $V(G, D)$ as:

$$\min_G \max_D V(G, D) = E_x[\log(D(x))] + E_z[\log(1 - D(G(z)))], \tag{2.1}$$

where x represents real data samples, z represents noise samples from the latent space, $G(z)$ represents the generated samples, and $D(x)$ and $D(G(z))$ represent the probabilities that the discriminator assigns to real and generated samples, respectively.[2] E_x represents the expected value of the discriminator for real samples, while E_z represents the expected value of the discriminator for generated samples. The two networks compete with opposing objectives:

1. The discriminator (D) tries to maximise the objective function $V(G, D)$. This means that D wants to correctly classify real samples, maximising $E_x[\log(D(x))]$, and identify fake samples, maximising $E_z[\log(1 - D(G(z)))]$.
2. The generator (G) tries to minimise the objective function $V(G, D)$, which is equivalent to minimising $E_z[\log(1 - D(G(z)))]$. In other words, G wants to produce samples that make $D(G(z))$ as close to 1 as possible, meaning the discriminator cannot differentiate between real and generated samples.

Because the training is modelled as a zero-sum game, when the discriminator correctly tells apart real and fake samples, it is rewarded and its model parameters are not updated, while the generator's parameters are adjusted. In contrast, if the generator tricks the discriminator, the generator's parameters remain unchanged, while the discriminator's parameters are updated due to its mistake in identifying the samples. The parameters are updated by using gradient-based optimisation methods, such as stochastic gradient descent or The Adaptive Moment Estimation (Adam) optimiser. The Adam optimiser was developed by Kingma and Ba (2014) and is a popular optimisation algorithm for stochastic gradient descent. This adversarial training happens until an equilibrium is reached.

Modelling the networks in this way helps us understand the dynamics of GAN training and the adversarial relationship between the generator and the discriminator. As the generator becomes better at creating realistic samples, the discriminator must also improve its ability to distinguish between real and fake samples, leading to a continuous process of co-adaptation between the two

[2]It must be noted that the notation used throughout the literature review is not necessarily the notation used in this research, however, each paper's notation is used for ease of reference.

networks. In summary, GANs involve a generator network that learns to generate new data, a discriminator network that learns to distinguish real and generated data, and adversarial training to optimise the networks.

GANs are atypical of traditional machine learning methods not only in their training but also in the fact that they generate data that cannot easily be assessed by traditional methods. This is particularly prevalent when using them to generate images as one cannot easily measure how "well" a GAN performs — even when the ground truth of an image is known, making a direct comparison between the real and generated images will be extremely computationally taxing as it will involve pixel-wise computations. An alternative to this is to use qualitative measures, however, this introduces subjectivity and potentially adds excess time to model training. To circumvent this, quantitative measures that assess image quality, diversity, and similarity of generated images to real images will be reviewed, which follows.

2.1.1 Evaluating generated images

Quantitatively assessing the quality and performance of generated images is crucial in the field of image synthesis. Various evaluation metrics have been developed to measure different aspects of image generation. In this subsection, several commonly used metrics are discussed, and their use is seen in the sections that follow.

Inception Score (IS), introduced by Salimans et al. (2016), is a metric that evaluates the diversity and quality of generated images. The intuition behind this metric is that a good generator should be able to generate images that are both diverse and resemble the distribution of the real data. It rewards generators that produce images that are diverse and contain a variety of different objects, while also being able to generate images that are similar to the real data. A higher IS indicates better quality and diversity. However, IS may not be well-suited for tasks like virtual try-on, as it emphasises sharpness and diversity, which may not align with the specific objectives of such applications.

A metric better suited to virtual try-on is perhaps the Fréchet Inception Distance (FID) score, proposed by Heusel et al. (2018), which measures the similarity between the distribution of generated and real images in feature space. A lower FID score indicates better similarity and quality. FID is commonly used for evaluating virtual try-on methods and provides insights into the performance of the model. However, this is somewhat of a subjective metric, as the range is from 0 to infinity, with lower scores indicating better quality. The metric may also be overly dependent on the dataset that is used (Choi et al., 2021).

Another metric that has application to virtual try-on is the Learned Perceptual Image Patch Similarity (LPIPS) score, introduced by Zhang et al. (2018), which measures perceptual similarity between images. It ranges from 0 to 1, with lower scores indicating higher perceptual similarity, and it is used to evaluate the resemblance between the original and synthesised images in paired experiments.

A method for assessing the quality and underlying structure of an image is Structural Similarity (SSIM), proposed by Wang et al. (2004), which measures the structural similarity between two images. It ranges from -1 to 1, with 1 indicating perfect structural similarity. SSIM is commonly employed to assess the structural quality and resemblance of generated images to the originals. Kernel Inception Distance (KID), introduced by Bińkowski et al. (2021), can also be used as it quantifies the similarity between the distributions of real and generated images. A lower KID score indicates a higher level of similarity between the distributions. KID is a useful metric for evaluating generative models, including image synthesis methods.

Lastly, Embedding Similarity (ES), proposed by Song et al. (2017), measures the quality of generated images by comparing the embeddings of different images that are used in generating

the final image. In a virtual try-on scenario, these could include images such as input garments, persons, and the try-on results. It assesses how well the model has transferred the desired features between the different images. ES has particular relevance in style transfer tasks. Whether a higher or lower ES value is more desirable depends on the specific application of the metric; however, in our case, a higher value is preferred as it indicates that the embeddings of the different images are more similar and as such, the features are bettered transferred between the images.

When evaluating image synthesis methods, a combination of these metrics should be used to obtain a comprehensive understanding of the generated image quality, diversity, structural similarity, perceptual similarity, distribution similarity, and feature transfer accuracy, all of which are desired in virtual try-on. GANs have become a popular and powerful technique for generative modelling, with many exciting theoretical and practical applications. One such area is virtual try-on, and the advancement of GANs have had a major part to play in that. To fully grasp what virtual try-on is and how it works, we explore it next.

2.2 Virtual try-on networks

Virtual try-on refers to the digital process that allows individuals to try on clothes or other fashion items such as makeup or glasses. Our primary focus, though, is on clothing, particularly tops. Essentially, it involves the digital superimposition of a selected item of clothing (referred to as the 'target garment') onto a different individual who was not originally wearing that garment (termed as the 'target person'). Han et al. (2017) highlight that although online shopping is increasingly easier and more convenient, consumers are still concerned with how a garment will look on them without first trying it on.

Numerous techniques leverage 3D measurements of body shape, either via depth-capturing cameras or inferring it using a 2D image using training data. These techniques are costly and time-consuming and are not feasible for large-scale implementation (Han et al., 2017). To overcome this issue, an image-based virtual try-on approach, which relies on plain RGB images without leveraging any 3D information is proposed. The goal of this approach is to synthesise photo-realistic images by placing an image of a target garment seamlessly onto the corresponding region of the target person while maintaining the characteristics of the target person.

According to Han et al. (2017), images generated for virtual try-on are expected to be perceptually convincing, and must meet the following criteria:

1. The body parts and pose of the target person must be preserved; this includes hair and any accessories.
2. The target garment should deform naturally according to the pose and body shape of the target person.
3. Patterns and colours of the target garment must be preserved; this includes textures, embroidery, and logos, amongst others.

To meet these criteria, Han et al. (2017) propose a Virtual Try-On Network (VITON); a coarse-to-fine framework that transfers the target garment onto the target person in the correct region from a 2D image. This framework consists of a multi-task encoder-decoder network which generates a coarse synthetic image of the target person wearing the target garment and a binary clothing mask[3] of the generated garment. This mask outlines the target garment shape in the correct region of the target person. As an example to illustrate what a mask looks like, consider

[3]With any mention of a mask, it is assumed that it is referring to a binary mask, consisting of pixels of a value that is either 0 or 1 (black or white, respectively).

(a) Target garment to mask. (b) Resulting binary mask.

Figure 2.1: Example of a binary mask generated from a garment.

Figure 2.1, where the garment in Figure 2.1a is used to create the mask shown in Figure 2.1b. The details of the garment are ignored, and the result is a black and white *mask* which contains the shape and outline of the garment.

Following the encoder-decoder is a refinement network; this network learns the optimal composition of the garment and applies it to the output of the encoder-decoder using the clothing mask. This refinement network adds details such as shadows, creases, patterns, and textures onto the target garment. These details are learned from both the target garment (patterns, textures, logos, etc.) and the target person (shadows, creases, etc.).

When given a reference image of a person, I, and an image of a garment, g, the goal of VITON is to synthesise a new image, $\hat{I}$, where the target person from image I is wearing the target garment g. Both input images, I and g, have a resolution of 768×1024 (width $\times$ height), while the synthesised image, $\hat{I}$, has a resolution of 256×192. The garment must be transferred as naturally as possible onto the corresponding body region of the target person, while also maintaining that person's pose and features. Han et al. (2017) highlight that the key to high-quality image synthesis is to learn a proper transformation from product images (garments) to clothes on any given body, and then to optimise this transformation. The authors also highlight the importance of sufficiently diverse data being used in training, as the model will need to generalise the synthesis of images for any arbitrary garment, including different styles, colours, patterns, etc., as well as for different body regions (torso, legs, etc.).

VITON, which synthesises new virtual try-on images given an image I and garment g, is broken into three main steps. The first is to create a clothing-agnostic person representation which contains a set of features, including pose, body parts, and face and hair. This is to constrain the synthesis process. The second step is where the image $\hat{I}$ is synthesised with an encoder-decoder architecture. This architecture is conditioned on the person representation from the first step, as well as the target garment g, and it results in a coarse image. Lastly, the coarse synthesised image is improved to account for visual details and deformations in the garment g. This is done with a refinement network. All of these phases are further explored next, starting with the clothing-agnostic person representation framework.

2.2.1 Clothing-agnostic person representation

A main technical challenge of a virtual try-on synthesis is to deform the target clothing image to fit the pose of a person, so to counter that, three clothing-agnostic representations of the target person are created. The first represents the pose of the target person, which uses parts affinity fields to estimate the pose (Cao et al., 2017). This pose estimator has been further refined and is now known as OpenPose (Cao et al., 2019). The pose of a person is represented as a set of

(a) Target person. (b) Pose key points.

Figure 2.2: OpenPose pose representation showing a target person and their pose key points.

18 coordinates, with each coordinate being transformed into an 11×11 heatmap. Doing this leverages the spatial layout of the key coordinates, and allows us to capture finer details about the target person's pose. Capturing the pose details is important because pose variations lead to different deformations in the garments, and it will affect the synthesised image $\hat{I}$. An example of this representation can be seen in Figure 2.2, which shows an image and its corresponding pose key points.

After the pose heatmap has been created, image segmentation is applied to the target person. This is done to segment different aspects of the target person's body, hair, and clothes, and is done using a human parser known as *Look into Person* (Gong et al., 2017). Doing this allows the authors to build a human body mask. This is done by transforming the segmented regions into a 1-channel binary mask, where 1 represents the body (excluding hair and face), and 0 otherwise. The mask is then downsampled to a lower resolution to avoid any artefacts where the target person's body and the target garment conflict. An example of this conflict can be seen in Figure 2.3, where the person's arm is covering a portion of the garment, which may result in poor segmentation, and subsequently, poor masking. Segmenting and building a mask of the target person's body are important as the appearance of a garment is dependent on the shape of the body. Han et al. (2017) state that learning how to transfer a garment depends on the target location of that garment and the target person's body shape. A trivial example to highlight this would be to consider the difference between transferring a garment, such as a blouse, and a pair of sunglasses. The blouse transfer needs to consider an array of complexities, such as torso width, height, shape, etc., while the sunglasses transfer needs to consider far less, as the difference in shape and proportion of a face is significantly less than that of a torso across a group of people.

Lastly, in the person representation phase, the *Look into Person* human parser is used again, but this time to extract the hair and face regions of the target person. Doing this ensures the identity and features of the target person, such as skin colour, hairstyle, facial features, etc., are maintained. These identity features are parsed to the model when generating new images, and they aid in ensuring a seamless virtual try-on while keeping the target person's identity.

All three outputs from the steps above: the pose heatmap, the human body mask, and the hair and face regions, are concatenated to form a clothing-agnostic person representation, denoted by p. Note that $p \in R^{m \times n \times k}$, where m and n are the respective height and width of the feature map in pixels, and k is the number of channels, or layers, of the feature map. Han et al. (2017) set it up so that $m = 256$, $n = 192$, and $k = 22$. This person representation, p, contains all the needed information regarding the target person. Convolutions are applied to model the relations between the different layers in p and this aids in creating a perceptually convincing

Figure 2.3: An example of body-garment conflict where a section of the target garment is obscured by the person's arm.

virtual try-on image. The next phase of VITON is the synthesis of the coarse image $\hat{I}$.

2.2.2 Coarse image synthesis

To synthesise a coarse (or unrefined) virtual try-on image, the authors propose using a multi-task encoder-decoder generator framework that takes the person representation, p, and the target garment, g, as input to synthesise this image. This is done through reconstruction of the person representation, p, so that the target garment, g, is naturally transferred to the corresponding region of the target person.

In brief, an encoder-decoder is similar to an autoencoder and it follows a nearly identical architecture. An autoencoder attempts to recreate the input data, while an encoder-decoder will attempt to generate a new representation of the input data. Encoder-decoders are commonly used in machine learning and deep learning for tasks such as image and speech recognition, natural language processing, machine translation, and in our case, image synthesis. The encoder-decoder architecture consists of two parts: the encoder and the decoder. The encoder takes input data and compresses it into a lower-dimensional representation, often referred to as encoding. The encoder typically consists of a series of layers that apply mathematical transformations to the input data, gradually reducing its dimensionality. The decoder then takes the compressed representation produced by the encoder and reconstructs the data to generate new data based on that representation. The decoder typically consists of a series of layers that reverse the transformations applied by the encoder, gradually increasing the dimensionality of the output (Sutskever et al., 2014). The encoder-decoder architecture is useful in this application because it enables the system to learn meaningful representations of the input data in a lower-dimensional space, making it easier to process and manipulate, and as such, synthesise images from it.

Multi-task encoder-decoder generators, as used by Han et al. (2017), merely combine the encoder-decoder architecture with the ability to perform multiple tasks simultaneously. This architecture works well when the tasks share an underlying structure, as is the case here. These multi-task encoder-decoders typically have one encoder with multiple decoders (one for each task), and these decoders share parameters to improve the efficiency and accuracy of both tasks. In our case, there is one encoder that encodes both the person representation, p, and the target garment, g; and there are two decoders, one to generate the image $\hat{I}$, and the other to generate the clothing mask, now denoted by $\hat{m}$. A multi-task encoder-decoder is chosen because in addition to guiding the network to focus on the correct body region in p, the predicted clothing mask, $\hat{m}$, will be further utilised to refine the generated result in the refinement network.

The encoder-decoder architecture used is a general form of the U-Net architecture, which is

characterised by its effective implementation of skip connections. These connections enable direct information transfer between the encoder and decoder parts of the network, enhancing the data assimilation beyond what is achieved through convolution and pooling alone. Developed by Ronneberger et al. (2015), the U-Net architecture is notable for its 'U'-shaped design. It consists of a contracting path to capture contextual details and a symmetric expanding path for precise localisation. This configuration excels in performing detailed, pixel-level segmentation, particularly valuable in applications requiring careful attention to small regions.

The encoder-decoder generator network can be represented by $G_C(g, p)$. This function takes two arguments: the first is the target garment, g, and the second is the person representation, p. The resulting approximation by $G_C(g, p)$ is a 4-channel output denoted as $(\hat{I}, \hat{m})$. $\hat{I}$ encompasses the first 3 channels, which are the RGB channels of the synthesised image $\hat{I}$. Typically, each pixel in a colour image is represented by channels for each of the colours: red, green, and blue. Each of the RGB channels is an $m \times n$ (height $\times$ width) matrix of pixels. So, the 3 channels here are merely the resulting synthesised image, $\hat{I}$, separated by the red, green, and blue colour channels of the image. The fourth channel is the generated garment mask, like the example shown in Figure 2.1b. Because the mask is a black and white, or greyscale, image, there is only one channel.

Now with the generator and output functions defined, Han et al. (2017) train G_C on paired data, where the input image and output image are the same — this is so that there is a ground truth to use. G_C synthesises an image $\hat{I}$ that is as close to I as possible and where $\hat{m}$ is as close to the pseudo ground truth mask, m_0, as possible. The authors find m_0 by parsing I to the *Look into Person* human parser; the mask, m_0, is the prediction of the human parser here for the target region — which is predicted based on the garment g parsed to G_C. In computer vision, pseudo ground truth is a technique used to generate an approximate, or *pseudo*, version of the ground truth annotations for a dataset, which is the garment in our case.

To train the encoder-decoder generator, such that $\hat{I} \simeq I$ and $\hat{m} \simeq m_0$, one needs to minimise a loss function. A simple way to do this would be to train the network with an L_1 loss, but Isola et al. (2018) showed that this resulted in blurry images when using colour (RGB) images, however, it did work well for greyscale images. So, to circumvent this issue for the RGB image, a perceptual loss that models the distance between the respective feature maps of images I and $\hat{I}$ is minimised. This perceptual loss is calculated by a visual perception network, as shown by Dosovitskiy and Brox (2016), Johnson et al. (2016), and Ledig et al. (2017). However, the L_1 loss function is still used to train the network in synthesising the clothing mask $\hat{m}$ as it is a greyscale image. Therefore, the entire encoder-decoder network can be trained with a combined loss function, where the perceptual loss between I and $\hat{I}$ is combined with the L_1 loss between $\hat{m}$ and m_0, as follows:

$$L_{G_C} = \alpha_I \left\| P_{I \to \hat{I}} \right\| + \alpha_m \left\| L_{1_{\hat{m} \to m_0}} \right\|. \tag{2.2}$$

In this equation, L_{G_C} represents the loss function to be minimised for the entire encoder-decoder network. $P_{I \to \hat{I}}$ represents the perceptual loss between images I and $\hat{I}$, and $L_{1_{\hat{m} \to m_0}}$ is the L_1 loss between the masks $\hat{m}$ and m_0. The hyperparameters α_I and α_m are the relative weight contributions of each of the separate loss functions. Having separate weights allows the modeller to choose the relative importance of each of the loss functions. By minimising this equation, the encoder-decoder network learns how to transfer a target garment onto a target person with minimal perceptual loss, and with as much detail in the garment as possible.

Han et al. (2017) note that although the generated image generally conforms to the desired pose, body parts, and identity of the original image, finer details of the target garment are missing. Details such as intricate patterns, text, and logos, among others, are often excluded by the model. They attribute this to the fact that generator frameworks are typically optimised

Figure 2.4: An example of hair-garment conflict where a section of the target garment is obscured by hair.

to synthesise globally perceptually convincing images based on the ground truth, as opposed to generating details. Essentially, what this means is that generators will typically focus on getting the main, and easily noticeable aspects of an image correct. Meaning the pose, body parts, personal identity, garment colour, and garment location are prioritised over details such as the garment texture, logos, and text. To overcome this, VITON uses a refinement network to improve the quality of the image, with a particular focus on the garment. This is the final phase of the VITON process.

2.2.3 Virtual try-on image refinement

The refinement network, denoted by G_R, is trained to render the blurry and coarse image $\hat{I}$ with realistic details in the garment. These details are learned from both the target garment and the target person. The refinement network has two stages: the first is where a warped clothing item is created, and the second is where a composition mask is created from the target garment.

To create the warped clothing item, Han et al. (2017) use information directly from the target garment g to fill in details on the coarse result in $\hat{I}$. A naive and simplistic approach to do this might be to *'copy and paste'* the target garment over the already existing garment to preserve all the details, however, clothes deform according to the pose and shape of the target person, so this will not work. A more appropriate approach is that of Belongie et al. (2002), which estimates a Thin Plate Spline (TPS) with shape context matching to add realistic shape and deformation to the garment. Specifically, VITON extracts the mask of g (as in Figure 2.1b) and computes shape context TPS warps between this mask and mask $\hat{m}$, which is generated by the multi-task encoder-decoder. These TPS parameters are used to transform the original garment g into a warped version, denoted by g'. This new warped garment, g', conforms to the target person's pose and body shape, and it keeps the detail of the target garment g.

The final stage of the refinement network learns how to composite the details of the target garment, g, with the warped garment and the occlusion details from the target person's pose. This composition should seamlessly combine g' with the targeted clothing region of $\hat{I}$, which will factor in occlusions such as arms across the body — such as in Figure 2.3 — and hair in front of the garment — as in Figure 2.4, where the model's hair falls over her right shoulder and arm. The synthesis of this warped composite garment, denoted by c', onto the coarse image of the target person in $\hat{I}$ results in a refined result of the virtual try-on, which is now denoted by I'.

This refinement network, G_R, like the encoder-decoder network has a concatenated input. The

coarse image $\hat{I}$ and the warped garment c' are the inputs to G_R, which are then converted to a 1-channel composition mask m'. This m' mask is then synthesised onto $\hat{I}$ to produce I'. Mathematically, this is represented by $G_R(\hat{I}, c') = I'$.

The composition mask, m', is a 1-channel matrix, where $m' \in [0,1]^{m \times n}$, with m and n being the height and width of the matrix. These dimensions are inherited from the original image I; in fact, those dimensions are carried throughout the entire network as the output image I' has the same size as I. $m' \in [0,1]^{m \times n}$ represents how much of the warped garment c' is in the final composite mask. As the mask is a composite, each pixel is a measure of how much information it inherits from both the warped garment c' and the coarse image $\hat{I}$ at the same pixel position. Essentially, each pixel in m' is a proportional sum of both c' and $\hat{I}$. We can determine the composition of I' with:

$$I' = m' \cdot c' + (1 - m') \cdot \hat{I}. \tag{2.3}$$

In this equation, $\cdot$ is the dot product, or matrix multiplication, of the respective matrices. In the first part of the equation, $m' \cdot c'$, is the proportion of the warped garment per pixel in the final image I', while $(1 - m') \cdot \hat{I}$ is the proportion of the coarse synthesised image $\hat{I}$ per pixel in I'. Both dot product operations result in an $m \times n$ matrix, which are then added together to form the final refined image I'. Similar to the encoder-decoder network, the refinement network is optimised to minimise perceptual loss between I and I', as well as L_1 loss between c' and m'. The overall loss function to be minimised is shown in Equation (2.4):

$$L_{G_R} = \alpha_{I'} \left\| P_{I \to I'} \right\| + \alpha_{m'} \left\| L_{1_{c' \to m'}} \right\|. \tag{2.4}$$

As before, there are two hyperparameters to be tuned, $\alpha_{I'}$ and $\alpha_{m'}$, which are the relative weights of the separate loss functions.

This concludes all three stages of the VITON of Han et al. (2017). In summary, the first stage generates the person representation of the target person, including pose, body representation, and face and hair details. The second stage synthesises a coarse image of the target person wearing the target garment and a clothing mask of the target garment. This is done using a multi-task encoder-decoder network trained to minimise both perceptual and L_1 loss. Lastly, the coarse synthesised image from stage two is refined to include as much detail from the target garment and deformations caused by pose and body shape as possible. This is done in a refinement network, which results in a perceptually convincing virtual try-on where the target person is wearing the target garment.

2.2.4 Implementation details and results

Now that the structure and methodologies of VITON are understood, the focus shifts to the implementation details and results. To train the model, and optimise its hyperparameters, adversarial training is used. Here, the generator network is the multi-task encoder-decoder generator network and the refinement network, while a binary classification network is used as the discriminator. This training approach follows that of training GANs, as explained by Goodfellow et al. (2014).

In their training setup, Han et al. (2017) leverage the encoder-decoder work of Radford et al. (2016b) and Jetchev and Bergmann (2017), which leads them to use the Adam optimiser with a fixed learning rate of $\alpha = 0.0002$, and exponential decay hyperparameters of $\beta_1 = 0.5$ and $\beta_2 = 0.999$. Adam typically includes adaptive learning rates, however, since the training data for virtual try-on is typically very large, using a fixed learning rate is helpful because the adaptive learning of Adam may not adapt quickly enough, and it will drastically slow the network down. It is also assumed that by using a fixed learning rate the training data will not change significantly

over time. The encoder-decoder network is trained for 15 000 steps, while the refinement network is trained for 6 000 steps. Both networks are trained with a batch size of 16, and the resolution of the images they are trained on is 256×192.

To understand how well VITON works, Han et al. (2017) use both qualitative and quantitative measures of performance. Qualitative measures here are merely a subjective level of how good the virtual try-on looks. Simply, they randomly sample results and analyse them visually and determine if they are a good fit or not. This qualitative measure is difficult to replicate, as people have different interpretations of what is good or not. While measuring model success or failure will not be the primary focus with qualitative metrics, they may serve to guide model enhancement. A possible resolution to this issue is to conduct a survey, where people are asked to rate the virtual try-on image. These ratings could then be used to assess the perceptual quality of the images synthesised by the model.

Quantitatively, Han et al. (2017) use IS to assess the quality of the generated images. It has particular relevance in GANs, but has been applied in similar settings as ours by Zhang et al. (2017), Odena et al. (2017), and Ma et al. (2018). As discussed in Subsection 2.1.1, the IS measures the diversity and quality of the generated images. A higher IS indicates better quality and diversity of the generated images. According to Han et al. (2017), they have an IS of 2.514 ± 0.130, which is compared to an IS of 3.312 ± 0.098 on real data.

The authors highlight that automatic metrics such as IS do not reward the task of virtual try-on, only the diversity and sharpness of the image are rewarded. Although this metric can be used as a gauge, they suggest not putting too much emphasis on it. The suggested route is conducting surveys with randomly sampled synthesised images to gain an understanding of how perceptually convincing they are to the human eye. The authors highlight that VITON generates more photo-realistic results than other state-of-the-art generative models, suggesting that the model is a success for virtual try-on.

VITON was the pioneering image-based virtual try-on method, and subsequent methods are all somewhat rooted in it. Based on their success, multiple authors have recreated and adapted, and in some cases, improved the results of Han et al. (2017). These include the following:

CP-VTON
: Developed by Wang et al. (2018a), Characteristic-Preserving Virtual Try-On Network (CP-VTON) improves on VITON by handling large spatial misalignment between input images and targets. Rather than using shape context matching to compute correspondences of interest points, CP-VTON uses a geometric matching module to learn the TPS to create the warped garment. CP-VTON produces slightly more photo-realistic results when compared to VITON.

VTNFP
: Virtual Try-On Network with Feature Preservation (VTNFP), developed by Yu et al. (2019b), has the same structure as VITON, however, the major differentiator of VTNFP is the body segmentation map prediction module. As opposed to using the *Look into Person* human parser to generate the body segmentation map, the authors create their own segmentation map generation module. This module is a multi-task encoder-decoder that takes the warped garment and the person representation as input to the network and generates a more realistic segmentation map. The improvement over *Look into Person* lies in that the custom segmentation map generation module is designed specifically for virtual try-on, as opposed to the more general application of *Look into Person*.

MG-VTON
: Multi-pose Guided Virtual Try-on Network (MG-VTON), the first major deviation from VITON, allows a user to not only virtually try-on garments, but they can also upload a

desired pose to try the garment on with. MG-VTON, developed by Dong et al. (2019), is also constructed in three stages: the first involves generating a segmentation map based on the target person, and the target pose. This is done with an encoder-decoder generator. Secondly, a deep Warping GAN warps the target garment to match the person's body shape in the desired pose. Lastly, the third stage is a refinement network like in VITON, which preserves and adds as much detail as possible. The synthesised images from MG-VTON achieve better results than VITON, but the comparison is not fair as they apply VITON to situations where the pose is different from the target person, thus meaning VITON was tested and compared against MG-VTON with a task that it was not designed for — pose generation. The authors do not explicitly test only the virtual try-on capabilities of MG-VTON, so no concrete conclusion can be drawn about its application to virtual try-on without pose differentiation.

ACGPN

To overcome the issues when there are large occlusions, either from the target person's garment (such as long sleeves, turtle neck, etc.) or from their pose (an arm across the body, hair over the garment, etc.), Yang et al. (2020) developed the Adaptive Content Generating and Preserving Network (ACGPN). This is a network that aims to predict what parts of the target person need to be synthesised or preserved, and this is done by predicting a semantic layout of the reference image that will be changed after the try-on. For example, if the target person is wearing a long-sleeved jacket, but wants to virtually try on a short-sleeved shirt, the semantic layout should predict that the target person's arms need to be synthesised, along with the target garment, while their remaining features will be preserved. To do this, a semantic layout generation module utilises semantic segmentation of the reference image to progressively predict the desired semantic layout after try-on. Following segmentation, ACGPN warps the target garment according to this semantic layout, after which an inpainting module for feature fusion integrates all the information to generate each part of the semantic layout. ACGPN produces impressive results, outperforming VITON, CP-VTON, and VTNFP in both IS and on human survey studies.

Although the above algorithms developed by the various authors make improvements on the work of Han et al. (2017), they are not substantial enough to warrant an in-depth study of the respective papers. The work of Choi et al. (2021) is the first major work after VITON that makes significant strides in terms of 2D photorealistic synthesised images for virtual try-on, and because of that, it will be reviewed in detail in the next section.

2.3 High-resolution virtual try-on networks

Although the work of Han et al. (2017), and the subsequent research based on their work, can create perceptually convincing virtual try-on images, Choi et al. (2021) highlight a major gap in the research — all the synthesised virtual try-on images have a low resolution. VITON, and the works using it mentioned above, all take high-resolution images (768×1024) and synthesise virtual try-on images with low resolution (256×192). This drawback negatively affects the virtual try-on performance as customers are expecting rendered images in the same resolution as the input images and images on typical online clothing store websites, which are 768×1024 (Choi et al., 2021). To this end, a high-resolution virtual try-on network called Virtual Try-On Network with High-definition (VITON-HD), which synthesises high-resolution images, is proposed.

To understand the impact of producing low-resolution images, consider Figure 2.5, which shows a comparison between a high-resolution image (Figure 2.5a) and a low-resolution image (Figure 2.5b). Both images are of the same model, wearing the same clothes on, however, one has

(a) High-resolution image. (b) Low-resolution image.

Figure 2.5: Comparison between high-resolution and low-resolution images.

a resolution of 768×1024, while the other has a lower resolution of 256×192. This drop in resolution is visually noticeable and detracts from the photorealistic nature that virtual try-on needs. Figure 2.5a is at the same resolution as the input images for VITON and all the works discussed above, while Figure 2.5b is at the resolution of the rendered virtual try-on images for these same methods. One can understand how the drop in quality, evident in Figure 2.5b, can affect customer experience as outlined by Choi et al. (2021). The drop in resolution between input and output images is what VITON-HD seeks to overcome by synthesising images with a resolution of 768×1024 as opposed to 256×192.

Choi et al. (2021) state that the rendered images from previous methods have a low resolution because of the following reasons: the first is that there is a misalignment between the warped garment and the target person's body, resulting in misalignment regions. The artefacts in this misalignment region become noticeable as the size of the image increases, so images are kept at a smaller size to reduce these artefacts becoming too noticeable. Secondly, the authors state that the U-Net architecture (Ronneberger et al., 2015), which is used by all the previous virtual try-on methods, is insufficient in synthesising occluded body parts in high resolution. Using a simple U-Net architecture for high-resolution image synthesis leads to unstable training and poor-quality generated images, while also only refining the images once at the pixel level is insufficient at preserving garment and person details (Wang et al., 2018b). VITON-HD handles both of these problems to produce high-resolution, photorealistic virtual try-on images.

Much like Han et al. (2017), Choi et al. (2021) highlight criteria that any virtual try-on system should meet at a minimum. They highlight four criteria, with the first three being the same as Han et al. (2017); the last, and new, criterion is that any body parts initially occluded by the person's clothes in the original image must be rendered properly. This requirement is important as it handles the first problem discussed earlier, where occlusions and misalignment artefacts are exposed in high-resolution images. To meet all the criteria, VITON-HD introduces and uses a new clothing-agnostic person representation that thoroughly eliminates clothing information and creates a segmentation map of the target person. This clothing-agnostic segmentation map is then parsed, along with the pose map and target garment to produce a new segmentation map that contains a clothing segment, along with previously occluded segments. As in CP-VTON, a geometric matching module is then used to warp the clothing garment to fit the target person, after which a novel Alignment-Aware Segment (ALIAS) normalisation method is applied to generate the virtual try-on images. Each of the steps is further explored next.

(a) Input image I. (b) Clothing-agnostic image I_a. (c) Segmentation map S_a.

Figure 2.6: Clothing agnostic person representations used by VITON-HD.

2.3.1 Clothing-agnostic person representation

To generate the new clothing-agnostic person representation, Choi et al. (2021) first identify areas of improvement in previous work. VITON, CP-VTON, and VTNFP all generate a coarse body shape mask to help synthesise the image, but all fail to accurately synthesise realistic body parts initially occluded, as well as detailed body parts such as the hands. To handle occlusions, ACGPN employs a detailed body mask, which is used as input for a neural network to predict what needs to be synthesised or not; this network attempts to discard as much of the clothing information that needs to be excluded from the final image. Discarding the clothing information entirely is important since if it is not done, misalignment occurs. However, neither the coarse body shape mask of VITON, CP-VTON, and VTNFP, nor the neural network of ACGPN can perfectly eliminate the clothing information. This is because the body shape mask used by all the prior methods includes the shape of the clothing. Refer back to Figure 2.1 to see how a mask can still provide information about the shape of a garment. VITON-HD seeks to create a completely clothing-agnostic person representation to reduce misalignment regions caused by residual clothing information.

To do this, a clothing-agnostic image, I_a, is produced from the input image I, along with a clothing-agnostic segmentation map, S_a. These can be seen in Figure 2.6. The input image used to create I_a and S_a is shown in Figure 2.6a, while the clothing-agnostic image I_a is shown in Figure 2.6b. The clothing-agnostic segmentation map S_a is shown in Figure 2.6c. We can see that there is no trace of the garment worn by the person in I in either I_a or S_a.

To create these clothing-agnostic person representations I_a and S_a, Choi et al. (2021) use OpenPose (Cao et al., 2019) to create a pose map P, and a Part Grouping Network (PGN) developed by Gong et al. (2018) to predict a segmentation map S. The predicted segmentation map, S, is used to remove the clothing region while preserving the rest of the image. The pose map, P, is used to remove the arms of the target person while keeping the hands, as seen in Figure 2.6c and Figure 2.6b. Now that the clothing information has thoroughly been removed from the person representation, the next step in the VITON-HD process is to create the new segmentation map.

2.3.2 Segmentation map and warped garment generation

This segmentation generator is an encoder-decoder network, with a U-Net architecture.[4] It takes three inputs, the clothing-agnostic segmentation map S_a, the pose map P, and the target

[4]The U-Net architecture can work well here as the segmentation map does not need to be a high-resolution image.

garment, g — this can be represented by $G_S(S_a, P, g)$. As an output, it predicts the segmentation map $\hat{S}$ of the target person wearing g. Mathematically, this relationship can be represented by $G_S(S_a, P, g) = \hat{S}$.

To train the segmentation generator network G_S, the optimal mapping between S and $\hat{S}$, where the original garment information is completely removed, is learnt. The loss function to be minimised in the training of the network is a sum of the conditional adversarial loss and pixel-wise cross-entropy loss between S and $\hat{S}$. Pixel-wise cross-entropy loss is a typical loss function used for semantic segmentation where each pixel of an image is labelled, and the model is trained to predict that label for each pixel. Semantic segmentation, in this case, is a multi-class classification problem as the segmentation map is made up of multiple regions, and cross-entropy is one of the most commonly used loss functions for multi-class classification tasks. This pixel-wise cross-entropy loss is represented by L_{CE}, while the conditional adversarial loss is represented by L_{cGAN}. The loss function, L_S, for the segmentation generation network, can be represented as the sum of L_{cGAN} and L_{CE}, as in Equation (2.5):

$$L_S = L_{cGAN} + \lambda_{CE} L_{CE}, \tag{2.5}$$

where the hyperparameter λ_{CE} corresponds to the relative importance between the two losses.

The next step in VITON-HD is to deform the target garment, g, to roughly align it with the corresponding region of the target person. As a first step in this process, a target garment mask is found by extracting the clothing region in the semantic segmentation map $\hat{S}$. This mask is denoted by $\hat{S}_g$. Following the work of Wang et al. (2018a), a geometric matching module to warp the target garment to match the shape and style of the target person is used. This geometric matching module takes the clothing-agnostic person representations I_a and P, along with the mask $\hat{S}_g$ and the target garment g as inputs into the model.

The model has two distinct steps: the first is finding a correlation matrix between the features extracted from I_a, P, and g, after which a neural network is used to predict the TPS transformation parameters. These parameters are represented by $\theta \in \mathbb{R}^{2\times5\times5}$. The next step is to warp g according to θ, and this is denoted by the function $\mathcal{W}(g, \theta) = c'$.

To train this module, the L_1 loss between c', which is the warped garment produced by $\mathcal{W}$, and the garment extracted from the input image I, which is now denoted as I_c, is found. Additionally, following the work of Yang et al. (2020) for ACGPN, a second-order difference constraint is introduced to reduce obvious distortions in c' introduced by $\mathcal{W}$. The overall loss function to be minimised for warping the garment can be seen in Equation (2.6):

$$L_{warp} = \left\| L_{1_{c' \to I_c}} \right\| + \lambda_{constr} L_{constr}. \tag{2.6}$$

Here, L_{constr} is the second-order difference constraint, and λ_{constr} is the hyperparameter to set the relative importance of L_{constr}.

Now with the segmentation map, which takes into account the clothing-agnostic person representation and the target garment, predicted, and the warped garment being generated, the final step of VITON-HD is to synthesise the final, high-resolution, virtual try-on image.

2.3.3 Try-on image synthesis

The aim here is to generate the final synthetic image, I', based on the outputs from the previous stages. Considering the overarching concept, the clothing-agnostic person representation (I_a, P) is fused with the warped garment c' and is guided by the segmentation map $\hat{S}$. To do this, the novel ALIAS conditional normalisation method is applied with the novel ALIAS generator. ALIAS normalisation allows for semantic information to be preserved, and for the removal of

(a) Mask $\hat{S}_g$. (b) Warped mask $M_{c'}$. (c) Misalignment mask M_{mis}.

Figure 2.7: Overview of misalignment mask generation.

misleading information from the misalignment regions. This is done by finding the masks of the misalignment regions, and leveraging them and $\hat{S}$ to preserve the semantic information and remove misalignment.

ALIAS normalisation has two inputs: the first is the synthetic segmentation map $\hat{S}$, and the second is the misalignment binary mask, denoted by M_{mis}. This misalignment mask highlights any areas that are not aligned between the clothing region of $\hat{S}$ and the mask generated by masking the warped garment. The mask generated from the warped garment is found in the same way as the other clothing masks are found, and it is denoted by $M_{c'}$. It first involves creating a mask from the target garment, denoted by M_c, and then parsing it to an adapted version of the warping function $\mathcal{W}$, which is now $\mathcal{W}(M_c, \theta) = M_{c'}$, where θ is the TPS transformation parameters. This function applies the transformation to the clothing mask, as opposed to the garment itself. The authors do not explicitly state why they choose to create the warped mask in this manner, but a simpler technique may be to merely apply a binary mask to the warped garment c' directly.

To find M_{mis}, we need to see what regions of $\hat{S}_g$ are excluded from $M_{c'}$. Choi et al. (2021) first find what regions of $\hat{S}_g$ and $M_{c'}$ intersect with each other, and call this M_{align}, this can be seen in Equation (2.7). They then subtract the intersecting regions from $\hat{S}_g$ to find the misalignment mask M_{mis}, as in Equation (2.8):

$$M_{align} = \hat{S}_g \cap \mathcal{W}(M_c, \theta), \tag{2.7}$$

$$M_{mis} = \hat{S}_g - M_{align}. \tag{2.8}$$

This is better understood through a set of images highlighting the process that happens to find the misalignment mask, which is shown in Figure 2.7. The clothing segment mask $\hat{S}_g$ can be seen in Figure 2.7a, while the mask of the warped garment $M_{c'}$ can be seen in Figure 2.7b. Finally, the resulting misalignment mask M_{mis} is seen in Figure 2.7c. This mask highlights the misalignment between the first two masks, and this is what the ALIAS normalisation handles.

The main idea behind this ALIAS normalisation is to remove any misleading information that is found in the misalignment regions. If there are any misaligned regions in the warped clothing mask $M_{c'}$, they are considered as part of the background, and as such, are excluded in the synthesis of the final image. Performing normalisation to misaligned regions allows for the removal of background information that causes artefacts in the final image, leading to more photorealistic results. ALIAS normalisation is applied in the generator to synthesise the final virtual try-on images. The details of each of the networks and models are discussed next.

2.3.4 Model details

The ALIAS generator is a CNN that takes two sets of inputs: the first is the concatenation of the segmentation map $\hat{S}$ and the misalignment mask M_{mis}, this is denoted by $(\hat{S}, M_{mis})$. Secondly, it also takes the concatenation of the clothing-agnostic image I_a, the pose map P, and the warped garment c' to form (I_a, P, c'). Both of the sets of inputs, $(\hat{S}, M_{mis})$ and (I_a, P, c'), are injected into every layer of the generator. The structure of the ALIAS generator is akin to that of a U-Net encoder-decoder, where the skip connections are replaced with the input data injection, and the encoder portion is removed, leaving only the decoder. This simplified approach can be followed as the work done by the encoder portion has already been carried out in the previous steps.

The generator is made up of a series of Residual Blocks (ResBlks) with upsampling layers. A ResBlk is composed of two or more convolutional layers, with a shortcut connection added to allow the output of one convolutional layer to be added to the output of another layer. ResBlks are useful in deep learning applications because they help to reduce the vanishing gradient problem and improve the flow of gradients during training. Additionally, they can help to improve the accuracy and stability of deep neural networks (He et al., 2015). Each ResBlk in the ALIAS generator is made up of three convolutional layers and three ALIAS normalisation layers. Due to the nature of the network, each ResBlk operates at a different resolution, so to counter that, the input injected into each layer is resized. This means that both $(\hat{S}, M_{mis})$ and (I_a, P, c') are resized according to the layer they are injected into. Before each ResBlk, the resized inputs are concatenated with the activation from the previous layer after passing through a convolutional layer. Each ResBlk refines this concatenated activation output, and by doing this, the network performs multi-scale refinement at a feature level as opposed to doing it once at a pixel level. Doing this preserves the clothing details significantly better than predecessors. Following Wang et al. (2018b) and Park et al. (2019), the ALIAS generator is trained with conditional adversarial loss, feature matching loss, and perceptual loss.

Choi et al. (2021) train both the segmentation generator and the geometric matching module at a resolution of 256×192, as opposed to 768×1024. They empirically found that both of these modules perform better when trained on that resolution, and there is a significant decrease in memory cost. In an inference phase after these steps, the output is upscaled to the 768×1024 resolution. This works successfully without detail loss as there are no personal identity or garment details that are involved with these modules, it is either a segmentation map, which resolution does not affect drastically, or it is the TPS parameters that are being upscaled.

The segmentation generator has a similar structure to U-Net, with four convolutional layers with the following number of respective output channels: 64, 128, 256, and 512. Each of these layers contains a 2×2 max pooling layer. There are four upsampling layers, with 2×2 upsampling in each layer. The respective number of output channels in these layers is 512, 256, 128, and 64. Lastly, there is a 3×3 convolutional layer with 13 output channels followed by the Softmax activation function. Based on the U-Net architecture, there are skip connections between every corresponding layer in the downsampling and upsampling networks.

The geometric matching module has two feature extractors and a neural network. Each feature extractor has six layers, all with a stride of 2 and a 4×4 filter; each layer has batch normalisation (apart from the last) and uses the ReLU activation function. The number of respective output channels in each of the feature extractor layers is 64, 128, 256, 512, 512, and 512. A correlation matrix is calculated from the two extractors, and the neural network predicts the TPS parameters from the correlation matrix. The neural network has four convolutional layers; the first two use a 4×4 filter, and a stride of 2, while the last two layers use a 3×3 filter. Each has batch normalisation and uses the ReLU activation function. The number of output channels in each layer is 512, 256, 128, and 64, respectively. These convolutional layers are followed by a

fully connected linear layer with a tanh activation function.

The ALIAS generator consists of a 3×3 convolutional layer with 1 024 output channels, followed by eight ALIAS ResBlks. The number of respective output channels from the eight ALIAS ResBlks is 1 024, 1 024, 1 024, 512, 256, 128, 64, and 32, respectively. The first seven use 2×2 upsampling. Lastly, there is a 3×3 convolutional layer with 3 output channels — these channels represent the RGB channels — which uses the tanh activation function. Following Miyato et al. (2018), spectral normalisation is applied to every layer.

The Adam optimiser is used for all three of the modules; both the segmentation generator and the geometric matching module use $\beta_1 = 0.5$ and $\beta_2 = 0.999$, while the ALIAS generator uses $\beta_1 = 0$ and $\beta_2 = 0.9$. The segmentation generator is trained for 200 000 iterations with a batch size of 8, while the geometric matching module is trained for 50 000 iterations, also with a batch size of 8. The ALIAS generator is trained for 200 000 iterations, but with a batch size of 4. With these details, the results that VITON-HD obtain are discussed next.

2.3.5 Results

As before, both qualitative and quantitative measures are used to understand the performance of VITON-HD. They compare their model's performance with both CP-VTON and ACGPN. Qualitatively, VITON-HD outperforms both as it creates more perceptually convincing images. The authors highlight that garment details are preserved, and they attribute that to the multi-scale refinement that happens at the feature level. They also note that VITON-HD synthesises body shape, and previously occluded body parts, well, regardless of what original garment is being worn by the target person. The effectiveness of ALIAS normalisation is investigated by comparing the results of VITON-HD with and without it. The authors prove qualitatively that ALIAS normalisation can fill the misaligned regions with the textures and patterns of the target garment by removing misleading background information, and artefacts that are introduced due to misalignment are kept to a minimum.

Quantitatively, the model is tested in both paired and unpaired experiments. In a paired setting, the target garment is the same as the original garment, and in an unpaired setting, the target garment is new. The unpaired setting is most akin to the real world, however, to gain a better understanding of model performance, pairing the data can work well, as we aim to reconstruct the original image. Again, VITON-HD is benchmarked against CP-VTON and ACGPN with three metrics. The metrics include SSIM, LPIPS, and FID; these metrics are the most common for virtual try-on according to the authors. Based on the results and notes of Han et al. (2017), the IS is not used as a metric as it cannot reflect whether clothing details have been preserved or not. The SSIM and LPIPS are used in the paired experiments, while FID is used for the unpaired experiment. Each metric is discussed in Subsection 2.1.1.

At a high resolution (768×1024), and in a paired setting, VITON-HD outperforms CP-VTON and ACGPN in all three metrics, and by significant margins too. These results can be seen in Table 2.1. VITON-HD shows a very good structural similarity between the original image and the synthesised image (which is the original image recreated, as it is a paired setting), as shown by the higher SSIM value of the model. The LPIPS value of VITON-HD is better than both ACGPN and CP-VTON, which shows it generates images with a higher degree of perceptual similarity. Lastly, the FID score shows that VITON-HD again far outperforms the other methods, which means it is capable of producing more diverse and realistic images. It is noted that the authors do not compare the FID scores to real images, as Han et al. (2017) do with the IS. This is a potential area of improvement to remove some level of subjectivity in judging model performance.

Based on the qualitative and quantitative measures, we can see that VITON-HD is superior to

Table 2.1: Quantitative performance metrics comparison between VITON-HD, ACGPN, and CP-VTON in a paired setting. ↑ and ↓ represent the desired result for each metric, with ↑ meaning higher results are better, while ↓ means lower results are better.

Model	SSIM ↑	LPIPS ↓	FID ↓
VITON-HD	0.895	0.053	11.74
ACGPN	0.856	0.102	43.39
CP-VTON	0.786	0.158	43.28

previous virtual try-on methods. It synthesises images at a high resolution of 768×1024, which means there is no resolution loss compared to the input images, therefore, the results are more photorealistic and appealing for a customer (Choi et al., 2021). The major improvement lies in removing the misalignment regions, and the distracting information in those regions which introduce artefacts into the synthesised image. Although VITON-HD is a major improvement in virtual try-on, the authors do suggest some improvements that can be made, and these are discussed next.

2.3.6 Improvements to high-resolution virtual try-on networks

High-resolution Virtual Try-On Network (HR-VITON), developed by Lee et al. (2022), is a high-resolution virtual try-on network with misalignment and occlusion-handled conditions. It improves on the existing work (specifically VITON-HD) by removing the misalignment between the warped garment and the segmentation mask. The authors point out that the existing methods warp the clothing item to fit the target person, and then generate the segmentation map of the person wearing the garment prior to fusing the two. This is where the misalignment between the two occurs; there is an information disconnection which leads to artefacts in the final image.

To this end, Lee et al. (2022) propose the novel HR-VITON framework which uses a unified module where the warping and segmentation stages are one stage with information exchange between them. HR-VITON also uses discriminator rejection which filters out all incorrect segmentation map predictions. As the architecture of HR-VITON is largely the same as VITON-HD, only the differences, and improvements, will be discussed here. This subsection will focus on the proposed unified module of warping and segmentation, as well as on discriminator rejection.

As a brief overview of the entire process of HR-VITON consider the following: the network contains two main stages: (1) a try-on condition generator; and (2) a try-on image generator. The try-on condition generator is the only difference between HR-VITON and VITON-HD. Nonetheless, given the clothing-agnostic person representation, and the target garment g, the try-on condition generator simultaneously deforms g to create the warped garment c' and creates the segmentation map $\hat{S}$. A discriminator rejector is then applied to filter out incorrect segmentation map predictions. Subsequently, the try-on image generator synthesises the final image using the output from the try-on condition generator.

The try-on condition generator has two encoders: a clothing encoder E_c, a segmentation encoder E_s; and a decoder. This generator has 4 inputs: the target garment g, the clothing mask $\hat{S}_g$, the clothing-agnostic segmentation map S_a, and the pose map P,[5] with g and $\hat{S}_g$ as inputs for E_c, and S_a and P as inputs for E_k. Given (g, $\hat{S}_g$) and (S_a, P), the feature pyramids $\{E_{c_k}\}_{k=0}^{4}$ and $\{E_{s_l}\}_{l=0}^{4}$ are extracted from each respective encoder. A feature pyramid is used to extract features from the input image at different levels of abstraction. These features are then combined in a top-down manner to create a pyramid of feature maps, which helps the

[5] As in VITON-HD, OpenPose and PGN are used to find S_a and P, respectively.

model to efficiently detect objects at different scales (Lin et al., 2017). In this case, each of the two feature pyramids has four layers. These extracted features are then fed into the Feature Fusion Blocks (FFBs) of the decoder, where the feature maps of the two encoders are fused to predict both the segmentation map and the warped garment simultaneously. An FFB is used to combine the high-level semantic features from the encoder with the low-level spatial features from the decoder and earlier layers in the network (Lin et al., 2017).

Each FFB is followed by two pathways: the *flow* pathway, and the *segmentation* pathway. The flow and segmentation pathways generate the appearance flow maps F_{f_i} and the segmentation feature maps F_{s_i}, respectively. These two pathways share information to estimate the appearance flow and segmentation map simultaneously. In the try-on condition generator of HR-VITON, there are four FFBs, with the outputs $F_{f_{i-1}}$ and $F_{s_{i-1}}$. F_{f_0} and F_{s_0} are the outputs of the respective encoders, E_c and E_s, and are the features parsed to the first FFB. As output of the FFBs, $F_{f_{i-1}}$ and $F_{s_{i-1}}$ are concatenated with E_{s_i} to create F_{s_i}, while $F_{f_{i-1}}$ deforms g to c'_i. The information exchange and concatenation are important in ensuring that the warped garment and the segmentation map align with each other. Finally, F_{f_4} and F_{s_4} are parsed through a condition-aligning block, which removes the non-overlapping regions between the two, resulting in the warped garment c', on which the warped garment mask $M_{c'}$ is found. The segmentation map $\hat{S}$ is found here too. This process is similar to the misalignment mask generation as in VITON-HD (shown in Figure 2.7), where the misalignment regions are removed, however, with HR-VITON, both the mask and the segmentation map have misalignment regions removed through the FFBs.

Now that the unified module for garment warping and segmentation map generation has been covered, and the fact that the try-on image generator is the same between HR-VITON and VITON-HD (explained in Section 2.3) we shift our attention to the discriminator rejection of HR-VITON. Razavi et al. (2019) introduced rejection sampling, which is based on the probability that a pre-trained classifier assigns to the correct class. Furthermore, Azadi et al. (2019) used this to develop discriminator rejection sampling, where a discriminator rejects the generated sample at test time. Lee et al. (2022) use discriminator rejection to filter out any low-quality generated segmentation maps by the try-on condition generator.

Azadi et al. (2019) state that the acceptance probability $p_{accept}(x)$ is defined as:

$$p_{accept}(x) = \frac{p_d(x)}{L \cdot p_g(x)}, \tag{2.9}$$

where p_d and p_g are the distribution and the implicit distribution of the generator, respectively, while L is a normalising constant, and x is the generated image. Lee et al. (2022) use the least-squares GAN loss, meaning the optimal discriminator can be defined as:

$$D^*(x) = \frac{p_d(x)}{p_d(x) + p_g(x)}. \tag{2.10}$$

This means that the acceptance probability from Equation (2.9) can now be represented using the optimal discriminator $D(x)$ as follows:

$$p_{accept} = \frac{D(x)}{L \cdot (1 - D(x))}. \tag{2.11}$$

This equality is only satisfied when $D = D^*$. The normalising constant, L, is defined as:

$$L = \max_x \frac{D(x)}{1 - D(x)}. \tag{2.12}$$

Table 2.2: Quantitative performance metrics comparison between HR-VITON, VITON-HD, ACGPN, and CP-VTON in a paired setting. The arrow notation is as before.[7]

Model	SSIM ↑	LPIPS ↓	FID ↓	KID ↓
HR-VITON	0.892	0.065	10.91	0.0018
VITON-HD	0.873	0.077	11.59	0.0025
ACGPN	0.850	0.112	43.39	0.0373
CP-VTON	0.786	0.158	43.28	0.0376

However, this is intractable,[6] so x is constructed from the segmentation map and input conditions (the pose map P, the garment g, the masked garment M_g, and the clothing-agnostic person representation S_a), and L is obtained using the entire training dataset, which is paired. Lee et al. (2022) accept x if p_{accept} is above a certain threshold between 0 and 1. The higher the threshold, the tighter the conditions for acceptance, and the more perceptually convincing the segmentation map will be, however, a balance between accuracy and computational power must be struck. Doing this allows HR-VITON to filter out any incorrect segmentation maps, improving the performance of the network.

The above highlights the differences between VITON-HD and HR-VITON, and how the latter improves on the former by unifying the try-on condition generation steps and using discriminator rejection sampling. The authors provide both quantitative and qualitative results to show HR-VITON's superiority when synthesising virtual try-on images in high resolution. HR-VITON is compared against VITON-HD, CP-VTON, and ACGPN, using SSIM, LPIPS, FID, and KID. Notably, Lee et al. (2022) add KID as a metric to understand the similarity between the distributions of generated and real images. The proposed method outperforms all the others on all the metrics, across multiple resolutions, however, the most noticeable improvement is seen at high resolution (768×1024). At this resolution, HR-VITON improves on VITON-HD (the next closest competition) in the following ways: a 2.2% higher SSIM score, a 15.6% lower LPIPS score, a 5.9% lower FID score, and a 27.5% lower KID score. These results can be seen in Table 2.2, which compare HR-VITON to VITON-HD, ACGPN, and CP-VTON in a paired setting.

The results showcase HR-VITON's ability to synthesise more perceptually convincing images for virtual try-on than other methods. The authors also conduct an ablation study to highlight the effectiveness of both the feature fusion blocks and the condition-aligning block in the unified try-on condition generator, and the discriminator rejection. The results show that HR-VITON is the best-performing virtual try-on network by a considerable margin at the 768×1024 resolution.

Although the above methods all demonstrate the ability to produce highly perceptually convincing virtual try-on images, they potentially lack real-world application. The reason for this is that all the methods investigated here require the target garment, g, to be in a separate image, and alone in the image too (as shown in Figure 2.1a). In the online retail space, this may not always be possible to achieve, as not all retailers have images of just the garment in the correct conditions, making these methods very subjective to the environment in which they operate. To overcome this gap, we investigate any methods that use target garments for virtual try-on that are not standalone images.

[6] In this context, 'intractable' refers to a problem that is too complex or computationally demanding to solve directly or straightforwardly with the given resources or methods, requiring the use of indirect or approximate techniques instead.

[7] The results for VITON-HD, ACGPN, and CP-VTON differ between Lee et al. (2022) and Choi et al. (2021); nonetheless, we report on the results as given in each paper.

(a) Target person (I_p) (b) Target garment (I_g) (c) Result (I_t)

Figure 2.8: Virtual try-on with garment extraction overview showing the target person, the target garment worn by a person, and the resulting synthesised image (Lewis et al., 2021).

2.4 Virtual try-on with garment extraction

Producing photorealistic virtual try-on images needs to be both effective and practical for the online retail industry. Currently, the methods that produce photorealistic results, such VITON-HD and HR-VITON, are not practical and robust enough for when the target garment is not standalone (i.e. in an image by itself). Having a model that can produce photorealistic virtual try-on images from a pair of images with both people and garments in them would be both practical and effective. There is far less literature regarding this approach to virtual try-on, and only two papers are reviewed here, which are the works by Lewis et al. (2021) and Ishikawa and Ikenaga (2022). Both of these papers successfully extract a garment from an image, however, they both produce images at a low resolution.

To better understand the difference between the two broad approaches to virtual try-on in terms of target garment access, consider Figure 2.8. In Figure 2.8a, we see the target person (I_p) — this is on whom we want to put the target garment (I_g), which is the blouse worn in Figure 2.8b. The resulting virtual try-on (I_t) is seen in Figure 2.8c.

There are two broad approaches to virtual try-on with garment extraction. The first is garment transfer where images are synthesised via layered interpolation, which refers to a technique used to generate intermediate images by blending multiple layers or levels of information. It involves smoothly transitioning between different layers or levels of a given image to create a new interpolation of the virtual try-on. The second approach first extracts the garment from an image, in a separate module, which is then refined and parsed to a traditional virtual try-on system to synthesise the try-on image with the extracted garment. Both approaches will be discussed next.

2.4.1 Layered interpolation

TryOnGan, developed by Lewis et al. (2021), uses a novel layered interpolation algorithm to blend the synthesis of a person and a garment. It is based on StyleGAN2 (Karras et al., 2020), which is a GAN focused on generating high-quality, photorealistic images, and it works in the following manner:

The generator network:

1. The generator starts with a mapping network that transforms input latent vectors (z) into intermediate latent vectors (w) in the W space. This disentangles the factors

of variation and enables better control over the image generation process.

2. The generator takes a small constant tensor (usually 4×4 or 8×8 spatial grid) as the initial input. A constant tensor is a multi-dimensional array (tensor) whose elements have a fixed value that does not change throughout the computation.
3. The generator architecture is designed to progressively add layers, starting with low-resolution images and gradually increasing the resolution as training progresses.
4. The intermediate latent vectors (w) are converted into style vectors (s) that are used to control the style at different resolutions in the generator's layers using adaptive instance normalisation layers.
5. Spatially varying noise is added to the generator's layers to introduce stochastic variation and improve image diversity.
6. Weight demodulation is introduced to improve the quality of generated images by better preserving fine details and resolving artefacts. This is done by normalising the weights in the convolutional layers of the generator. It involves dividing the weights of a convolutional layer by the per-feature-channel standard deviation of the weight tensor. This process adjusts the scale of the weight tensor, preventing the style modulation from over-amplifying certain features in the generated image, which could have led to artefacts.

The discriminator network:

1. Similar to the generator, the discriminator uses a progressively growing architecture, starting with high-resolution images and gradually decreasing the resolution as it processes the input.
2. The discriminator extracts features at different scales to capture both local and global information in the input image.
3. The discriminator outputs a single scalar value, indicating whether the input image is real or generated.

Both the generator and discriminator are trained using adversarial training, where the generator tries to create realistic images that the discriminator cannot differentiate from real images, while the discriminator attempts to correctly classify images as real or generated. This competition between the two networks leads to the generation of high-quality, photorealistic images. Lewis et al. (2021) adapt StyleGAN2 for virtual try-on tasks by training the model to generate images of people wearing different outfits by utilising the model's latent space to interpolate between images. This is done by training a pose-conditioned StyleGAN2 network that outputs RGB images and segmentations. A summary of the steps followed by the authors to do this are:

1. As a first step, *PoseNet* (Kendall et al., 2016) is used to generate a pose heatmap with 17 channels (based on 17 pose key points).
2. The StyleGAN2 architecture is modified to condition the generator on the pose information. This can be done by replacing the constant input tensor with an encoder that takes a pose representation, from the pose heatmap, as input. This helps disentangle the pose from other factors of variation and prevents undesired pose changes during the try-on process.
3. An auxiliary segmentation branch is added to the generator that outputs the garment and person segmentation masks alongside the RGB image. This helps in better disentanglement and allows for more accurate clothing transfer.

4. The model is trained on a prepared dataset where the pose heatmaps are generated separately with *PoseNet*.
5. To use the model for real images, the real images are projected onto the latent space of the trained StyleGAN2 model. This is achieved by optimising the perceptual loss (Zhang et al., 2018). In practice, this step entails projecting the target person image and the target garment image onto the latent space, and the optimal latent space interpolation between the two is found. This is defined as the process where, given an image of a person I_p in a specific outfit and an image of another person — or the same person — in a different garment I_g, the goal is to find an optimal interpolation between the latent representations of the two images. This allows for the transfer of the garment from I_g to I_p, while preserving the identity of I_p.
6. To do this, the trained StyleGAN2 generator is used to synthesise the virtual try-on result I_t, where the person p appears in the garment g.

In the context of TryOnGAN, the objective is to optimise the interpolation coefficients in the style blocks of StyleGAN2 to achieve the desired virtual try-on results. By fine-tuning these coefficients, the model can generate a photo-realistic synthesis of a person wearing a garment from a different image, effectively transferring the outfit while maintaining the person's appearance and pose. This optimisation process helps create realistic and accurate virtual try-on images. To summarise how Lewis et al. (2021) create an image-based virtual try-on network, interpolation coefficients between StyleGAN2's style blocks are optimised to merge style codes from two distinct images, enabling the semantic transfer of a region of interest from one image to another. This technique can be applied to images generated by StyleGAN2 or to real images after projecting them into the latent space.

The optimisation process is guided by three loss terms: editing-localisation loss, garment loss, and identity loss. The editing-localisation loss ensures that only the region of interest is edited, whilst the garment and identity losses ensure the preservation of the garment's details and the person's identity, respectively. The loss function to be optimised can be seen in Equation (2.13) below, with the alpha (α) values representing the relative weights of each of the individual loss functions.

$$L = \alpha_1 L_l + \alpha_2 L_g + \alpha_3 L_i. \tag{2.13}$$

The editing-localisation loss (L_l) term aims to ensure that the network only interpolates styles within the region of interest (i.e. the top). This is achieved by defining a term, M, that measures spatial overlap between the semantic regions in the image and the activation tensors. Instead of using *K-means* clustering for semantic cluster memberships as in Collins et al. (2020), the algorithm utilises segmentation outputs from the network. M is calculated per layer for both person and garment images, and high values represent channels corresponding to other segments. L_l is computed to encourage low interpolation coefficients for all other segments, focusing only on the segment of interest.

The garment loss (L_g) aims to transfer the correct shape and texture of the garment of interest by using embeddings to compute the perceptual distance between the garment areas in two images (Simonyan and Zisserman, 2015). Binary masks are created for the garment in both images (I_p and I_g), which are then applied to the RGB images by element-wise multiplication. The images are then blurred with a Gaussian filter and downscaled to 256×256 resolution. L_g is calculated as the perceptual distance between the two masked images using the weighted difference between the image features.

Finally, the identity loss term (L_i) helps the network preserve the person's identity by focusing on the hair and face regions of the images. It is calculated using segmentation labels from both

Table 2.3: Quantitative and qualitative performance metrics comparison between TryOnGAN, ACGPN, and CP-VTON in an unpaired setting. The arrow notation is as before.

Model	FID ↓	ES ↑	User study ↑
TryOnGAN	32.2	0.32	62.6%
ACGPN	66.8	0.22	31.3%
CP-VTON	87.0	0.27	6.1%

the person image (I_p) and the try-on image (I_t). The identity loss follows a similar process as the garment loss, measuring the perceptual distance between the masked images of the person's identity.

The method is fully automatic and eliminates the need for manual clustering. For real images, the algorithm first projects them into the extended latent space before applying Adam optimisation. The algorithm is more flexible than previous approaches, allowing for continuous query vectors that enable localised semantic edits (Lewis et al., 2021).

TryOnGAN was tested with both qualitative and quantitative metrics, and compared with CP-VTON and ACGPN. Notably, Lewis et al. (2021) do not test the models in a paired setting, but rather in an unpaired one. These results can be seen in Table 2.3, showing both the quantitative and qualitative metrics. Qualitatively, TryOnGAN outperforms both CP-VTON and ACGPN in a user study, with 62.6% of participants stating that it produces more photorealistic results than the other two methods. Quantitatively, the FID and ES scores are used, with each metric explained in Subsection 2.1.1. TryOnGAN outperforms the other two methods in both of the metrics, which highlights that it is capable of producing more photorealistic results when compared to other state-of-the-art methods.

Although TryOnGAN was compared with two other virtual try-on methods and outperformed them in all metrics, the validity of the comparison is potentially biased. The tests used two input images containing people, requiring CP-VTON and ACGPN to extract garments, a task they were not designed for. This approach disproportionately disadvantages the non-garment extracting models. A more equitable comparison might involve using a standalone garment image as the input for the CP-VTON and ACGPN models, and a garment-on-person image for the TryOnGAN model. When applying each method to the task they are designed for, we get the following results: CP-VTON has an FID score of 43.28, while ACGPN has a score of 43.39 (ES was not measured previously) (Choi et al., 2021). We can see that TryOnGAN still outperforms those two methods on the FID metric, however, it is not as significant as before. Nonetheless, it is substantially outperformed by the high-resolution VITON-HD and HR-VITON networks. VITON-HD has an FID score of 11.59, while HR-VITON has a score of 10.91 (Lee et al., 2022). This suggests that there are methods that produce more photorealistic, high-resolution virtual try-on images; however, because TryOnGAN can extract garments from preexisting images, which the other methods cannot do, it may have a more practical application. Next, we focus on the second approach to virtual try-on with garment extraction, which first extracts the garment in a separate module.

2.4.2 Standalone garment extraction

Ishikawa and Ikenaga (2022) develop an image-based virtual try-on system with a garment extraction module that can adapt to any posture. The authors point out that previous methods (apart from TryOnGAN) are only compatible with pre-prepared clothing models (i.e. when the garment is in a standalone image), and they are not useful on unprepared clothing models. This claim is supported by the performance of the models compared to TryOnGAN by Lewis et al. (2021). To this end, Ishikawa and Ikenaga (2022) develop a virtual try-on system that

automatically creates a clothing model from an image without the need to have a standalone garment image. The entire model is broken into three modules: the first is the clothing extraction module, followed by a geometric matching model, after which the try-on module synthesises the virtual try-on image. Both the geometric matching module and the try-on module use the same architecture as CP-VTON (Wang et al., 2018a), which has been proven to be less effective than both VITON-HD and HR-VITON (Choi et al., 2021; Lee et al., 2022). It is for this reason that only the garment extraction module is investigated and considered.

The garment extraction module is based on Self-Correction for Human Parsing (SHCP), which is a person segmentation method that labels each semantic part (Li et al., 2022). The module is centred around extracting the target garment (in our case, the top) from the semantic segmentation performed by SHCP. Once the garment extraction module is provided with an image of a person wearing the target garment, SHCP is applied for semantic segmentation. The target garment is output in blue by SHCP, so a clothing mask is created by extracting only blue $(0, 0, 128)$ from the segmentation. The actual garment is then extracted by performing mask processing based on the input image and the generated clothing mask. Because the generated mask is a binary mask, with 1 indicating the garment, and 0 otherwise, simple matrix multiplication can be applied to the input image to keep only those pixels that comprise the target garment. The final proposed model filters the pixels not containing the target garment (which will be black, or $(0, 0, 0)$) to be white $(255, 255, 255)$. The output is then resized to match the required resolution for input into CP-VTON.

Ishikawa and Ikenaga (2022) state that when garment information is lost, such as when a portion of the garment is covered by an arm, or hair, for example, it is problematic as supplementing that information proves difficult. An example of this loss can be seen in Figure 2.3 and Figure 2.4 in Section 2.2, where the woman's arm and hair leave parts of the garment occluded, respectively. They, therefore, apply the following restrictions to their method: the target model is front facing, and no arms or hair overlap with the clothing. These restrictions may reduce the practical ability of this method, so additional methods to perhaps overcome this issue are discussed briefly below:

1. **Context-aware inpainting**: Deep learning-based inpainting techniques, such as GANs or variational autoencoders, can be used to fill in missing or occluded garment regions based on the surrounding context. These techniques learn to generate plausible content by training on a large dataset of clothing images. These models can be fine-tuned on specific garment datasets to get better results (Yu et al., 2018).

2. **Use of symmetry**: Exploit the inherent symmetry of many garments to predict the appearance of occluded parts. If one side of a garment is visible and the other side is occluded, the occluded part can be estimated by mirroring the visible part (Huang et al., 2005).

3. **Pose-guided synthesis**: One can use information about the human body's pose to guide the synthesis of the missing garment parts. There are deep learning models, such as Pose-Transfer GANs, that can transfer the appearance of a garment from one pose to another. These models can be used to generate the missing parts based on the pose information (Ma et al., 2018). This is used by Lewis et al. (2021) in TryOnGAN.

4. **Transfer learning**: Train a deep learning model on a large dataset of various garment types and use transfer learning to fine-tune the model on the specific dataset. In this way, the model will learn to generate plausible garment parts based on the features learned from the larger dataset (Yosinski et al., 2014).

Using some of these methods[8] in conjunction with the architecture of Ishikawa and Ikenaga

[8]This research does not explore all methods due to resource constraints, but investigating the impact of the

(2022) for garment extraction, one could potentially generate high-quality and high-resolution virtual try-on images by leveraging later and more improved methods such as HR-VITON for the try-on module. Doing this will not only leverage the high-resolution and photorealistic image synthesis capabilities of HR-VITON but will also allow the virtual try-on system to be more practical across the online retail space by allowing users to virtually try-on garments worn by other people, not needing garments in a standalone image. However, we must further consider the crucial process that fills in the visual gap left in the image when the original garment is virtually removed. A visual gap is left in the image when there is either a body-garment or hair-garment conflict in the garment that is to be extracted for the virtual try-on. In the next section, we delve into the specifics of this process and illustrate how it forms an integral component of the virtual try-on process, particularly when using garment extraction.

2.4.3 Image inpainting

In the context of virtual try-on, image inpainting is leveraged to fill in the regions of the image where the garment has been obscured in some way, typically by hair, another garment, or a body part. Doing this results in a more realistic depiction of the extracted garment — one that is akin to a standalone garment image. This can further add to the practicality that garment extraction virtual try-on methods have compared to more traditional approaches, such as HR-VITON. Adding a context-aware inpainting module to the garment extraction network of Ishikawa and Ikenaga (2022) could overcome the shortfalls the authors point out, particularly around information loss.

Image inpainting is a technique that uses the context of the surrounding pixels to fill in missing or corrupted parts of an image. The aim is to make the filled-in part look as natural and seamless as possible. Deep learning-based inpainting methods, often using GANs, have shown to be particularly effective.

In the context-aware approach to image inpainting, the algorithm understands the semantics of the image content, making the filled parts blend better with the surrounding area. This means it will be aware of textures, structures, and other visual features in the rest of the image, and try to replicate these when filling in the blank spaces. Context-aware methods tend to produce more coherent and visually pleasing results than traditional inpainting methods (Liu et al., 2018).

Image inpainting has had success in other areas, such as object removal, regional editing, super-resolution, stitching, and others (Kınlı et al., 2020). Although image inpainting is not a novel method, there have been very few applications of it in virtual try-on with garment extraction. Previous work using inpainting in virtual try-on focuses on filling in the gaps in the final synthesised image, as opposed to refining the extracted garment image (Kubo et al., 2019). We aim to leverage the context-aware inpainting capabilities of previously developed methods to create a standalone extracted garment image as close to the ground truth as possible.

The work of Yi et al. (2020) will be studied as a model to use for context-aware inpainting. The details of their method are discussed in Section 4.2.2, however, a summary of the context-aware approach they use for inpainting images at high resolutions is discussed here. The authors do not explicitly apply their technique to a garment image, however, it has application for several image types, including garments.

An improvement on existing work, the work of Yi et al. (2020) uses Contextual Residual Aggregation (CRA) for ultra-high-resolution image inpainting. Prior work, such as Kınlı et al. (2020), Liu et al. (2018), and Yu et al. (2018) produce good results, but at resolutions lower than our desired 768×1024. Yi et al. (2020) train the model in such a way that it can handle images

unexplored methods on the final virtual try-on result could be beneficial.

(a) Original image. (b) Image with mask. (c) Result after inpainting.

Figure 2.9: Context-aware image inpainting overview showing the original image to be inpainted, highlighting the region to inpaint, along with the binary mask to direct the model to the desired region. The result is also shown showing the desired region inpainted.

with up to 8K resolution, outperforming previous methods by some degree. To achieve this, they propose the novel CRA mechanism whose pipeline includes a generator, which is the only component that undergoes training. It processes high-resolution images, first by down-sampling them to 512×512, followed by an up-sampling procedure to obtain a large, blurry image of the same size as the original input. The generator then takes this low-resolution image and fills in any existing holes. This process is complemented by the calculation of attention scores by the generator's Attention Computing Module (ACM). Another crucial operation performed by the generator is the computation of contextual residuals, which are found by subtracting the large blurry image from the raw input. These residuals are further aggregated in the mask region through a process that combines them with the attention scores via the Attention Transfer Module (ATM).

The final image is obtained by adding these aggregated residuals to the up-sampled inpainted result. This procedure results in a sharp output in the mask region. Outside the mask region, the final output is identical to the raw input. To further illustrate how inpainting works, consider the example shown in Figure 2.9. Here, we can see that the original image is of a woman wearing a white shirt with the word "POLO" written across the chest. For this example, we are aiming to apply inpainting to the word to remove it from the shirt. To do this, a binary mask of that region needs to be found — this is the mask that directs the algorithm as to what pixels to inpaint, and is shown in the middle image.[9] After inpainting, the result is shown in the last image; here, the pixels that were not considered for inpainting remain exactly as they were in the original image, while the masked pixels have been inpainted to produce a coherent and realistic result with the word "POLO" removed.

In this example, inpainting was used for image manipulation, and this is but one example of how inpainting can be applied to fashion images. Other uses of inpainting could be for artefact removal (such as a blemish on the clothing item or anywhere else in the image), and for image correction. The latter is of particular interest to us as when extracting garments, defects and occlusions are bound to occur, and inpainting could be a way to circumvent that. The intended use of inpainting for this research is to apply inpainting for garment correction after the garment is extracted from an image. To the author's knowledge, no prior virtual try-on work has applied inpainting in this manner — it is only Kubo et al. (2019) that uses inpainting for correction and artefact removal in the final synthesised image.

[9]Note that the binary mask is inversed here when compared with the previous ones shown in the report; this is because the model needs to ignore those pixels, even though they are the area of interest.

Using context-aware inpainting will allow for more realistic garment images to be parsed to the try-on modules, achieving more overall photorealism. Getting the extracted garment image as close to the ground truth image as possible will allow similarly impressive and photorealistic try-on images to be synthesised as in HR-VITON, with the added practicality of the work of Ishikawa and Ikenaga (2022). The models built for this research attempt to do that. This brings an end to the literature review on virtual try-on systems, with concluding remarks about the literature to follow.

2.5 Conclusion

Based on the literature, one can see that virtual try-on systems have the potential to transform the online fashion retail space. With the advent, and subsequent improvement of GANs, virtual try-on networks can produce high-resolution photorealistic images where a person can "try on" a chosen garment, not limited to style, shape, or colour. VITONs were introduced by Han et al. (2017), however, their work has been improved on by numerous authors, with the most photorealistic results being produced by Choi et al. (2021), Lee et al. (2022), and Lewis et al. (2021).

In summary, any virtual try-on system must meet the following criteria: the person's identity needs to be preserved; the garment must deform naturally according to the person's body; the style and shape of the garment need to be preserved; and lastly, the virtual try-on system should be practical in terms of its implementation. HR-VITON meets the first three criteria, while potentially lacking practicality in its application, however; TryOnGAN has immense practical implementation ability, but does not produce results as photorealistic as HR-VITON. Going forward, the high-resolution photorealistic capabilities of HR-VITON should be combined with the garment extraction ability and practicality of the work of Ishikawa and Ikenaga (2022), which should be aided by context-aware inpainting.

An important consideration that Han et al. (2017) mention, which is carried throughout the literature, is that any proposed virtual try-on method should be trained on data that is similar to what will be seen in testing and production. Ideally, a model for virtual try-on will be trained on data where the target person is wearing the target garment in different poses, and where the same person is wearing many different garments, however, in reality, this will seldom be the case. It is for this reason that the model is trained on data that is similar to what will be available in reality, and a more generalised approach to virtual try-on is developed. The data used is discussed in the chapter that follows.

3 Data

In machine learning, high-quality, relevant, and diverse data are crucial for training and building effective models. If the data are poor or limited, the resulting model is likely to underperform or fail to generalise well to new situations. In this chapter, we delve into the heart of this study by examining the dataset that forms the foundation of the research on virtual try-on. The focus will be on paired images of women[10] and clothing items and the data will be described before conducting exploratory data analysis (EDA) to gain valuable insights. By the end of this chapter, the data that drive virtual try-on will be understood, enabling virtual try-on models to be built using it.

The dataset used for this study is introduced by Choi et al. (2021), with the same data being used by Lee et al. (2022) too.[11] It is a dataset of 13 679 high-resolution (768×1024) frontal-view woman and top clothing image pairs, from the Zalando online retail store. The dataset has already been split into a training set with 11 647 image pairs, and a testing set with the remaining 2 032 image pairs. As previously mentioned, the dataset is paired, insofar as the top worn by the woman in the image is also present in a separate image by itself. An example of this can be seen in Figure 3.1, where the garment worn by the woman in Figure 3.1a is in a separate, but paired image, as shown in Figure 3.1b.

Having the dataset structured like this is ideal for training virtual try-on systems, as it allows the models to be trained with the ground truth, assuming a paired training setup. As stated by Choi et al. (2021), the pairs of images featuring an individual and a garment are utilised to assess a paired configuration, while garment images are randomised for an unpaired configuration. The objective of the paired configuration is to reconstruct the image of the individual while maintaining the original clothing item, whereas the unpaired configuration aims to alter the clothing item on the individual's image by replacing it with a different item. The paired configuration is akin to training and testing on known data, while the unpaired configuration is akin to testing on unknown data (i.e. there is no ground truth, and one cannot make a complete assessment of the model quality). Now that we understand the structure of the data, and what it will eventually be used for, we explore it further in the next section.

3.1 Exploratory data analysis

Data are the backbone of any machine learning model, however, to create a model that truly captures and models the essence of the problem, the data that the model is built on should be high-quality, diverse, and representative of the system to be modelled. In our case, the data need to be of a high enough resolution to produce perceptually convincing and photorealistic

[10]The words 'women' and 'person' are used interchangeably, with both referring to the images containing people. Based on the wording used by Choi et al. (2021), the dataset is only considered to contain images of women.

[11]The authors state that they scraped the Zalando website, a publically traded German online retailer, active across Europe, and have made the dataset publically available here.

(a) Frontal-view image. (b) Paired garment.

Figure 3.1: Paired data overview showing a person wearing a garment and that same garment in a standalone image.

results, while also being diverse enough to cover an array of garment styles, patterns, textures, and colours; and being robust enough to handle different skin tones, hair colours, poses, and body shapes. This is important as any virtual try-on system will be personal to the user who wants to try the clothes on; this being said, the developed model needs to factor in differences in human appearance and be able to handle any style and type of garment. To understand how well the data used to build the models captures the essence of the problem, EDA is performed.

Image data, unlike tabular or textual data, is composed of a grid of pixels, each with its colour intensity values. These values are typically represented as integers, with each pixel containing red, green, and blue channels (RGB) or, in some cases, additional channels such as alpha for transparency. Consequently, the data points within an image are highly interconnected, and the spatial relationships between pixels play a vital role in understanding the content and features of the image. Performing EDA on image data requires a different approach due to the inherent complexity of the data structure. While traditional data often relies on summary statistics, correlations, and distributions, image data demands more specialised techniques for analysis.[12] The primary goal of EDA for image data is to gain insights into the visual patterns, relationships, and potential anomalies present in the dataset. The EDA performed here will help us understand the nuances of the images, and whether or not effective virtual try-on systems can be developed with it. In summary, the following will be investigated:

1. **Clothing colour analysis:** Analyse the colours of the clothing items.
2. **Clothing type analysis:** Examine the different types of clothing items worn by the women in the images.
3. **Skin colour analysis:** Understand the distribution of skin colours among the people in the images.
4. **Hair colour analysis:** Analyse the hair colour distribution among the people.
5. **Pose analysis:** Inspect the distribution of poses in the images.
6. **Image quality analysis:** Assess the quality of the images in terms of resolution, lighting conditions, blurriness, colour, and general structural quality.

Various techniques will be used for the above. Visualisations play a key role in understanding complex data, so they will be used frequently throughout this chapter. We start our EDA by

[12]Some techniques used here for the analysis are not explained technically, such as image segmentation, garment masking, pose analysis, etc. as they are used in formulating the model, and they will be discussed in depth in the next chapter.

analysing the garment colours across the different images.

3.1.1 Clothing colour analysis

In the context of training machine learning models for virtual try-on applications, diverse garment colours are pivotal in fostering robust and accurate models by ensuring effective generalisation, mitigating biases, enhancing feature learning, addressing colour-based challenges, and promoting adaptability to new clothing items and styles. Consequently, incorporating a wide range of garment colours in the training dataset is essential to achieve a reliable and unbiased virtual try-on experience that remains relevant as fashion trends evolve.

To accomplish this, we use the garment images and remove the background of the image to analyse the garment colour without interference from the background. The images all have a similar colour for the background, so a binary mask is created by identifying pixels within a specified colour range (white-type background in this case). The mask is then inverted and applied to the original image using a bitwise operation, resulting in an image with only the garment and a black background. After this, the dominant colour of the garment is extracted using the *K-means* clustering algorithm. The image is first reshaped into an array of pixels, and black pixels from the masked background are removed. Then, the *K-means* clustering algorithm is applied to the remaining pixels, with 3 clusters — one for each RGB channel.

Subsequently, the cluster with the most pixels is considered the dominant one, and the average colour of those pixels is deemed as the dominant colour for that image. Once all the images are processed, and the average garment colour is found per image, we again apply the *K-means* algorithm to group the colours based on similarity. This similarity is based on the RGB values of the colours and their brightness. It was decided to not follow the conventional *K-means* clustering method where the optimal number of clusters is found first, and then the clustering is applied. K is predetermined, and the garments are grouped into 40 different clusters, for simplicity's sake. The value of 40 is somewhat arbitrary, as the number of clusters here is not what is important, what is important is whether there is a fair distribution of colour in the garments. The 40 clusters merely represent 40 different colours that the garments are grouped into; this is sufficient to understand the distribution of the diversity of garment colours. The average colour per group is returned as the value for that group, and the distribution of the colours across the 11 647 images in the training set[13] can be seen in Figure 3.2.

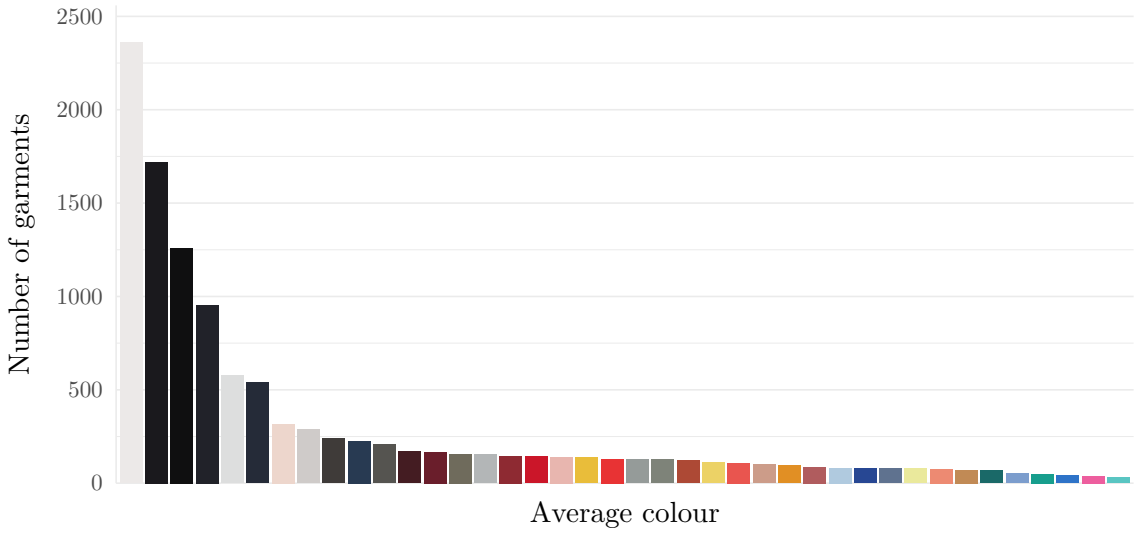

Figure 3.2: Distribution of average training set garment colours.

[13]Only the training set is used throughout this chapter so as not to introduce any information leakage.

This figure shows the number of garments in each of the 40 different colour groups, and they are arranged in decreasing order by count. The outcome of this is somewhat expected, with the lighter colours (closer to white and cream) and much darker colours (close to black and navy blue) dominating the dataset, with more garments being darker. These colours can be classified as neutral. The distribution of colours seemingly follows the Pareto principle, where approximately 80% of the garments come from 20% of the colours. Apart from that, there is a reasonable spread of colours across the data, with an array of colours, both bright and dull, extracted from the garments. There is a fair distribution of reds, blues, and greens, and the subsequent secondary and tertiary colours too. Both warm and cool colours are well represented across the data, as well as neutral colours. The data used to build this figure can be found in Table A.1, in Appendix A.[14]

Overall, it is deemed that there is sufficient diversity in the colours of the garments across the data, which will bode well in building a robust virtual try-on system. The colours in the dataset accurately capture the essence of the problem we are trying to model. The next characteristic of the garments explored are their different types.

3.1.2 Clothing type analysis

Once again, to build a robust model that has no bias toward a particular style of clothing, we must ensure our dataset contains enough diversity in garment style. Because we are dealing with only tops, introducing diversity here is somewhat more complicated as typical fashion datasets are across various garment types, such as pants, shoes, tops, etc. Here, diversity refers to the garment size (length and width), shape, neckline, sleeve length, and patterns. The garment colour is purposefully removed from the analysis to not introduce 'diversity' which has already been captured in the previous analysis.

To accomplish this, we use only the garment dataset where the garments are first converted to greyscale, thereby removing the colour, but not other important features. Next, features from the garments are extracted using the VGG16 architecture (Simonyan and Zisserman, 2015). The VGG16 architecture is well-suited for feature extraction due to its deep and complex convolutional layers, which can learn rich and detailed representations of the input images. This pre-trained VGG16 model extracts 4 096 features from the images, which are then parsed to a *K-means* clustering algorithm.

Because we are dealing with unlabelled data, we have to find an appropriate number of clusters to group the data into. To do this, we iterate from 1 cluster up to 30 clusters — this essentially means that we test to see how many different garment groups there are in the data. For each of those iterations with a different number of clusters, we calculate the distortion value.[15] The distortion in *K-means* refers to the sum of the squared distances between each data point and its assigned cluster centre. We then plot these values against the number of clusters in what is known as an elbow plot, which can be seen in Figure 3.3.

What we are looking for in this plot is an 'elbow' in the line. This is not a perfect method, as it introduces some subjectivity, however, it is the convention. An 'elbow' is the point where the graph starts to level off, indicating that adding more clusters does not significantly improve the clustering performance. In this plot, we can see this occurring at approximately 5 clusters, indicating that there are 5 main 'groups', or 'types' of garments in our dataset.

Using this information, we decided to group the garments into 5 main clusters. Doing this essentially means we have 5 main categories of garments in the data. It must be noted that

[14]Here, the actual colours are important as they represent the true colours in the dataset, so a colourblind friendly palette cannot be used, hence the inclusions of the tabular data to be as accommodating as possible.

[15]This is sometimes referred to as the inertia or sum of squared errors.

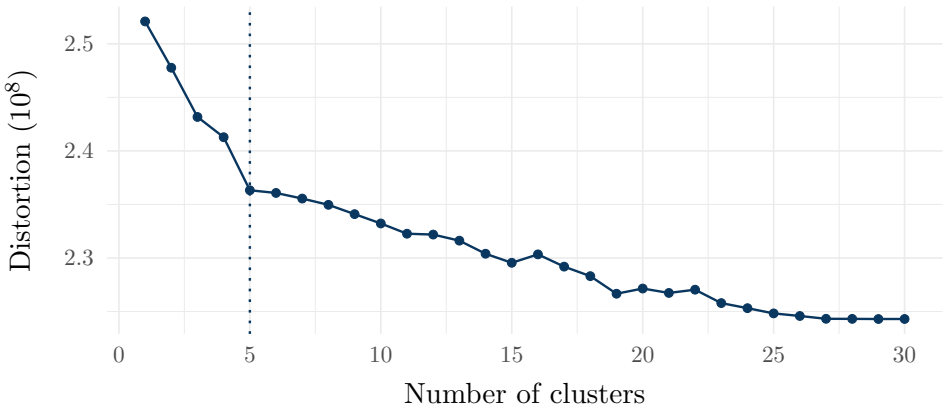

Figure 3.3: Elbow plot to find an optimal number of garment type clusters.

because the data are unlabelled, we cannot determine the accuracy of the clustering to see if the garments were classified correctly. The inverse is true to an extent, where we assign a label of some sort to the clusters after the fact. The distribution of the number of garments that fall into each of the 5 clusters can be seen in Figure 3.4.

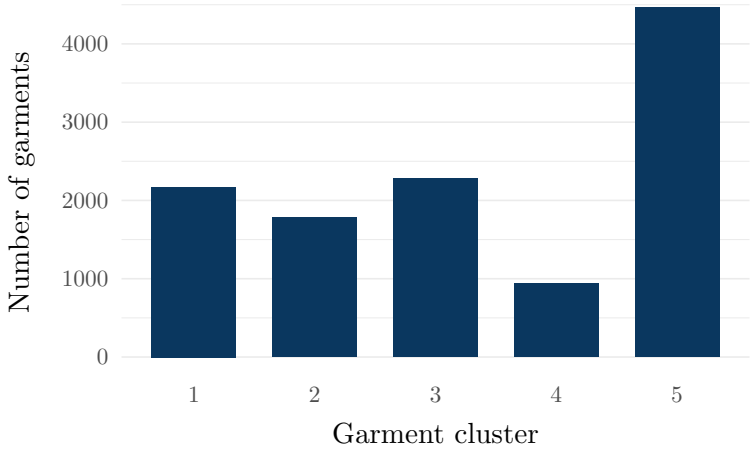

Figure 3.4: Number of training set garments per cluster.

In this plot, we can see that the different clusters contain different numbers of garments; Cluster 5 has the most by far, followed by a relatively equal number between Clusters 1, 2, and 3, while Cluster 4 contains the smallest number. What this shows is that there is an overwhelming majority of the garments that are part of Cluster 5 in the dataset. To get a better understanding of the different clusters, look at Table 3.1, which shows 3 randomly sampled garments from each cluster, along with a brief description of the garment category,[16] and the number of garments that are a part of that cluster.

From this table, we can see that the clustering based the majority of the similarity between garments on their shape and style. The garment colour had no impact on the clustering. Sleeve length and shape group the garments in Clusters 1 and 3, while style groups the garments in Clusters 2 and 4, which are distinctive garment types — more formal blouses, and tank tops, respectively. Cluster 5, those which are classified as T-Shirts, contains the most garments. This could mean that T-Shirts make up a majority of the actual garment types, or because of their lack of a definitive style and shape, T-Shirts cover an array of garments, and it is a very broad

[16]This description can be seen as a 'label' of the data, but it is assigned after the fact based on visual inspection.

Table 3.1: Clustered garment examples, definitions, and counts.

Sample images	Cluster description
	**Cluster 1** Sleeveless and short sleeve tops 2 172 garments
	**Cluster 2** Blouses 1 778 garments
	**Cluster 3** Long sleeve tops 2 285 garments
	**Cluster 4** Tank tops 942 garments
	**Cluster 5** T-shirts and others 4 470 garments

category in which garments can fall. Consider the garments shown in Figure 3.5; they are not the exact definition of a T-Shirt, they just happen to be closer to a T-Shirt than to another garment type, such as a blouse or tank top, for example. If the number of clusters were to be increased, perhaps a more fitting cluster would be found where these anomaly-type garments would fall. Because many garments potentially fall into this 'grey area' of close to a T-Shirt, but not a T-Shirt, it is deemed the reason why Cluster 5 has significantly more observations than the other clusters.

Figure 3.5: Potential anomalies in Cluster 5 showing outlying garment types.

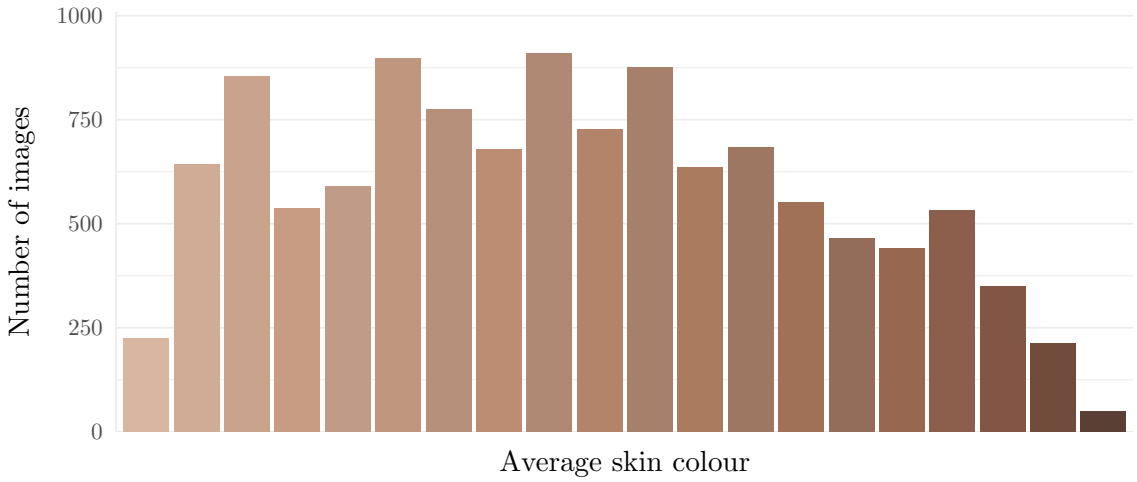

Figure 3.6: Distribution of average skin colours in the training set.

Notwithstanding this, there is a seemingly adequate number of different garment types that make up the data, and the model built with it should not have any bias toward a specific garment style, or colour. Next to be explored is the distribution of skin tones of the women in the images.

3.1.3 Skin colour analysis

The inclusion of diverse skin tones in virtual try-on systems' training data is crucial for addressing model bias and variance. The benefit of including diverse skin tones in the data is twofold. First, it ensures accurate predictions for various skin tones, fostering inclusivity and equity. Second, it reduces overfitting, enhancing the model's ability to generalise to new instances and improving performance on unseen data.

To understand the distribution of the skin tones in the data, a similar process to extracting the average garment colour is followed. Here, a person's skin tone is assumed to be the same on their face and the rest of their body. The images are parsed to a deep learning-based technique for detecting faces in images known as Multi-Task Cascaded Convolutional Networks (MTCNN) (Zhang et al., 2016). Once MTCNN detects and bounds the faces in an image, a masking type process is followed, and only the pixels that make up the face region are kept. From this, the face image is converted from RGB colour space to the Lab colour space, which is a colour model that represents colours based on their lightness, along with two colour components for capturing differences in their green-red and blue-yellow colour dimensions. Lab colour space is more suitable for skin colour analysis as it better represents the human perception of colour differences (Weatherall and Coombs, 1992). Then the average skin colour of the face in the Lab colour space is found by calculating the mean value of each colour channel across the face pixels.

The computed average skin colours are then grouped using *K-means* clustering, resulting in a specified number of distinct colour clusters. As in the garment colour analysis, we set the number of clusters beforehand, and in this case, 20 clusters are used.[17] The average colour for each group is calculated, providing a summary of the skin tones present in the given dataset. The distribution of the different skin tones can be seen in Figure 3.6. The data for this figure can be found in Table A.2, in Appendix A.

In this plot, the colours are arranged in decreasing order of brightness, with the lightest skin

[17] 20 skin tone groups are sufficient to understand the distribution of the diversity.

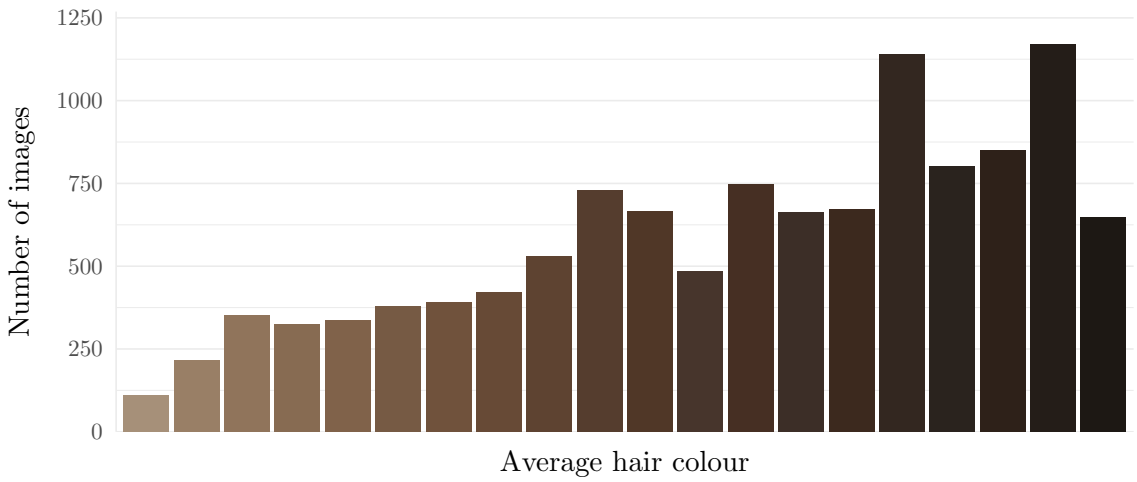

Figure 3.7: Distribution of average hair colours in the training set.

tone on the left, and the darkest tone on the right. We can see that there is a good spread of data across the different skin tones. There is an even spread between light and dark tones in the data, which means different skin colours are well represented in the data. At the extremes — the most pale skin tone, and the darkest skin tone — there is less data, which is to be expected.

This means that the data that the model is trained with will not introduce a bias towards specific tones, making it a more inclusive and fair system for end users. From a modelling perspective, the model will be more robust and will be able to generalise well to unseen skin tones, which is desirable. Similar to skin colour, the hair colour of the person in the images used is an important factor, particularly when handling occlusions, so it is investigated next.

3.1.4 Hair colour analysis

Hair colour, along with skin colour, make up an important part of a person's identity. Because one of the requirements of a virtual try-on system is to retain the person's identity, building a model that factors in hair colour is vital. Apart from building a model that preserves the identity, it also needs to handle occlusions, and a common occlusion to deal with is when there is hair in front of a garment (as shown in Figure 2.4). Because the person's hair has a twofold effect on the model, having diverse hair colours in the data is pivotal in building a robust virtual try-on system.

To analyse the distribution of hair colours across the dataset, we follow a similar process as in the skin colour analysis. In summary, a semantic segmentation mask is applied to the image, from which we extract only the hair region in the image. The average hair colour is found by calculating the mean value of each colour channel across the hair pixels. *K-means* clustering is again applied; this groups the average hair colours per image into different clusters, and the average hair colour is found in each cluster.[18] The resulting distribution of hair colours across the 11 647 images can be seen in Figure 3.7. The data for this figure can be found in Table A.3, in Appendix A.

Not too dissimilar to the skin tone colours, there is a relatively large range of hair colours in the data. However, the darker hair shades seemingly dominate the data, with the majority of the observations being brunette or darker, with fewer observations being blonde and lighter. This in itself may not be problematic, as we cannot ignore the connection between skin tones and hair colours — people with darker skin tones generally have darker hair, however, the converse is not

[18] Again, 20 clusters is deemed sufficient.

entirely true. This is because the melanin production in their melanocytes[19] is higher, resulting in higher levels of eumelanin, which is responsible for darker skin and hair colours (Jablonski and Chaplin, 2000).

In conclusion, the distribution of hair colours in the data may appear 'too' dark based on an initial overview without considering the connection to skin colour. However, once we link the two, the distribution appears more representative of the population, and it is deemed to capture the essence of the problem, and ensures sufficient diversity. The last aspect of the contents of the images, either the person or the garment, to be investigated is the distribution of different poses of the people, which follows.

3.1.5 Pose analysis

Utilising a variety of poses enhances the robustness of the system, enabling it to accurately handle different body positions and orientations that users may adopt whilst trying on virtual clothes. This, in turn, leads to a more realistic representation of how clothing items fit and appear on users, as well as a system that can cater to the nuances of diverse body types, clothing styles, and user interactions.

One key advantage of training a virtual try-on system with an extensive array of poses is its ability to generalise effectively across various situations. This not only improves the overall performance of the system but also ensures that it caters to a broad audience, offering a seamless experience for all users. Furthermore, training with a wide variety of poses allows the system to adeptly handle occlusions that may arise from users posing in different ways, thereby ensuring the accurate rendering of clothing items even when they are partially obstructed by other body parts. An example of this type of occlusion can be seen in Figure 2.3, where the woman's pose leads to a portion of the garment being hidden. These factors collectively contribute to an improved user experience and superior overall system performance, making it a crucial aspect of the development process.

The process of determining the diversity in the poses is much the same as the clustering approach that was followed in finding the garment type diversity. OpenPose is first used to extract the pose key points from an image of a person, which are then parsed to a *K-means* clustering algorithm. An example of these pose key points can be seen in Figure 2.2. The optimal number of clusters is found, and then the images are clustered accordingly.

Here, we can see that the major points are extracted, such as the wrists, elbows, shoulders, hips, arms, legs, spine, neck, hands and fingers, and facial features. Any points that are occluded (the majority of the points in this case, for example), are predicted by OpenPose. After these features are extracted for every image, they are used to find the optimal number of clusters. Once again, we use the elbow plot method to determine the optimal number of clusters, and this can be seen in Figure 3.8.

In this plot, the 'elbow' is not as evident as before, however, it seems to appear best at 4 clusters. Other candidates could be 8 or 17 clusters, however, as the number of clusters is not of overall importance, 4 clusters are sufficient to describe the distribution of diversity. Overall, the elbow plot has a gradual decline as the number of clusters increases, barring a few points where it increases. This can make it difficult to determine the optimal number of clusters, as there is no distinct point where adding more clusters ceases to provide significant improvements in clustering quality.

A gradual decline in the elbow plot without a distinct elbow point may offer valuable insights into the characteristics of a dataset; however, the interpretations are not definitive. One possible

[19] Melanin is produced by cells called melanocytes, which are found in the epidermis and hair follicles, and both hair and skin colour is largely attributed to the distribution of melanin in our bodies.

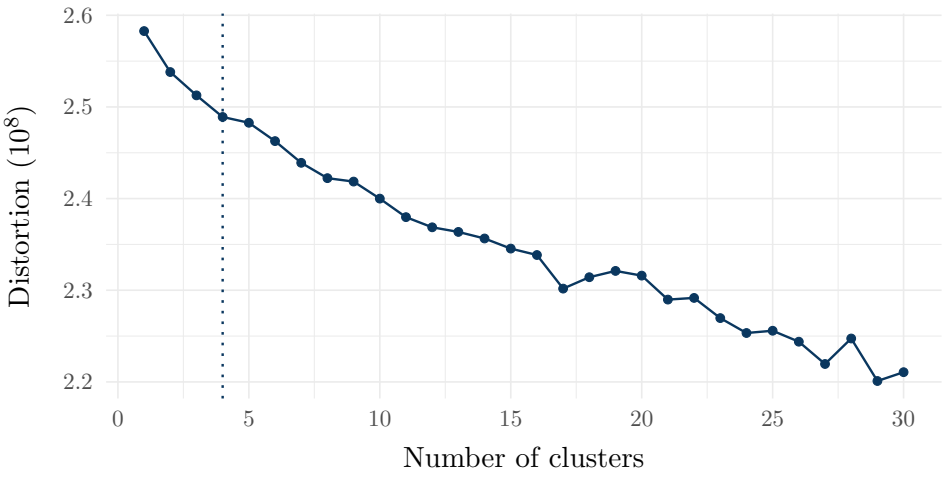

Figure 3.8: Elbow plot to find an optimal number of pose type clusters.

explanation for this pattern is the high diversity within the dataset, which could encompass numerous overlapping or closely related features. The absence of a clear elbow point may suggest that the data does not naturally form well-separated groups, thus posing a challenge for clustering algorithms in identifying distinct clusters. Another potential reason for this observation is the complex structure of the data, which might not be easily captured by clustering algorithms. Nonetheless, 4 clusters will be used, as it is believed that this will give sufficient insight into the distribution of poses. The distribution of the number of images belonging to each of the 4 clusters can be seen in Figure 3.9.

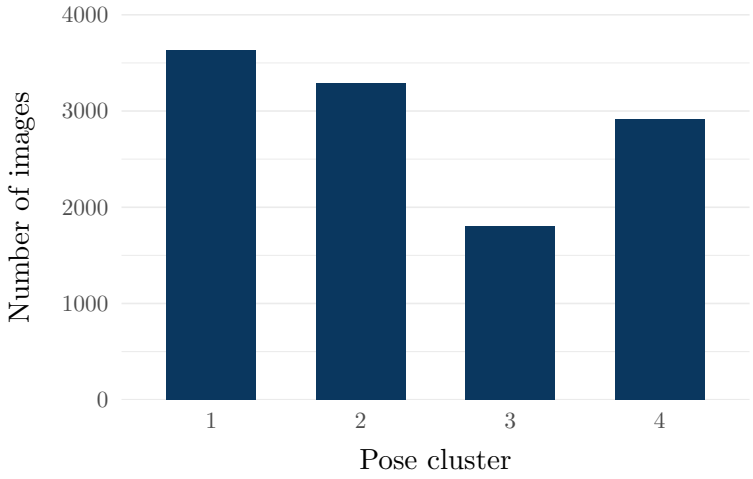

Figure 3.9: Number of training set images per pose cluster.

Cluster 3 has the least number of images, by some margin, while the remaining three clusters have a relatively equal number of the remaining images spread across them. Interpreting the distribution of the number of images across the different clusters, it is determined that there are an adequate number of different poses, even if the differences are subtle across the data. In conclusion, the combination of OpenPose and *K-means* clustering has revealed that there are different pose groupings in the dataset, shedding light on the dataset's structure and diversity. Next to be analysed is the quality of the images.

3.1.6 Image quality analysis

From a machine learning perspective, high-quality images are critical for the efficacy of virtual try-on applications. The quality of the images used in these applications directly impacts the ability of the machine learning models to accurately identify and understand the various elements within the image, such as the shape and dimensions of the user's body, the pose they are in, and the characteristics of the clothing item. If the images are of low quality, these elements may not be discernible, leading to inaccurate and unconvincing try-on results. In contrast, high-quality images provide clear and detailed information, enabling a more precise and realistic virtual try-on. Thus, image quality is critical for the functionality and believability of these applications. The benefits of using high-quality images for virtual try-on are listed below:

1. **Enhanced learning capability**: High-quality images contribute significantly to the richness and diversity of training data for machine learning models. This additional detail aids in feature extraction, leading to better learning and increased accuracy of the models.
2. **Increased model accuracy**: The fidelity of models in predicting the fit and look of a garment on a person is greatly influenced by the quality of images. High-quality images facilitate better object detection and segmentation, leading to improved performance of the models.
3. **Reduced pre-processing**: Images of high quality lessen the need for extensive pre-processing steps, such as noise reduction or the application of super-resolution techniques, before they are fed into the models. This contributes to savings in computational resources and time.
4. **Improved fine detail recognition**: High-resolution images allow for the better discernment of fine-grained features such as textures, patterns, and subtle colour variations. These details, when captured accurately, enhance the realism of a virtual try-on experience.
5. **Resilience to lighting conditions**: High-quality images are more robust to variations in lighting conditions, retaining more information under a variety of lighting scenarios. This ensures consistent performance of the models, regardless of lighting conditions.
6. **Robustness to scale variation**: Images of high quality are more robust to scale variations. This means that essential features are retained even when images are resized, enabling the trained models to generalise better to different input sizes.

To analyse the quality of the images, CleanVision's *ImageLab*, which is open-source software for auditing images, will be used (Northcutt et al., 2021). Both the garment images and the images of the women will be analysed. Basic image information, such as aspect ratio, size, and colour profile will be investigated, along with more in-depth image characteristics such as brightness, blurriness, and entropy. Duplicates in the data are also found and subsequently removed from the data.

Of the 11 647 images of the women analysed, no major issues were found, with the same being found for the garment images. All the images are the same resolution (768×1024), which means they all have the same aspect ratio. None of the images are greyscale, all are in the RGB colour model, and none are overly overexposed or underexposed. However, there were 95 images in both the women and the garment images that were duplicated. These duplicates are removed from both datasets, resulting in a total of 11 552 images across both datasets.

Delving into more technical characteristics of the images, we look at the entropy, blurriness, and brightness of each of the images. Entropy is a measure of randomness that can be used to characterise the texture of the input image. High entropy images have a wide range of pixel values and may contain a lot of 'noise', or a lot of texture and detail. Low entropy images,

on the other hand, may be blurry or only contain a few different pixel values. Blurriness is a measure of the absence of sharpness in an image and is often identified by the presence of clear edges and details in the image. A high blurriness value indicates a clearer image, while a lower value indicates a more blurry image.[20] Lastly, brightness is a measure of the overall lightness or darkness of an image. It is a fundamental characteristic of an image, defined by the average intensity of the pixels in the image.

Again, both the women and garment images are investigated. Because entropy, brightness, and blurriness are generally interconnected and changes in one can impact the others, it makes sense to investigate them together, holistically. The relationships between the different metrics are summarised below.

1. **Entropy and brightness**: A very dark or very bright image may have low entropy because many of the pixel values are similar (either near black or near white). This could result in a loss of detail or texture in the image.
2. **Entropy and blurriness**: A blurry image typically has low entropy because the details and textures are smoothed over, resulting in fewer unique pixel values. Conversely, an image with high entropy is likely to have more detail and less blurriness.
3. **Blurriness and brightness**: Excessive brightness can lead to blurriness in an image. When an image is too bright, it can become overexposed, causing a loss of detail in the brightest areas. Similarly, if an image is too dark, it can become underexposed, causing a loss of detail in the darkest areas. Both overexposure and underexposure can lead to an appearance of blurriness because they reduce the contrast between different parts of the image.

To look at the distribution of each of the metrics across the dataset, we create a scatterplot[21] showing the relationship between the three metrics. Consider Figure 3.10, which shows the entropy of images versus their blurriness. Brightness is introduced by the size of the points plotted, meaning all three metrics are captured in one visualisation.

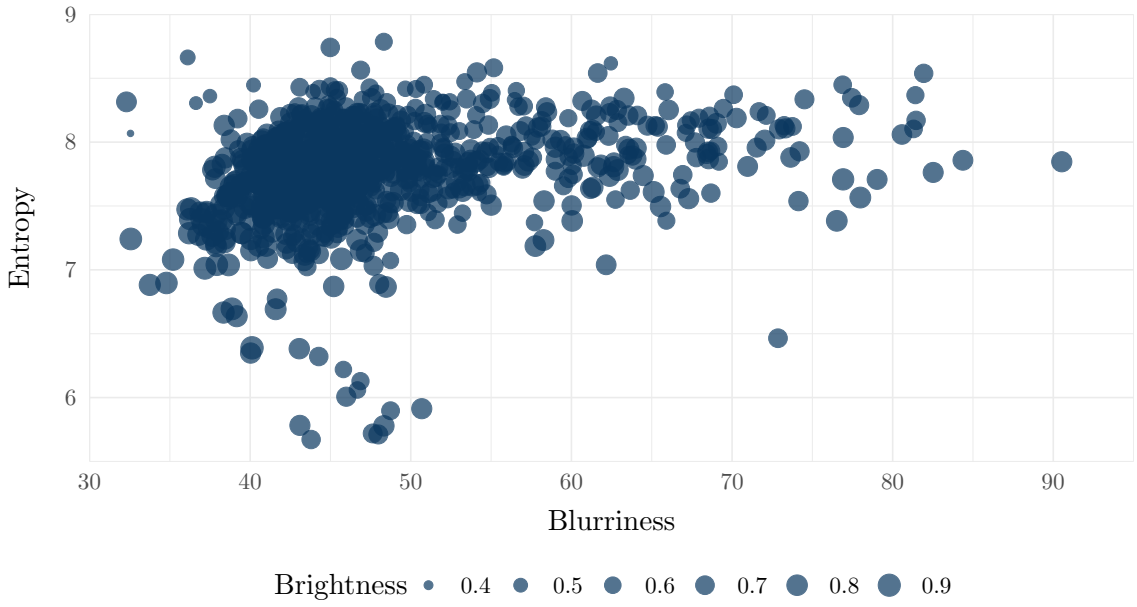

Figure 3.10: Scatterplot of women images showing image blurriness, entropy, and brightness.

[20] This is somewhat counterintuitive, however, it is the metric used by CleanVision's ImageLab.
[21] For visualisation purposes, a random sample of 1 000 images was used to create the plots.

In this plot, we can see the distribution of entropy ranges between 5 and 9, while the blurriness ranges from 30 and 100. Brightness typically ranges from 0.4 to 0.9. The entropy of an image (H) is calculated as:

$$H = -\sum_{i=0}^{L-1} p(i) \log_2 p(i), \tag{3.1}$$

where i represents the particular intensity value, and L is the total number of possible intensity levels in the image (for an 8-bit image, this would be 256 levels, ranging from 0 to 255). The probability of occurrence of intensity level i in the image, denoted as $p(i)$, is found using the relative frequencies of each intensity level i. If a particular intensity value does not occur in the image, then $p(i)$ is zero and by convention, $0 \log_2 0$ is also taken to be zero, which aligns with the continuous extension of $x \log_2 x$ as x approaches zero. Brightness (B) is computed as the normalised average pixel intensity of the image, which is the sum of the intensities of all pixels divided by the total number of pixels, and it is given as:

$$B = \frac{1}{N} \sum_{i=1}^{N} I_i, \tag{3.2}$$

where N is the total number of pixels in the image, and I_i is the intensity of the i^{th} pixel. Blurriness is measured using a Laplacian filter for edge detection, where the variance of the filter response is calculated, with a lower variance indicating a blurrier image and a higher variance indicating a sharper image. This variance is calculated as:

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (L_i - \mu)^2, \tag{3.3}$$

with σ^2 the variance of the Laplacian filter response, L_i is the Laplacian filter response at the i^{th} pixel, while μ is the mean of the Laplacian filter response.

Entropy and blurriness are relative metrics, meaning we should look at them in the context of each situation. For example, consider the blurriness values in Figure 3.10, the majority of the images seem to be very blurry, as they are grouped on the lower end of the scale, however, all are still well above 0, and once they were investigated, they were deemed to be not blurry. The images on the far right have a high blurriness value, which just means that they are significantly sharper than the other images. The same can be said for the entropy values; images with a low value may be less complex when compared to the others, but still hold enough detail to be high-quality overall. Brightness can be treated as an absolute value, meaning we can compare the values to known values. Ideally, the average brightness should be around 0.5 (normalised), here, the average brightness across the women images is 0.71. However, due to each image possessing a white background, there is a disproportionate increase in brightness, which means the measured results will be higher than they otherwise would be. Despite this, there is a sufficient range of brightness levels across the different images.

The same is done for the garment images, and the resulting distributions can be seen in Figure 3.11. Here, we can better see the relationship between blurriness and entropy — as images become clearer (a higher blurriness value), so too do the entropy and complexity identified. Because the garment image is a standalone image, with less 'going on' in the image, meaning there is no additional information other than the garment, we would expect the entropy values to be lower, and the blurriness values to be higher than the women images. This is the case, with entropy ranging from 3 to just over 7 (compared to 5 to 9 for the women images), while more images are clearer and sharper. The brightness across the two datasets is much the same.

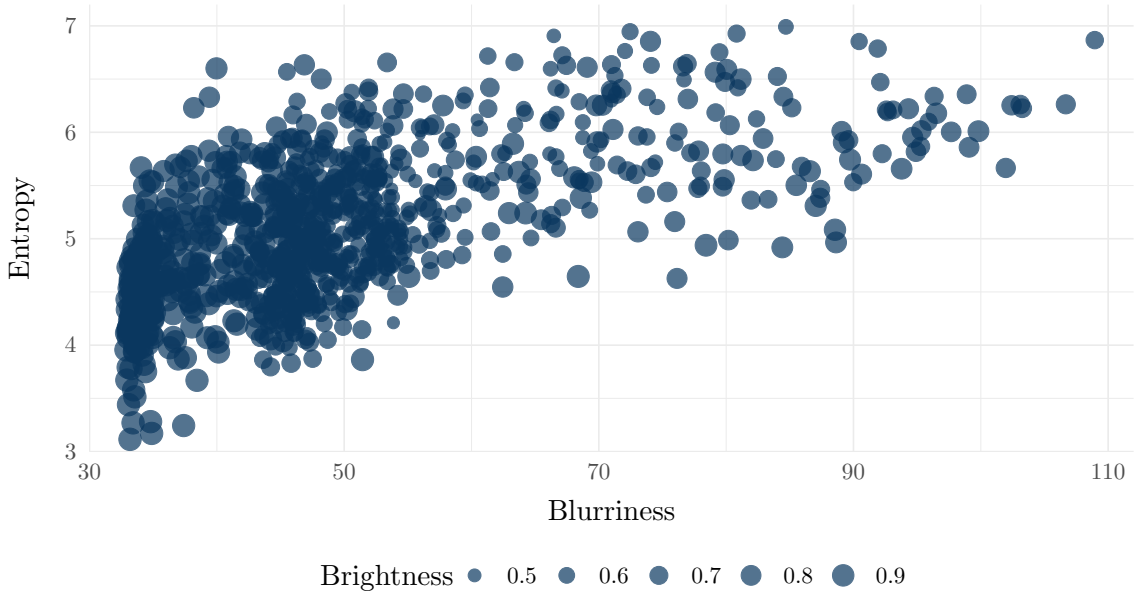

Figure 3.11: Scatterplot of garment images showing image blurriness, entropy, and brightness.

Table 3.2: Average quality metrics in the different datasets.

Metric	Women images	Garment images
Entropy	7.78	5.14
Blurriness	47.6	49.8
Brightness	0.71	0.74

Consider Table 3.2, which shows the average values for each of the three metrics across the women and the garment datasets. Both the average blurriness and brightness are similar between the two groups, however, there is a notable decrease in the entropy of the garment images. This highlights the impact of 'noise' in the data on the perceived entropy, with things such as hair, accessories, body features, and pose adding to the complexity of the women images.

To further understand each of these metrics, consider the figures that follow. Comparisons between women images and garment images with both high and low entropy are shown in Figure 3.12. The images of women are displayed on the left; the one with lower entropy appears slightly blurred and plainer compared to the clear and crisp image of the woman adjacent to it. The high entropy image features nuanced and intricate patterns on both the garment and the woman, contributing to its overall complexity. The low entropy garment is very difficult to make out in the image itself, because it blends in with the background, and not a lot of information can be extracted from it. On the other hand, the high entropy garment has a lot of detail and patterns, making it more complex.

Figure 3.13 shows a comparison between clear and blurry images. The first woman image is clear, and the edges are easily distinguished, while the image next to it is more blurred. The differences in the garments are interesting, as patterns seemingly make edges more discernible, and allow the image to be sharper, as shown in the first garment image. The second garment image is considered blurry as a direct result of its brightness, making it very difficult to see what is going on in that image, which is akin to blurriness.

Lastly, Figure 3.14 shows images with different brightnesses. As expected, images that have more colours closer to black are considered duller than those images with more colours closer to

(a) Low entropy (b) High entropy (c) Low entropy (d) High entropy

Figure 3.12: Comparisons between low and high entropy images.[22]

(a) Clear (b) Blurry (c) Clear (d) Blurry

Figure 3.13: Comparisons between clear and blurry images.[22]

white, which are considered bright.

(a) Bright (b) Dull (c) Bright (d) Dull

Figure 3.14: Comparisons between bright and dull images.

In these images, the relationship between brightness, blurriness, and entropy can be seen, and how it affects the image quality. Overall, apart from the duplicates, the images in both datasets are high-quality. There is a good distribution of entropy, clarity, and brightness across the images, meaning a robust, high-quality, perceptually convincing virtual try-on system can be built using this data. This brings an end to the EDA, with concluding remarks to follow.

[22]Zoom in to get the best comparison between the images.

3.2 Conclusion

In conclusion, the dataset utilised in this research encompasses a wide array of paired images, with each pair consisting of a garment image and a corresponding image of a woman wearing that garment. Detailed EDA reveals substantial diversity in garment, skin, and hair colours. This broad variety suggests the potential for a comprehensive and robust virtual try-on system capable of generalising well. The dataset also exhibits complexity and versatility, demonstrated by the range of poses and garment types contained therein. These attributes are essential for the development of a robust model.

Quality assessment of the data, which focused on entropy, blurriness, and brightness, shows no significant issues that could potentially compromise the reliability of the virtual try-on system. However, 95 duplicate images were found and were removed.

Overall, the dataset, with its diversity, richness, and high quality, presents a suitable foundation for the construction of an effective virtual try-on system, which produces photorealistic results akin to that of Lee et al. (2022). The next chapter of this research will delve into the model formulation and research design. Insights gathered from the data analysis will guide this process, ensuring that the devised model utilises the quality and diversity of the images it is built on. This approach is expected to lead to more accurate and reliable outcomes in the virtual try-on system, producing high-resolution photorealistic virtual try-on images.

4 | Methodology

In the opening chapter, we introduced the concept of virtual try-on, outlining the specific research objectives of this study. Chapter 2 took a deep dive into the existing body of literature surrounding the topic of virtual try-on, creating a comprehensive landscape of the current understanding in this area. In the third chapter, the data that the models are built on was explained and explored, providing context and understanding of the images the model can expect. Here, in this chapter, we transition from theory to practice, as we delve into the technical intricacies of virtual try-on, unearthing the complexities and challenges inherent in its implementation, and how we use that to answer the research question.[23] For the sake of completeness, the research question, with specific objectives as outlined in Chapter 1, is repeated below:

> *How can the high-resolution photorealistic virtual try-on capabilities of the work of Lee et al. (2022) and Choi et al. (2021) be combined with the robust and practical nature of the work of Ishikawa and Ikenaga (2022) and Lewis et al. (2021) to build a system that can generalise at test time?*

This chapter starts with an overview of the proposed models and their architecture, and then it delves into the specifics of each module,[24] with two main areas: *garment extraction* and *virtual try-on image synthesis*. Each of these areas is further explained in their relevant sections. Where applicable, pre-trained models are used, assuming they are trained on the same data that we use in this study, and for the same purpose, and if not, the training process of the models is discussed as well. Using pre-trained models is common practice in virtual try-on tasks.[25] The application and implementation of the model are discussed in this chapter too, with the results and performance being explained in the chapter that follows. We start by explaining the proposed model in the next section.

4.1 Proposed model

The primary objective of this research can be summarised as follows: given an image, I_g, of a person wearing the target garment g, our goal is to extract this garment, denoted as g_e, and refine it using context-aware inpainting to produce the garment image g_r. This refined garment is then synthesised onto a target person in the image I_p to generate the final virtual try-on image I_v. To accomplish this, we employ two distinct standalone models: the garment extraction and inpainting model, and the virtual try-on image synthesis model.

The output of the garment extraction and inpainting model is used as input for the image synthesis model. As the models can function as standalone models, their training and optimisation

[23] It is assumed the reader is familiar with basic computer vision concepts, as they are not discussed in-depth in this chapter.

[24] The words 'model' and 'module' are used interchangeably, with 'model' never referring to the women in the images.

[25] Models trained to the required level for these tasks need substantial computational resources, so leveraging pre-trained models is a good alternative.

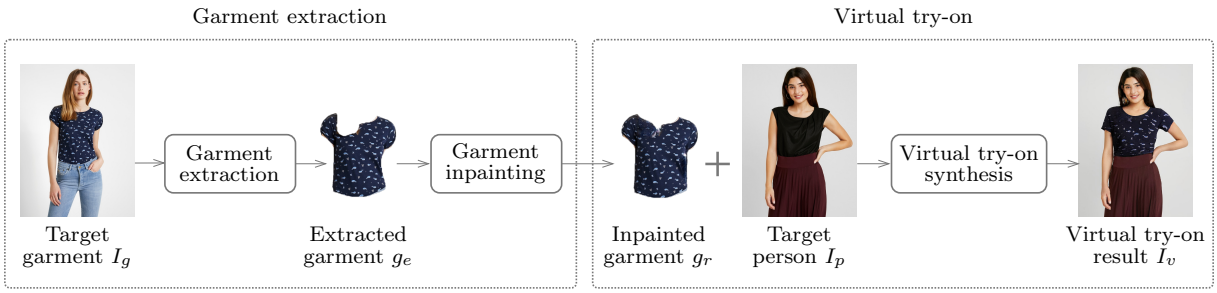

Figure 4.1: Model architecture overview showing both the garment extraction model and the virtual try-on model overviews with the target garment on a person, the extracted and inpainted garment, the target person, and the resulting virtual try-on image.

are handled separately.[26] However, they can be configured and architectured to be one overarching model that handles the entire virtual try-on with garment extraction process. This process can be seen in Figure 4.1, showing the high-level process of the entire model, with two distinct models: the garment extraction model shown on the left, and the virtual try-on image synthesis model shown on the right.

This summarised architecture shows the flow of the model from extracting the target garment g_e off of the person in I_g, to enhancing the garment image with inpainting to create g_r, and finally synthesising the final virtual try-on image I_v where the target person I_p is wearing the refined target garment g_r. In the sections that follow, each model is further explained, starting with the garment extraction model.

4.2 Garment extraction and inpainting models

Like the overall model, the garment extraction model has two distinctive submodels: the first extracts the garment from the input image I_g, and the second enhances the extracted garment g_e using inpainting to create g_r. Each of these submodels will be further explained in the sections that follow, starting with the garment extraction model.

4.2.1 Garment extraction

To extract the target garment, g from a given input image, I_g, an off-the-shelf model known as Part Grouping Network (PGN), developed by Gong et al. (2018), is used for semantic segmentation of the image. Semantic segmentation is a process in computer vision where each pixel in an image is labelled according to the category of the object it belongs to. As an example to show what the output of PGN looks like, consider the images in Figure 4.2. Here, the input image is shown, as well as the resulting segmentation map of that image, with different semantic regions masked in unique colours.

PGN is a unified architecture crafted for semantic part segmentation and instance-aware edge detection. The network is co-trained for these two subtasks, both of which are pixel-wise classification problems. The network's structure is based on Fully Convolutional Networks (FCNs), and it consists of a backbone sub-network, a branch dedicated to semantic part segmentation, another branch for instance-aware edge detection, and a refinement branch. We are focused on the segmentation branch, however, the edge detection branch output is concatenated with this output to result in the final semantic segmentation map.

[26] It would be interesting to explore the difference between training them as one model, as opposed to separate models, however, this is out of scope for this study.

(a) Input image of a person. (b) Resulting segmentation map.

Figure 4.2: An example illustrating what a segmentation map created by PGN looks like. The input image is shown on the left, with the resulting segmentation map shown on the right.

The backbone sub-network uses a repurposed DeepLab-v2 network, based on ResNet-101, as the human feature encoder (Chen et al., 2017). DeepLab-v2 is a semantic segmentation architecture that expands upon the original DeepLab model by introducing an Atrous Spatial Pyramid Pooling (ASPP) strategy. This technique involves employing multiple dilated convolutions with distinct rates in parallel on the input feature map, which are subsequently integrated. Since objects belonging to the same class can vary in size within an image, ASPP effectively allows for the accommodation of these different object scales.

ASPP is a technique used in convolutional neural networks for semantic image segmentation tasks. The technique allows the model to capture multi-scale context by using several parallel atrous (or dilated) convolutions with different rates. A summarised version of ASPP is shown in Figure 4.3, which shows the input feature map, four 3×3 kernels with varying dilation rates, and the concatenated output feature map. The goal in this snapshot is to classify the centre pixel, highlighted in orange. One can see that the different feature maps will capture information at varying distances from the pixel of interest, allowing more context to be derived. The rate refers to the rate of dilation in atrous convolutions, and it represents the 'gap' or 'skip' in the kernel. To elaborate, in a standard convolution, a 3×3 filter would be applied to a 3×3 region of the input. However, with a dilation rate of 2 in atrous convolution, for instance, the same 3×3 filter would be applied to a 5×5 region of the input, skipping over some input values based on the dilation rate.[27] Furthermore, ASPP can be broken down into two concepts:

1. **Atrous Convolution**: This is a type of convolution operation where the filter is applied to the input with gaps. By introducing these gaps, or dilation, the convolutions have a wider field of view over the input, capturing more spatial context in the image.
2. **Spatial Pyramid Pooling**: This involves conducting pooling operations at different scales, resulting in an output that is robust to changes in object scale and size.

ASPP combines these two concepts. It uses multiple parallel atrous convolutions with different dilation rates, which essentially means it applies filters of various sizes to the image simultaneously. This allows the model to capture objects and features at various scales. After the convolutions, the outputs are then concatenated and processed to yield the final segmentation result. The main advantage of ASPP is that it can capture a wide range of features with different scales in an image, making it highly effective for semantic image segmentation tasks, and this is what PGN is predicated on.

In the context of the PGN, the semantic part segmentation branch plays a crucial role in breaking down the input image into distinct parts. It achieves this by performing pixel-wise recognition,

[27] A dilation rate of 1 implies an undilated convolution.

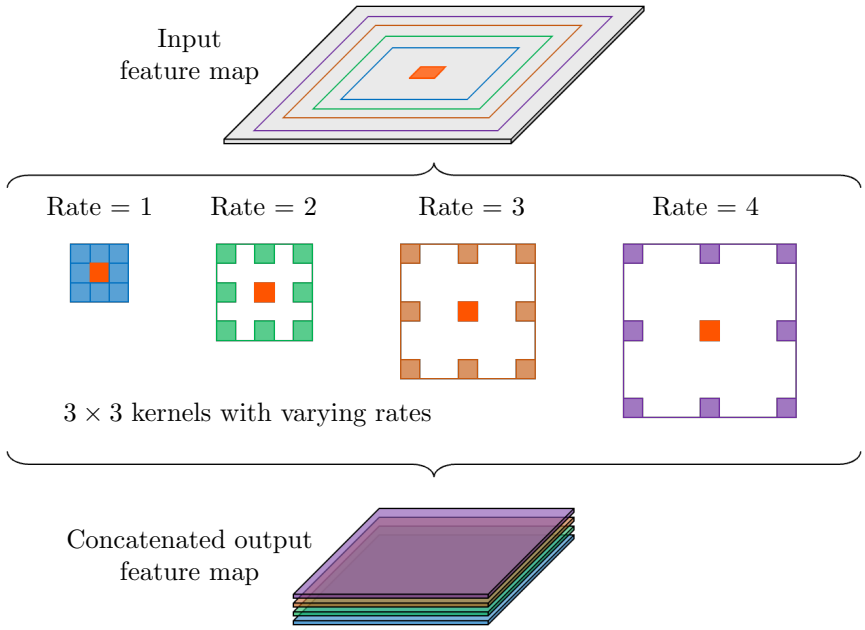

Figure 4.3: Atrous spatial pyramid pooling overview showing an input feature map, four 3×3 kernels with varying dilation rates, and the concatenated output feature map. These are all applied to classify the centre pixel, highlighted in orange.

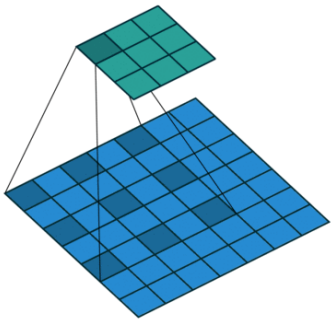

Figure 4.4: Animation showing how pooling works in an atrous convolution using a kernel with a rate of 2 (Kumar, 2020).[28]

which essentially means that each pixel in the image is classified individually, according to the semantic group it belongs to. For example, in an image of a person, pixels may be labelled as "arms", "legs", "hair", and so on, depending on which part of the person they represent. PGN is trained using the Crowd Instance-level Human Parsing (CIHP) Dataset, which consists of images that are labelled with pixel-wise annotations on 20 categories (Gong et al., 2018). Subsequently, PGN can label clothing item regions too, and not just body parts, making it ideal for garment extraction as different garments are segmented by the network.

The context aggregation pattern used in this branch is an essential technique that helps to capture more complex, context-aware information. This is done by applying average pooling with various kernel sizes. Pooling is a process that reduces the spatial size of the input by summarising regions of the input with a single value. An animation showing how pooling works in an atrous convolution can be seen in Figure 4.4, here the kernel has a rate of 2. By using different kernel sizes for pooling, the model can extract features at different scales, which can be useful for capturing more abstract or global features.

[28]To view the animation, use Adobe Acrobat Reader as the PDF viewer.

The output from this context aggregation operation is then fed into a 1×1 convolutional classifier. A 1×1 convolution, also known as a network-in-network layer, is a powerful tool in Convolutional Neural Networks (CNNs) that can be used to change the dimensionality of the input. It is typically used for channel-wise dimensionality reduction or increase. In this case, the 1×1 convolutional classifier is used to transform the complex, multi-channel features from the context aggregation into a set of class scores that represent the likelihood of each pixel belonging to each semantic part. These scores can then be interpreted as a segmentation map where each pixel is assigned the label of the semantic part that it most likely belongs to.

The next branch in the network is instance-aware edge detection, which imposes deep supervision at each side-output layer of the final three blocks of ResNet-101 to learn hierarchical representations for edge predictions. This branch uses ASPP to discern boundaries at multiple scales and a pyramid pooling module to gather more global information for enhanced reasoning. The refinement branch integrates segmentation and edge predictions back into the feature space, to mutually enhance segmentation and edge results, of which we only use the segmentation results. Now with the semantic segmentation map found, it can be used to extract the garment and create a standalone image.

Consider the segmentation map that is produced by PGN, shown on the right in Figure 4.2; there are multiple segments, each with a unique colour. For example, the hair region is mapped in red, while the face is blue, and the top garment region is mapped in orange. Having these unique colours per region allows us to easily identify and extract the desired region. The procedure to extract the garment using the output from PGN is shown in Figure 4.5, which shows the general architecture and the intermediate and final outputs. To extract the garment, the following steps are followed:

1. The semantic segmentation map, S_g, derived from the PGN model using I_g, is used as input. From it, only the pixels that match the top garment region are kept — these are the orange pixels.[29] All other pixels in the image are converted to black. This results in a black-and-orange map S_{g_e}.
2. S_{g_e} is converted to a binary mask, M_{g_e}, by a pixel-wise conditional operation, where if the pixel's colour matches the specified orange colour, it is converted to white, and black otherwise.
3. Because an image is a collection of pixels akin to a matrix, the concept of pairwise pixel multiplication can be used. The binary mask M_{g_e} is multiplied with the original target garment image I_g. Each image is essentially a 768×1024 matrix, and they are multiplied together to result in another 768×1024 image. However, only the garment remains in the final image, which is denoted as g_e.

Now that the garment has been extracted, it can be parsed to the context-aware image inpainting model. How the person in the original image is posing has a great effect on the success of the garment extraction model; PGN creates a segmentation map that is representative of the input image, meaning hair-garment or body-garment conflict will show in the segmentation map. This is seen in Figure 4.5, where the woman's hair is occluding a region of the garment — this results in a 'hole' in the extracted garment needing to be filled. This is the information loss that Ishikawa and Ikenaga (2022) refer to. To overcome this, context-aware image inpainting can be used to supplement that 'lost' garment information, that is, the parts of the garment occluded by either hair or body parts. This inpainting is discussed in the section that follows.

[29] The exact colour in RGB is (254, 85, 0).

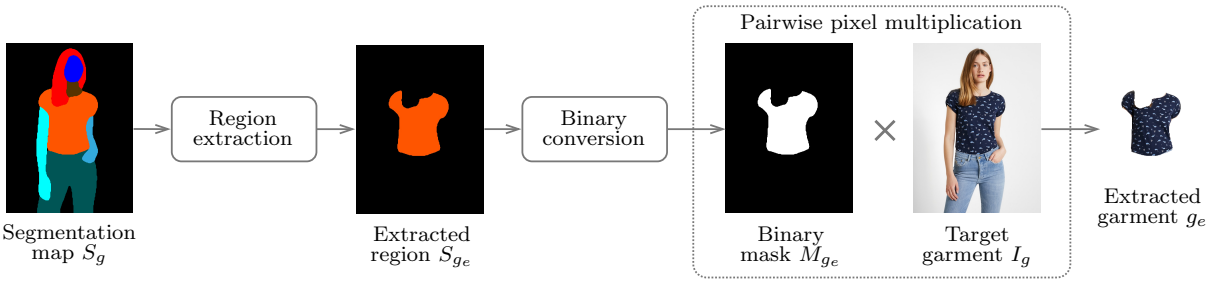

Figure 4.5: Garment extraction architecture showing the input segmentation map derived from PGN and the resulting extracted garment found by multiplying the target garment image and a binary mask.

4.2.2 Garment inpainting

As outlined in Section 2.4.3 in the literature review, the inpainting module used for this study is found in the work of Yi et al. (2020). Context-aware image inpainting will be used, having shown promise in their study as the algorithm uses the semantics of the surrounding parts of an image to generate 'context' for the holes that are to be filled. We utilise their pre-trained Contextual Residual Aggregation (CRA) model to perform the inpainting. Because image inpainting is not the primary focus of this research, the architecture of the CRA model will be explained, but not down to a granular technical level — the model is essentially used 'as-is'. Although the model is not trained specifically for garments, Yi et al. (2020) show that the image category used for training has little effect on the performance of the model across different types of images.

Figure 2.9 in Section 2.4.3 showcased the image editing capabilities of inpainting, however, for our application inpainting will be applied for image correction and restoration, particularly to the extracted garment image. Typically, inpainting is used as a method to remove unwanted objects from an image, although it does have generative capabilities as it fills in the desired areas of an image. Inpainting is a technique used to predict what the values of certain pixels will be by considering the surrounding context of those pixels.

In our method, we are extracting the garment from one image and using it in another; it is in this extraction where information loss occurs, either by hair-garment conflict, or body-garment conflict in the image the garment is extracted from. To illustrate this, consider Figure 4.6, which shows hair-garment conflict which leads to a hole in the extracted garment, i.e. information loss. The result, found by applying the CRA inpainting model, is shown on the right — here, the previously occluded parts of the garment have been filled with plausible values which are based on the surrounding context. Notice that intricate patterns, such as the small items on the garment, are generated by the inpainting algorithm as well, although not to an indistinguishable level. Some artefacts remain in the garment image, which is to be expected based on its complexity, however, the garment is more akin to the ground truth than before. This showcases context-aware inpainting's capabilities to be used for garment restoration.

Before delving into the technical details of the model, it is important to note that inpainting models require a binary mask to direct the algorithms as to where to fill in the missing information. Although this seems like a trivial task, creating a mask for an irregular shape and size, and in a non-random position is complex. The inpainting papers reviewed (discussed in Section 2.4.3) all either use fixed mask placement to test and train their models, or they use random mask generation, where the size and shape of a mask is randomly assigned per image. Neither of these methods is adequate for our desired application.

To overcome this shortfall, an approach of identifying pixels to be inpainted is developed. Although rudimentary, the approach has proven to be effective. Two assumptions are made when

(a) Hair-garment conflict.

(b) Extracted garment with hole.

(c) Inpainted garment.

(d) Original (ground truth) garment.

Figure 4.6: Image inpainting using CRA for garment restoration. Shown are the garment on a person highlighting the hair-garment conflict, the extracted garment with the hole, the inpainted image where the hole has been filled based on the surrounding context, and the ground truth of the garment image.

generating the binary masks: the first is that only hair or arms and hands can create occlusions in the garment, which means that only areas of a garment occluded by any of hair, arms or hands will be masked for inpainting. And the second is that an occlusion will only be masked if it is not just at the edge of the garment, meaning there is some garment visible on either side of the part causing the occlusion. Under these assumptions, we perform pixel-wise row and column operations in the image.

All the pixels that are within the arm and hand regions identified in the semantic segmentation map found by PGN are iterated over, and if the pixel in question has at least one pixel of the garment on either side (top and bottom) of it, then it is masked. If only one side of the pixel (either only top or only bottom) has at least one garment pixel, then it is not masked, and likewise for when there are no garment pixels in the same column. Similarly, for the hair region, row-wise operations are performed where the pixels are iterated over and masked only if garment pixels appear on either side — both left and right — of the pixel in question. It is not masked for all other cases. A pixel can be masked by both, or only one, of the operations. To illustrate this, consider Figure 4.7 which shows the original image, I_g, from which the garment is extracted, the semantic segmentation map, S_m, with only the areas of interest (i.e. the arms, hair, and garment), and the resulting binary mask found by the pixel-wise operations, which is denoted as M_m.

We can see that it is not a perfect fit, however, the mask is generally sufficient in directing

(a) Input image I_g. (b) Segmentation map S_m. (c) Inpainting mask M_m.

Figure 4.7: An overview of what a generated inpainting mask looks like compared to the original image and the related semantic segmentation map showing only the areas of interest.

the inpainting algorithm to the required areas. There are some obvious shortfalls, such as adding additional pixels to inpaint, and missing pixels that need inpainting; it is recommended a further study to improve this method be conducted, however, it is considered out of scope for this paper. The mask in Figure 4.7c is the mask used to generate the inpainted garment shown in Figure 4.6c; note the additional pixels not requiring inpainting being generated in the bottom right of the garment. Now with the mask generated, the methods used by Yi et al. (2020) are discussed next.

Contextual residual aggregation

The inpainting process involves a CRA mechanism, which uses a generator as its only trainable component. In summary, this method involves computing attention scores using the Attention Computing Module (ACM) and determining contextual residuals, which are the differences between the blurry image and the original. These residuals are then combined with the attention scores through an Attention Transfer Module (ATM), producing aggregated residuals. These aggregated residuals are added to the inpainted low-resolution image to produce a sharp output in the mask region, while the original image is maintained outside this mask area. This results in a detailed, high-resolution output that combines elements of the original image with enhancements made by the generator.

Traditional convolutions are problematic when dealing with irregular holes, leading to visual anomalies such as colour inconsistency and blurriness (Yi et al., 2020). Some methods such as partial convolution and gated convolution have been introduced to address these problems (Liu et al., 2018; Yu et al., 2019a). They offer improvements by utilising dynamic feature selection mechanisms and dealing with valid pixels only.

Another method, Contextual Attention, developed by Yu et al. (2018), allows long-range spatial dependencies during inpainting and fills holes with pixels based on the context both close to the hole, and the context further away. However, its two-phase process can be computationally demanding. CRA improves on this by computing the attention scores once and reusing them, reducing computational overhead. Furthermore, the practice of differentiating between an image and its blurred version to represent high-frequency images, used for tasks like edge detection and feature extraction, is inefficient. The CRA method employs this concept more effectively by decomposing the input image into low and high-frequency components, refining the image processing and hole-filling processes.

Consider Figure 4.8, which shows the general architecture and flow of the proposed CRA model. This model takes the high-resolution input image g_e and downsamples it proportionally so that

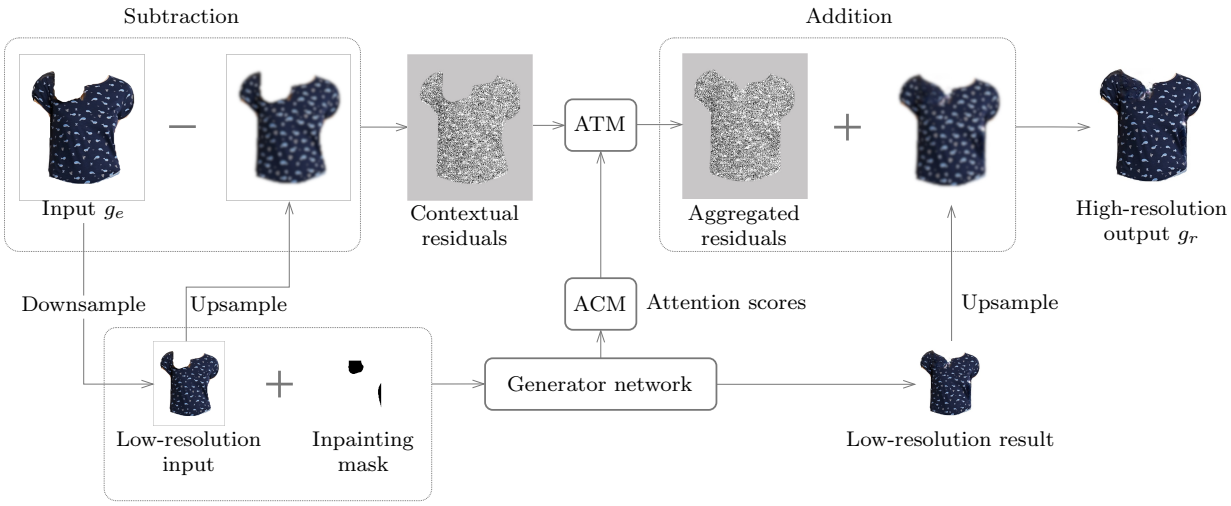

Figure 4.8: Contextual residual aggregation model architecture showing the high-resolution input image, the high-resolution inpainted output, and the intermediate outputs. Included are the contextual and aggregated residuals, the downsampled images, and the blurry upsampled images used to synthesise the final output, along with the inpainting mask.

at least one dimension is 512 pixels, in our case, the downsampled dimensions are 512×384. This is then upsampled back to the original dimensions to produce a blurry image, which is subtracted from the input image to produce the contextual residual image.

The model leverages Gaussian blurring, which replaces each pixel's value with a weighted average of its surrounding pixels' values, where the weights are determined by a Gaussian function, thus reducing noise and detail in the image. This is done to decompose the image into low-frequency and high-frequency components. The low-frequency components of the image are found by averaging the neighbouring pixels through Gaussian blurring. The high-frequency components, which are the contextual residuals, are found by subtracting the low-frequency image (i.e. the blurred image) from the original input.

What makes CRA distinctive is that it incorporates both feature-based context and residuals in its computations. The idea of contextual attention is used to calculate attention scores by determining the regional affinity between patches within and around the missing regions, guided by the calculated mask. This method enables the transfer of relevant features and residuals into masked regions. The process is executed through two primary components: the Attention Computing Module (ACM) and the Attention Transfer Module (ATM).

In the ACM, the attention scores calculation is based on the regional affinity derived from a high-level feature map. This feature map, P^l, is partitioned into patches, and it has l layers. The ACM then measures the cosine similarity between patches located both within and outside of the missing regions, as follows:

$$c_{i,j} = \left\langle \frac{p_i}{\|p_i\|}, \frac{p_j}{\|p_j\|} \right\rangle, \tag{4.1}$$

where p_i is the i^{th} patch from outside the masked region, and p_j is the j^{th} patch from inside the masked region. The formula calculates the inner product of these two vectors once they have been normalised using their respective Euclidean norms. The softmax function is then applied to each cosine similarity $c_{i,j}$ to obtain the attention score for each patch, as below:

$$s_{i,j} = \frac{e^{c_{i,j}}}{\sum_{i=1}^{N} e^{c_{i,j}}}. \tag{4.2}$$

Here, $s_{i,j}$ is the attention score of a patch, and N is the total number of patches outside of the masked region. In this case, a patch has dimensions of 3×3 pixels, and the high-level feature map, P^l, is $32 \times 32 \times 3$. P^l is found in the refinement network of the generator; a more comprehensive discussion of this topic follows in Section 4.2.2. Now with the attention scores of each patch found, the masked regions are filled in the lower-level feature maps. This is done by summing the weighted contextual patches by the attention scores in the ATM, as follows:

$$p_j^l = \sum_{i=1}^{N} s_{i,j} p_i^l. \tag{4.3}$$

In this equation, $l \in \{1, 2, 3\}$ represents the layer number. Yi et al. (2020) use 3 layers, however, more layers can be used as needed. p_i^l is the i^{th} patch outside the masked region extracted from layer P^l, while p_j^l is the j^{th} patch inside the masked region to be filled. As the input size varies, the size of the patches and the dimensions can vary, this allows CRA to inpaint images at ultra-high resolutions.

The next step is to calculate the aggregated residuals. The objective of residual aggregation is to compute residuals within the missing region to generate the sharp details of the masked content. The residuals corresponding to the missing content are determined by accumulating the contextual residuals, which have been weighted and derived from the preceding steps in the ATM. This aggregation happens as follows:

$$R_j^l = \sum_{i=1}^{N} s_{i,j} R_i^l, \tag{4.4}$$

where R^l is the aggregated residual image derived from layer l; moreover, R_i^l is the i^{th} patch in the residual image extracted from the unmasked regions in P^l, and R_j is the j^{th} extracted patch from the masked regions in P^l. This aggregated residual image, R^l, is then found by combining all instances of R_j^l. R^l is concatenated with feature maps at l different instances in the generator network to obtain the final result, g_r, which is discussed next.

Generator network

The generator network is a two-stage coarse-to-fine network; the coarse network generates an approximate version of the masked regions, M_m, in g_e, while the subsequent refinement network focuses on crafting more detailed and precise results using the output of the coarse network and the ATM. The entire generator network architecture is seen in Figure 4.9, which shows both the coarse and refinement networks and the input and output for each, including the ACM and ATM modules.

The coarse network takes an input image, g_e, and a binary mask, M_m, which shows the regions to be inpainted. The dimensions of this input image and the output image need to be 512×512, however, our images have previously been downsampled to 512×384. To this end, we add a pre-processing step where the image is resized to the correct dimensions. To do this is a straightforward task, and because our image backgrounds are always white (as we are dealing with the extracted garment), we can add additional padding of 64 white pixels on either side, left and right, of the image. Doing this resizing has no impact on the results as the additional pixels do not change the context of the inpainted regions.

Now, to augment the receptive field and to ensure computational efficiency, the input dimensions are reduced to 256×256 prior to undergoing convolution within the coarse network. This contrasts with the refinement network that operates at the original 512×512 resolution. The crude estimates produced by the coarse network are naively blended with the input by overlaying

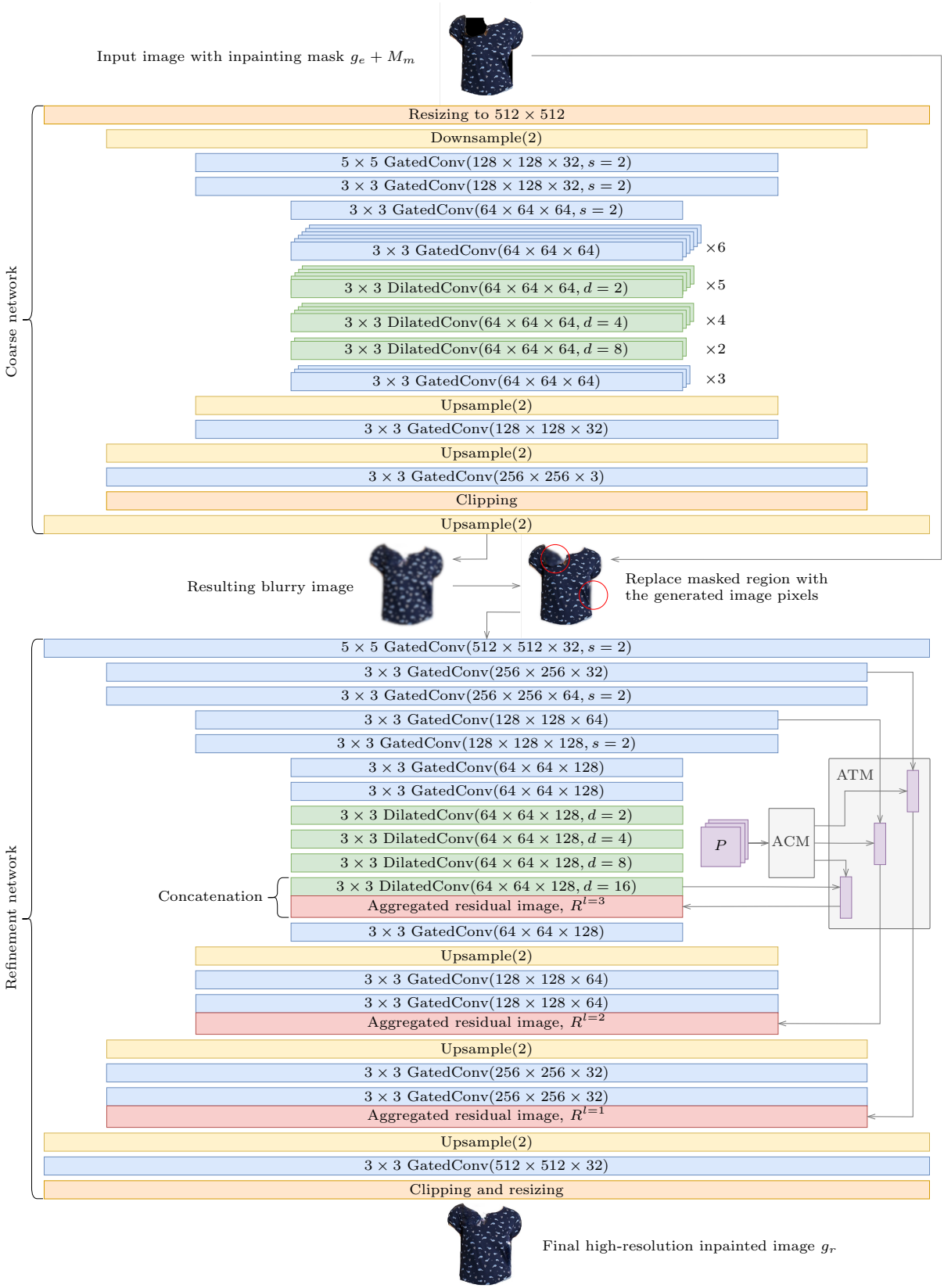

Figure 4.9: Inpainting generator network architecture. Both the coarse (top) and the refinement (bottom) networks are shown, as well as the ATM and ACM modules. The notation is read as $n \times n$ ConvType($m \times m \times c, s, d$), which translates to an $n \times n$ kernel with a stride of s and a dilation of d ($s = 1$ and $d = 1$ if omitted), and a feature map with dimensions $m \times m$ and c channels. The ConvType is either a gated or dilated convolution. All layers use 'same' padding. Upsample(n) means the dimensions are increased by a factor of n, likewise for downsampling being decreased by a factor of n.

the estimated region over the masked regions. Subsequently, this blended version serves as the input to the refinement network.

The refinement network employs a contextual attention mechanism, which uses high-level feature maps to compute attention scores, and then transfers this attention across multiple lower-level feature maps. This allows it to integrate and borrow contextual information from varying levels of abstraction. Both stages of the network incorporate dilated, or atrous, convolutions, augmenting the receptive field even further. For computational agility, the inpainting network is constructed to be lean yet deep.

All convolutional layers use the Exponential Linear Unit (ELU) activation function, which in contrast to Rectified Linear Units (ReLUs), allows negative values which push mean unit activations closer to zero. This is similar to batch normalisation, yet it is less computationally intensive. The shift of the mean towards zero enhances the learning process by aligning the normal gradient more closely with the unit's inherent gradient, owing to a diminished effect of bias shift (Clevert et al., 2016). All layers also use 'same' padding, which ensures that the output feature map is of the same width and height as the input feature map. This is especially useful when designing architectures where you want to preserve spatial dimensions through layers or when stacking multiple convolutional layers. Based on the work of Iizuka et al. (2017), batch normalisation layers are removed to improve colour coherency. Both the coarse and refinement networks have a clipping layer where the pixel values are ensured to be within the allowable range of 0 to 255. This layer in the refinement network also resizes the image to the original dimensions and removes the excess white pixels added before the coarse network, resulting in the final garment g_r.

Additionally, throughout the generator's layers, Light Weight Gated Convolutions (LWGCs) are employed. Gated Convolutions (GCs) for inpainting were introduced by Yu et al. (2019a) and they combine convolutional networks with a gating mechanism. A GC dynamically selects which features to pass forward by using a convolution to produce features and another convolution, followed by a sigmoid activation, to produce gates. The features and gates are then multiplied, allowing the network to emphasise or suppress certain features based on the gate values.

Traditional GCs stand out in addressing the challenges of irregular hole inpainting. However, one of their drawbacks is the computational requirements and associated parameters when compared to standard convolution. To address this, Yi et al. (2020) introduce LWGCs, which reduce the parameter count and computational demand, and also ensure efficacy. The fundamental formula for a GC's output is derived from two convolutions: one to compute gates and another for features. The product of the sigmoid-activated gate and the ELU activation function provides the final output. LWGCs allow the network to dynamically select features. With the architecture discussed above, the focus shifts to how the generator network is trained and optimised, which is discussed next.

Network optimisation

The training process for the network focuses on two main objectives: the adversarial loss and the reconstruction loss, both of which are discussed below:

Adversarial Loss: This loss ensures that the generated images are realistic and similar to the natural distribution of the training images. The chosen adversarial loss is the Wasserstein Generative Adversarial Network with Gradient Penalty (WGAN-GP) loss, which is known for stabilising the training of GANs (Gulrajani et al., 2017). It involves the discriminator output for real images $D(x)$, generated images $D(\tilde{x})$, and the gradient penalty, ensuring that the gradients of the discriminator are constrained to a certain norm. The mathematical representation is

given as:

$$L_d = E_{\tilde{x}\sim P_g}[D(\tilde{x})] - E_{x\sim P_r}[D(x)] + \sigma E_{\hat{x}\sim P_{\hat{x}}}[\|\nabla\hat{x}D(\hat{x})\|_2 - 1]^2. \tag{4.5}$$

Here, D is the discriminator output, G is the generator output; $x, \tilde{x}$, and $\hat{x}$ are real, generated, and interpolated images, respectively. P_r, P_g and $P_{\hat{x}}$ are their corresponding distributions, while E is the expected value. Moreover, $E_{\tilde{x}\sim P_g}[D(\tilde{x})]$ is the expected value of the discriminator scores over the generated images, while $E_{x\sim P_r}[D(x)]$ is the expected value of the discriminator scores over the real images. The term $\sigma E_{\hat{x}\sim P_{\hat{x}}}[\|\nabla\hat{x}D(\hat{x})\|_2 - 1]^2$ is the weighted gradient penalty, which is crucial for enforcing the Lipschitz constraint required by the Wasserstein loss (Gulrajani et al., 2017). The gradient penalty ensures that the gradients of the discriminator with respect to its input $\hat{x}$ do not become too large, which helps in stabilising the GAN training by avoiding mode collapse and ensuring a smoother gradient flow. Further to this, the adversarial loss for the generator is defined as:

$$L_{\text{adv}} = -E_{\tilde{x}\sim P_g}[D(\tilde{x})]. \tag{4.6}$$

Reconstruction Loss: This loss ensures that the generated image closely matches the original image, especially in regions that were not masked. There are two parts to this loss: The in-hole loss $L_{\text{in-hole}}$, which focuses on the areas of the image being inpainted or reconstructed, and the context loss L_{context}, which focuses on the regions surrounding the mask, ensuring that the inpainting is contextually coherent with the rest of the image, and they are as below:

$$L_{\text{in-hole}} = |G(x, m) - x| \cdot m, \tag{4.7}$$

$$L_{\text{context}} = |G(x, m) - x| \cdot (1 - m), \tag{4.8}$$

where m is the binary mask, and $\cdot$ is the dot product. The overall reconstruction loss is a weighted sum of the above two, with specific coefficients α_1 and α_2 assigned to each term:

$$L_{\text{rec}} = \alpha_1 L_{\text{in-hole}} + \alpha_2 L_{\text{context}}. \tag{4.9}$$

Yi et al. (2020) set $\alpha_1 = 1$ and $\alpha_2 = 1.2$. The coarse network is trained solely using this reconstruction loss, L_{rec}, while the refinement network is trained for both the reconstruction loss and the adversarial loss L_{adv}.

Overall, the coarse network and the refinement network are trained simultaneously. The final loss used for training the generator combines the reconstruction loss with the adversarial loss:

$$L_g = L_{\text{rec}} + \beta L_{\text{adv}}. \tag{4.10}$$

The coefficient β determines the relative importance of the adversarial loss in the combined loss. Yi et al. (2020) train the network with $\beta = 10^{-4}$. Following this training technique takes advantage of the adversarial framework to produce more realistic results.

In conclusion, by leveraging sophisticated architectures such as the one discussed, we can generate realistic and high-fidelity inpainted garments. Adding an inpainting layer to the overall network increases the realism and negates the shortfalls of previous methods which cited information loss due to hair-garment or body-garment conflicts (Ishikawa and Ikenaga, 2022). These inpainted garments, which are extracted from an image of a person wearing them, will be used in the next stage of the process, the virtual try-on model, which is discussed in the section that follows.

4.3 Virtual try-on model

With the garment extraction and inpainting architectures discussed in the previous sections, we shift our focus to the architecture of the virtual try-on synthesis model. This model is based on HR-VITON (Lee et al., 2022), which is an improvement on VITON-HD (Choi et al., 2021) and the pioneering work of VITON (Han et al., 2017). An overview and the results of these models are discussed in Sections 2.2 and 2.3; the technical details are discussed here, along with how they can be applied to this study.

Because HR-VITON is an improvement on VITON-HD and written largely by the same authors, the details of common components in the models are not discussed in the paper on HR-VITON. These details, however, are discussed in the VITON-HD paper. Therefore, both the work of Lee et al. (2022) and Choi et al. (2021) are used to create a final model for the virtual try-on image synthesis.

This final model is comprised of three main parts: the first is a pre-processing step, which gets the data into the correct form to parse to the remaining phases of the model. The second is a try-on condition generator, which performs the garment warping and creates the segmentation map of the target person wearing the target garment. Lastly, the try-on image generator, which synthesises the final virtual try-on image, is discussed. As before, each of these unique model parts is discussed in a subsection, starting with the pre-processing step.[30]

4.3.1 Pre-processing

As outlined in Section 2.3.1, the model needs to first create a clothing-agnostic representation of the target person image I_p. This is done to reduce the misalignment that occurs in the final image due to residual garment information left by the garment originally worn by the target person.

The overview of the pre-processing step's input and output for the virtual try-on model can be seen in Figure 4.10. Here, the target person image is shown, along with the resulting clothing-agnostic target person image I_a, the clothing-agnostic segmentation map S_a, and the dense correspondence D_a. As highlighted in Section 2.3.1, removing all traces of clothing in the targeted region (the torso in this case) reduces the chance for misalignment to occur in the final image due to residual garment information. The arms are also removed as they too can introduce information that leads to misalignment in the final image (Choi et al., 2021). In the resulting person image and segmentation map, one can see all traces of the target garment have been thoroughly removed, making them truly clothing-agnostic.

To complete this pre-processing step, three models are utilised: the first model is PGN, which is used to generate the segmentation map, S_p, of the target person. PGN has been discussed in Section 4.2.1, and it remains unchanged for its application here. The only difference between the application in the garment extraction model and its application here in the virtual try-on model is that here, the target person image, I_p, is parsed to PGN, while in the garment extraction model, the image of the person wearing the target garment, I_g, is parsed.

The second model used is OpenPose (Cao et al., 2019), which is used to generate the pose map of the target person, P_p, with the relevant pose key points. OpenPose is a widely used and state-of-the-art human pose detection and estimation tool. Both HR-VITON and VITON-HD use OpenPose as is as an *off-the-shelf model*, meaning they do not train the model themselves, nor update the weights or functions of the model. As we are using HR-VITON as is, we too use the OpenPose model off-the-shelf, negating the need to discuss the architecture. This introduces

[30]The literature review sections on these models will be extensively referenced here, with the reader expected to refer back to them for the sake of non-repetition

Figure 4.10: Pre-processing step for virtual try-on model showing the target person and the resulting clothing-agnostic person image, segmentation map, and dense correspondence as found by using PGN, OpenPose and Densepose, respectively.

a 'black box' element to the model, however, this is common practice in the case of very large models, particularly in image processing.[31]

Lastly, Densepose (Güler et al., 2018) is used to create the dense correspondence, D_a, from the target person. This output is particularly helpful in generating the warped garment, as the dense correspondence allows for accurate deformation. Unlike traditional pose estimation techniques (such as OpenPose) that only identify key points such as joints, Densepose provides a dense correspondence from every pixel in the 2D image to a 3D surface model of the body. This means it can accurately map the full shape and contours of the body, including areas that are not joints like the torso and thighs. Once the dense correspondence has been established, the clothing item can be realistically deformed to match the pose and shape of the person in the image. This can help simulate how the clothing will look and fit on the person's body, and it will improve the accuracy of the virtual try-on. Densepose is used as an off-the-shelf model, with its architecture not being discussed here.[32]

The outputs of this pre-processing step are injected into the remaining two models at different times. The clothing-agnostic person representation and the dense correspondence images are used in both the try-on condition generator and the try-on image generator models, while the clothing-agnostic segmentation map is used only in the try-on condition generator model, which is discussed next.

4.3.2 Try-on condition generator

In the foundational work of VITON-HD, this process was two separate models, where the garment deformation and target segmentation map generation are independent of each other. This means the models have no information exchange between them, however, in HR-VITON, these try-on conditions are generated simultaneously, which is the approach followed here. The output of this model is a semantic segmentation map of the target person wearing the target garment, S_p, and a warped garment, g'_r which is deformed to fit the shape of the target person's body.

When created separately, misalignment occurs as there is information loss between the segmentation map generation and the garment warping. This loss is negated by simultaneously creating the warped garment and the segmentation map, in one model with information exchange. The architecture of the try-on condition generator network can be seen in Figure 4.11, which shows a condensed version of the network, using the same notation as before. All convolutional layers are 3×3 with a stride of 1, are undilated, and use 'same' padding. Both the Feature Fu-

[31] OpenPose is a highly complex model, however, it is widely used off-the-shelf not just in virtual try-on use cases, but other image processing models too. The code and overview can be found here.

[32] A more thorough explanation of Densepose and the source code can be found here.

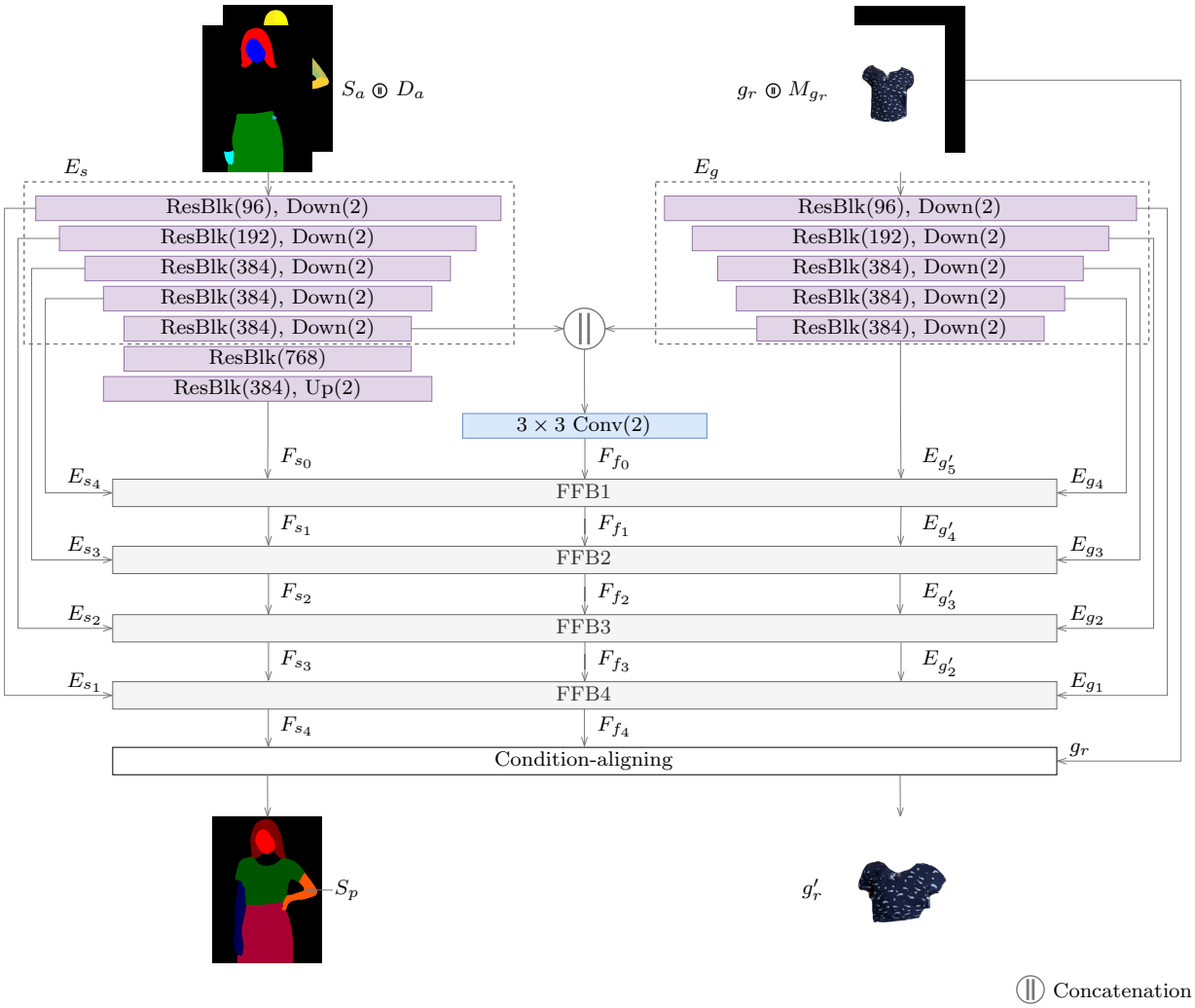

Figure 4.11: Try-on condition generator architecture showing the simultaneous segmentation map generation and garment warping. The two submodels (encoders), the segmentation encoder, E_s, and the clothing encoder, E_g, are represented. The inputs for each submodel are shown, as well as the final output of the try-on condition generator model. The notation is read as before, with ResBlk(m) indicating a residual block architecture with m channels. Down(n) translates to downsampling by a factor of n, likewise for the upsampling. FFB stands for feature fusion block. All convolutional layers are undilated, using a stride of 1 and a filter of 3×3 with padding. Conv(m) means a convolutional layer with m channels.

sion Block (FFB) and the condition-aligning layers are expanded in their architecture diagrams later.

The try-on condition generator consists of two encoders as submodels, and one decoder that has multiple outputs. The segmentation encoder, E_s, takes the clothing agnostic segmentation map, S_a, and the dense correspondence, D_a, as input. The garment encoder, E_g, takes the inpainted garment and its subsequent binary mask, g_r and M_{g_r} respectively, as input. Both encoders have the same structure, using five Residual Blocks (ResBlks), all of which are followed by a downsampling layer where the resolution is halved.

A ResBlk is a neural network architectural unit comprising several layers, including convolutional ones, which transform an input tensor.[33] Parallel to these transformation layers, a 'skip' or

[33] A tensor is a multidimensional array used to store and handle complex data structures.

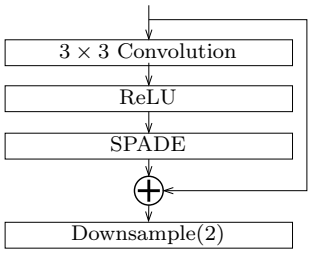

Figure 4.12: Residual block with SPADE normalisation architecture, followed by a downsampling layer, as used in the try-on condition generator.

'shortcut' connection directly channels the input tensor to the block's output, sidestepping the intermediary layers. When the input from this shortcut connection merges with the output from the transformation layers, they are typically added together. Such an architecture ensures that the input's original information is retained. Furthermore, it addresses challenges such as the vanishing gradient problem, thereby facilitating the training of deeper neural networks. All of the ResBlks in this context utilise Spatially Adaptive Denormalisation (SPADE) normalisation layers, as introduced by Park et al. (2019).

Unlike other normalisation methods, such as batch normalisation or instance normalisation, which normalise feature maps to have a mean of zero and a standard deviation of one, SPADE normalises the feature maps in a spatially adaptive manner. Here, SPADE uses a spatial map (such as a segmentation map) as additional input to modulate the normalisation parameters (both the scale and the bias) for each spatial location. This allows the normalisation to adjust to different regions of the image based on the provided spatial information. As a result, SPADE ensures that the normalisation process retains semantic spatial information which is crucial for virtual try-on image synthesis. The architecture of the ResBlks using SPADE can be seen in Figure 4.12.

The output features of the last ResBlk in both E_s and E_g, denoted as E_{s_5} and E_{g_5} respectively, are concatenated and parsed to a 3×3 convolutional layer whose output is then parsed to the first FFB of the decoder (FFB_1) as F_{f_0}. The output of E_g at the last layer (the fifth layer) is parsed directly into FFB_1 as $E_{g'_5}$. Additionally, E_s is followed by two ResBlks, and an upsampling layer before being parsed to the decoder at FFB_1. From each of the first four layers in E_s and E_g, a feature pyramid is extracted and injected into the related FFB. Here, the extracted feature pyramids are represented as $\{E_{s_{5-i}}\}_{i=0}^{4}$ and $\{E_{g_{5-i}}\}_{i=0}^{4}$ from E_s and E_g respectively. $E_{s_{5-i}}$ and $E_{g_{5-i}}$ are then parsed to FFB_i. The FFBs are discussed next.

Feature fusion blocks

The decoder network consists of four FFBs, and a condition-aligning block. The FFBs take the output from the previous step and the extracted feature pyramids, $E_{s_{5-i}}$ and $E_{g_{5-i}}$, as input to fuse the feature maps to predict the appearance flow map, F_{f_i} and the segmentation feature map, F_{s_i}. This is done through a series of layers, including warping, convolutional layers, ResBlks, and upsampling. The architecture of an FFB can be seen in Figure 4.13.

Each FFB is compromised of three main pathways: the first is the segmentation pathway, F_s, which stems from E_s; the second is the flow pathway, F_f, derived initially from the concatenation of the two encoders, and the previous FFB; and lastly, the garment pathway, $E_{g'}$, derived from the garment encoder, E_g. The initial segmentation pathway, F_{f_0}, comes from the encoder E_s, whose output is parsed through another two ResBlks, and an upsampling layer. $F_{s_{i-1}}$ is then concatenated with $E_{s_{5-i}}$, which is the feature pyramid from layer $5-i$ in the segmentation encoder. This concatenation, along with the result of the warping module, is parsed to a ResBlk

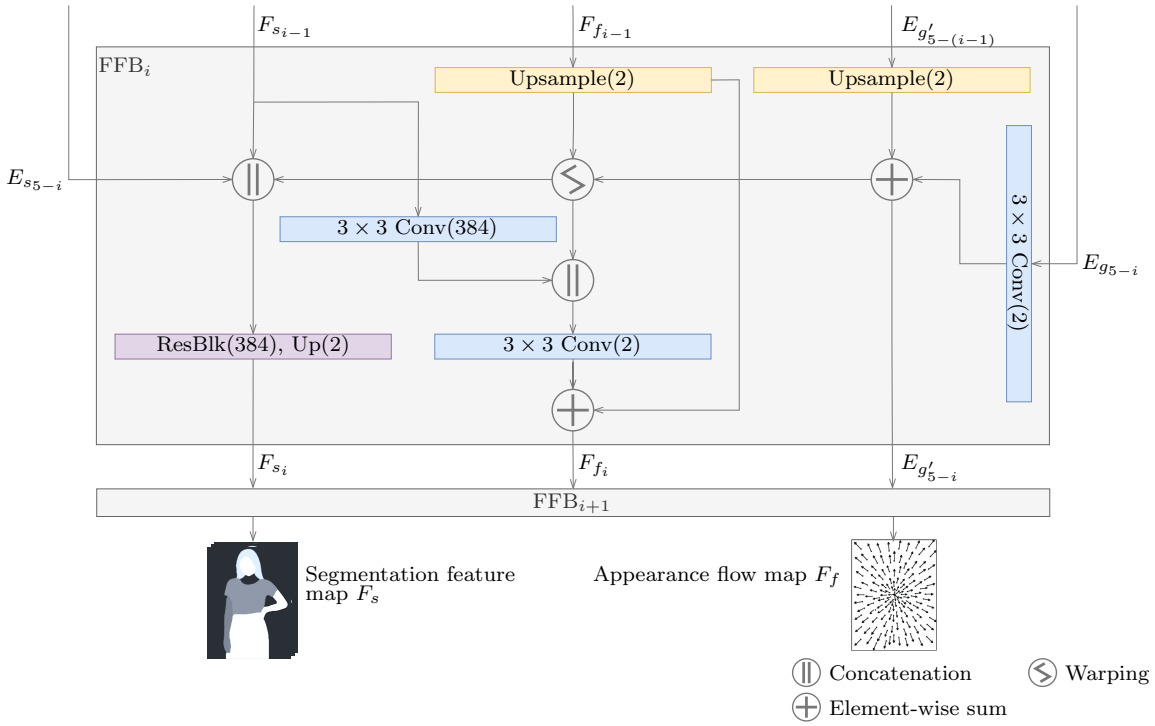

Figure 4.13: Feature fusion block architecture showing the appearance flow map, F_f, and the segmentation feature map, F_s, generation with information exchange. FFB$_i$ is shown with input from FFB$_{i-1}$ and output to FFB$_{i+1}$, as well as the output from both $E_{g_{5-i}}$ and $E_{s_{5-i}}$ which are injected into the FFB at various levels. Examples of the appearance flow map and the segmentation feature map as output are shown.

and an upsampling layer, which is subsequently parsed to FFB$_{i+1}$ as F_{s_i}. $F_{s_{i-1}}$ also directly goes to a convolutional layer, whose output is also concatenated with the warping module's output.

Similarly, the feature pyramid extracted from layer $5-i$ in the garment encoder is injected into FFB$_i$ as $E_{g_{5-i}}$, which undergoes a convolution. The result of this is added to the upsampled output of garment flow $E_{g'_{5-(i-1)}}$, which is subsequently parsed to the next FFB as $E_{g'_{5-i}}$. $E_{g'_5}$, the initial garment flow, is the direct output of the garment encoder. In addition to being parsed to FFB$_{i+1}$, $E_{g'_{5-i}}$ is also an input into the warping module.

The warping module is responsible for deforming the target garment to fit the target person's pose and shape. To do this, an appearance flow map is used. A flow map is merely a vector representation of each pixel's displacement, or movement. Moreover, in the flow map, each pixel is represented by a vector (u, v), which denotes the displacement of each pixel in the horizontal (u), and the vertical (v) direction, and it is measured in pixels. For example, if a pixel has a vector representation of $(-1, 3)$, that pixel will move one pixel to the left, and 3 pixels up in the warped image. The appearance flow map is found by considering the dense correspondence of the target person, and the target garment. Generally, the warping process can be summarised as follows, where for each pixel in the source image, we perform the following steps:

1. Consider the source image as a set of pixels, denoted as (x, y), where x represents the horizontal direction and y the vertical direction.
2. Retrieve the flow vector (u, v) from the generated flow map.

3. Compute the warped pixel location using:

$$x_w = x + u,$$
$$y_w = y + v,$$

   where x_w represents the pixel x in the warped image, likewise for y_w.

4. Interpolate the colour value from the source image at (x_w, y_w). This is done since these values might be non-integers.

5. Set the pixel value of the output image at (x, y) to the interpolated value.

6. Repeat for all pixels, handling boundary cases by manually overriding the vector to ensure no pixel is out of bounds.

The flow map has the same dimensions as the input image, and it consists of two channels, one for the horizontal displacement, u, and the other for the vertical displacement, v. As inputs to the warping module, $E_{g'_{5-i}}$ and an upsampled $F_{f_{i-1}}$ are used. F_{f_0} is found through the concatenation and convolution of the encoder's output, and $\{F_{f_{i-1}} \mid i > 1\}$ is based on the output of the previous FFB. The output of the warping module is concatenated with $E_{s_{5-i}}$ and $F_{s_{i-1}}$, as well as with the $F_{s_{i-1}}$ which went through a convolutional layer.

To generate the appearance flow map output of each FFB, denoted as F_{f_i}, the upsampled $F_{f_{i-1}}$ is added to the 2-channel output of a convolutional layer, which takes the warping module's output, and the convolved output of $F_{s_{i-1}}$ as a concatenated input. The last FFB in the decoder is slightly different from the first three in that it has no garment flow output, it only has the segmentation feature map, F_s, and the appearance flow map, F_f as output, as can be seen in Figure 4.11. The output of FFB_4 is F_{s_4}, or S_{raw}, and the final appearance flow map F_{f_4}. Both of these are used as inputs into the condition-aligning layer, which is discussed next.

Condition-aligning

As a final layer in the try-on condition generator, condition-aligning occurs. Here, the misalignment between the generated segmentation map of the target person wearing the target garment and the warped garment is removed. This is done by removing the non-overlapping regions of the warped garment mask and the extracted garment channel from S_{raw}. Consider Figure 2.7 in Chapter 2, which shows misalignment between two garments. Removing this misalignment reduces the chance of artefacts appearing in the final virtual try-on image. The architecture of the condition-aligning layer can be seen in Figure 4.14.

The condition-aligning layer has three inputs, the two outputs of FFB_4 (S_{raw} and F_{f_4}), and the inpainted garment g_r. The predicted appearance flow map, F_{f_4}, is first upsampled to the final resolution of 768×1024, and it is then used in the warping module along with the inpainted garment g_r. The warping module in this layer is as before, where a two-channel flow map is used to warp the garment image so that it accurately represents what the garment will look like on the target person. This results in a warped garment image g_r^w, from which a binary mask is found, which is denoted as $M_{g_r}^w$.

The final segmentation feature map predicted in FFB_4, S_{raw}, consists of multiple channels and it is used as an input for the condition-aligning layer. However, not all the channels are needed concurrently, only the channels that are needed at each stage of the condition-aligning process are extracted. Moreover, the feature map can be represented by $S_{\text{raw}}^{c,i,j}$, where c represents the channel, and i and j are indices across the spatial dimensions. The garment channel can be represented by $S_{\text{raw}}^{c=G,i,j}$, which is extracted from S_{raw} and multiplied, pixel-wise, with $M_{g_r}^w$. This results in the final garment channel for the segmentation map. Doing this removes any misalignment between the predicted segmentation map and the warped garment.

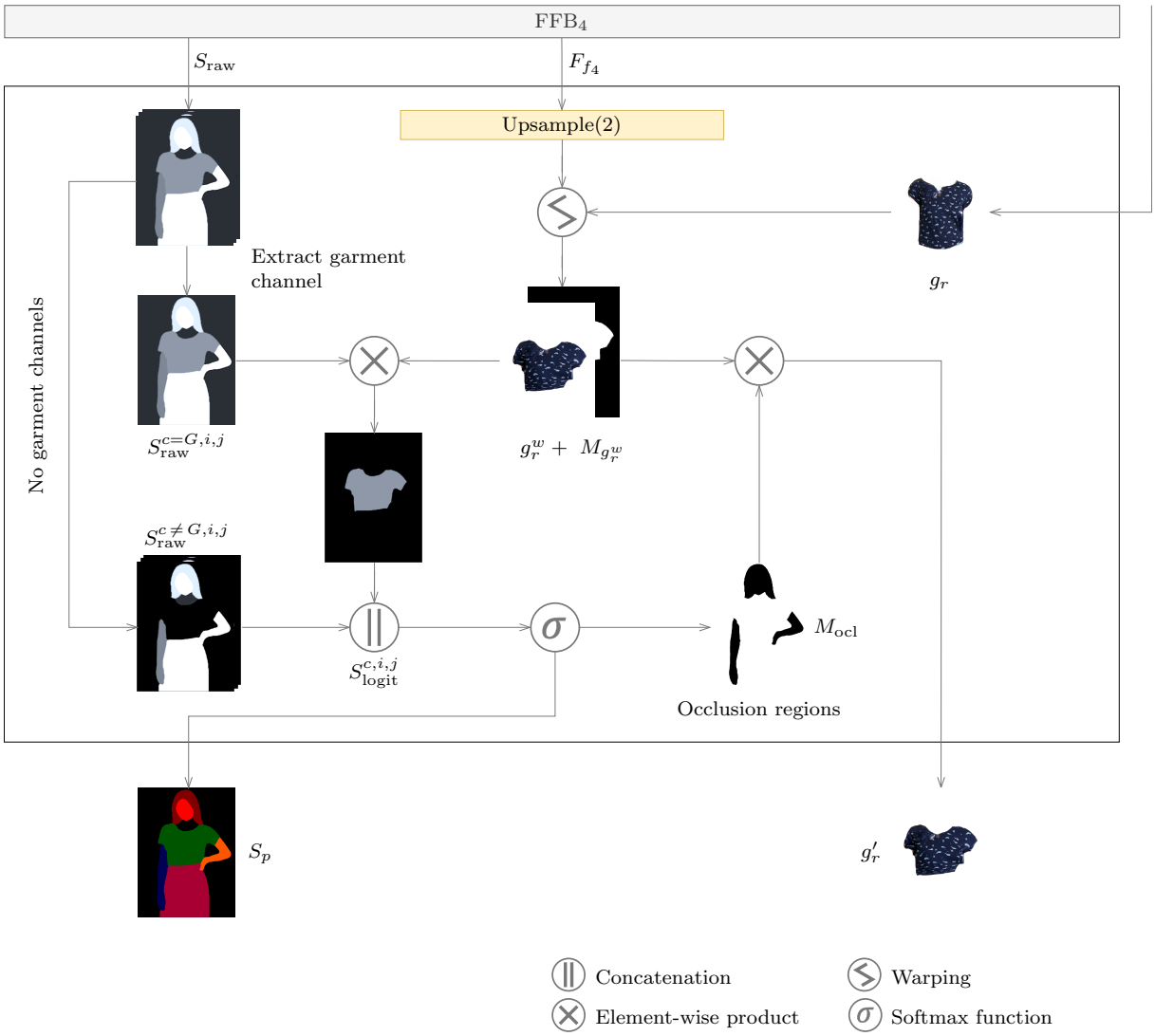

Figure 4.14: Condition-aligning layer architecture showing how misalignment is removed, as well as how occlusions are removed. The condition-aligning layer takes the outputs from FFB_4, S_{raw} and F_{f_4}, as input, as well as the original garment, g_r, and mask, M_{g_r}. The segmentation feature map output from FFB_4 is renamed from F_{s_4} to S_{raw} for the condition-aligning module. The warping module is as in the FFBs.

All non-garment channels, $S_{\text{raw}}^{c\neq G,i,j}$, are extracted from S_{raw} and added back with the garment channel $S_{\text{raw}}^{c=G,i,j}$ after the misalignment has been removed. This results in a raw, or unnormalised output for each channel, denoted as $S_{\text{logit}}^{c,i,j}$. In the context of a multi-channel feature map, as we are dealing with, *logit* refers to the unnormalised output before an activation function is applied. This can be represented, for all channels, as:

$$S_{\text{logit}}^{c,i,j} = \begin{cases} S_{\text{raw}}^{c,i,j} \cdot M_{g_r}^w; & c = G \\ S_{\text{raw}}^{c,i,j}; & c \neq G \end{cases},$$
$$S_p = \sigma(S_{\text{logit}}^{c,i,j}).$$

The results from this channel-wise operation are then concatenated, after which a depth-wise softmax function, σ, is applied, resulting in the final segmentation map, S_p. For each pixel (i, j), we have c *logits*, one from each channel. The softmax activation function is applied to these to get a probability distribution across the channels. This will tell us how likely each channel, and

subsequently, each colour in the segmentation map is for each pixel. In this way, we can generate the final semantic segmentation map of the target person wearing the target garment.

As an additional step to remove artefacts introduced by 'pixel squeezing', which is caused by excessive warping of the garment near body-garment or hair-garment conflict, occlusion handling is proposed. To do this is relatively straightforward; from S_p, only the hair, arms, and head regions are kept, which are converted to a binary mask, M_{ocl}. This binary mask is then multiplied, pixel-wise, with the warped garment, g_r^w, to ensure any pixels that are to be occluded in the final image are removed from the garment. Doing this ensures that the model does not try to include the occluded pixels in the final image, thus removing the so-called 'pixel squeezing' artefacts. This results in the final garment generated by the try-on condition generator network, which is represented by g_r'. This, along with the predicted segmentation map, S_p, are the outputs of the try-on condition generator model.

Following Lee et al. (2022), we employ a discriminator rejection method to filter out any low-quality predicted segmentation maps at test time. This is discussed in-depth in Section 2.3.6, however, for completeness' sake, it will be summarised here. Low-quality predicted segmentation maps, S_p in our case, are filtered out based on the probability a pre-trained classifier assigns to the correct class. Here, because we train the model in a paired manner, the real data's distribution is generated from the ground truth segmentation maps, $\hat{S}_p$, while the implicit distribution is derived from the predicted segmentation maps, S_p. The acceptance probability for a generated image, S_p, is defined concerning the distributions of the real data and the try-on condition generator's implicit distribution. The optimal discriminator is learned to represent this acceptance probability. S_p is only accepted if its corresponding acceptance probability surpasses a specified threshold. Using this approach allows us to discard erroneous segmentation maps, enhancing the network's overall performance. Now, with the architecture of the try-on condition generator network having been explained, we shift our focus to the loss functions used to optimise the model, which is discussed next.

Network optimisation

The training process for virtual try-on networks, which is integral to understanding the associated loss functions, is elaborated upon in Section 4.4. To briefly explain, the network employs a paired training strategy wherein the target garment is extracted from the target person image and then synthesised back onto the same person. Doing this allows us to have a ground truth for the network to use, and this is the method employed by Han et al. (2017), Choi et al. (2021), and Lee et al. (2022).

Using the paired images, the try-on condition generator network uses the pixel-wise cross-entropy loss, which measures the difference between the predicted segmentation map, S_p, and the ground truth segmentation map $\hat{S}_p$. The goal is to minimise the discrepancy between the two. Specifically, the pixel-wise cross-entropy loss measures the dissimilarity between predicted class probabilities and true labels for each pixel in the image. It is used in segmentation tasks to ensure accurate and detailed predictions by comparing the true class and the predicted distribution at a pixel level. Here, this loss is denoted as L_{CE}, and it is calculated as follows:

$$L_{CE} = -\frac{1}{N}\sum_{n=1}^{N}\sum_{k=1}^{K} y_{n,k}\log(p_{n,k}), \tag{4.11}$$

where N is the number of pixels in the image, K is the number of classes, or segments, $y_{n,k}$ is a binary indicator if class label k is correct for pixel n, and $p_{n,k}$ is the predicted probability that pixel n belongs to class k. Here, for each pixel and class, the formula calculates the log loss for that class and sums them, and then the loss is averaged across all the pixels. The goal is to minimise this loss.

To encourage the network to warp the garment to fit the target person's pose and body, L_1 loss, and perceptual loss, L_{VGG}, are used. These loss functions are applied to the intermediate output to aid in mitigating vanishing flow maps and to improve overall performance. L_1 is defined as:

$$L_1 = \sum_{i=0}^{3} \beta_i \cdot ||\mathcal{W}(M_{g_r}, F_{f_i}) - M_{g'_r}||_1 + ||\hat{M}_{g'_r} - M_{g'_r}||_1.$$

In this equation, $\mathcal{W}(M_{g_r}, F_{f_i})$ is the result of the warping module, and it is equivalent to $M_{g'_r}$. $\hat{M}_{g'_r}$ is the ground truth of the binary mask of the warped and occlusion-handled garment. This measures the absolute difference between the predicted and actual values for the binary mask of the garment and tries to get the predicted values as close to the actual values as possible. β_i determines the relative importance of each term in each step i.

The perceptual loss, L_{VGG}, developed by Johnson et al. (2016), uses a pre-trained VGG19 network (Simonyan and Zisserman, 2015) to extract feature maps. This loss computes the difference between feature maps extracted for both predicted and target images, emphasising perceptual similarity over pixel-level accuracy. It aims to capture structural and perceptual nuances, ensuring images appear visually similar to human observers. Here, the predicted warped garment image after occlusion-handling, g'_r, is used. The ground truth version of this image, $\hat{g}'_r$, is the garment extracted from the target person image. Formally, this loss can be defined as:

$$L_{\text{VGG}} = \sum_{i=0}^{3} \beta_i \cdot \phi(\mathcal{W}(g_r, F_{f_i}) - g'_r) + \phi(\hat{g}'_r - g'_r),$$

and as before, $\mathcal{W}(g_r, F_{f_i})$ is the result of the warping module and it is equivalent to g^w_r. ϕ is the function that uses the pre-trained VGG19 network to extract features, and β_i determines the relative importance of the terms in the i^{th} layer. The perceptual loss attempts to ensure that the predicted image not only looks visually similar to the target image but also has similar high-level features.

Although both the L_1 loss and the L_{VGG} loss are related to the garment and its warping, the L_1 loss uses the binary masks as it is focused on minimising the difference between the predicted and actual mask's position and shape. The perceptual loss, L_{VGG}, uses the warped garment images and attempts to minimise the difference in the appearance of the predicted warped garment and its ground truth, focussing on features, colours, textures, and patterns.

To enforce the smoothness of the final appearance flow map (F_{f_4}) a total-variation loss (L_{TV}) is applied. L_{TV} is found by regularising the final appearance flow map, and it is important in the flow estimation at coarse scales.[34] This total variation loss can be described as:

$$L_{\text{TV}} = ||\nabla F_{f_4}||_1. \tag{4.12}$$

The gradient, ∇F_{f_4}, measures the rate of change or variation in the appearance flow. The L_1 loss in the context of L_{TV} measures the absolute differences between neighbouring pixel values, and it penalises large variations in adjacent pixel values, ensuring a smooth appearance flow map. Lee et al. (2022) empirically found that regularising only the final flow map yielded more desirable results.

[34]When analysing images at coarse scales, finer details might be ignored in favour of broader structures or patterns, which is desired in a flow map — there are no fine details, however, patterns and structures of the flow vectors are important.

Finally, because the network is trained in an adversarial manner, as described in Section 2.1, we can minimise the conditional GAN loss between $\hat{S}_p$ and S_p.[35] This loss, denoted as L_{cGAN}, measures the difference between the predicted segmentation map S_p and the ground truth segmentation map $\hat{S}_p$. The goal here is to train the network to predict the segmentation map, S_p, to be as close to the ground truth, $\hat{S}_p$, as possible. We use the least-squares Generative Adversarial Network (GAN) loss, as proposed by Mao et al. (2017), which provides a more stable training process and helps reduce common GAN loss issues, such as mode collapse. It also results in generating higher-quality images. Formally, where 'TOCG' represents the 'Try-On Condition Generator', this loss can be defined as follows:

The set of input conditions is represented by:

$$C_{\text{TOCG}} = \{S_a, D_a, g_r, M_{g_r}\}.$$

The set of conditions is pivotal in the conditional GAN loss, as they are what guides the network, and it differentiates the conditional GAN loss from the conventional GAN loss.

The discriminator loss, $L_{D_{\text{TOCG}}}$, is found by:

$$L_{D_{\text{TOCG}}} = \frac{1}{2}[E(D(C_{\text{TOCG}}, \hat{S}_p) - 1)^2 + E(D(C_{\text{TOCG}}, S_p)^2)], \tag{4.13}$$

where the discriminator, D, takes two inputs: the set of conditions for the try-on condition generator, C_{TOCG}, and either the ground truth segmentation map, $\hat{S}_p$, or the generated segmentation map, S_p. The expected value is denoted as E. The generator loss, $L_{G_{\text{TOCG}}}$, is calculated as:

$$L_{G_{\text{TOCG}}} = \frac{1}{2}E(D(C_{\text{TOCG}}, S_p) - 1)^2. \tag{4.14}$$

As in Equation (2.1), these two losses are combined, resulting in the following:

$$L_{\text{cGAN}_{\text{TOCG}}} = \max_D \min_G (L_{D_{\text{TOCG}}} + L_{G_{\text{TOCG}}}). \tag{4.15}$$

Combining these separate loss functions into one overarching function to be applied to the network results in the following:

$$L_{\text{TOCG}} = \beta_{\text{CE}} L_{\text{CE}} + \beta_{L_1} L_1 + \beta_{\text{VGG}} L_{\text{VGG}} + \beta_{\text{TV}} L_{\text{TV}} + \beta_{\text{cGAN}} L_{\text{cGAN}}. \tag{4.16}$$

The try-on condition generator is trained end-to-end using this objective function. The β terms are hyperparameters that control the relative importance between the different losses. β_{cGAN} and β_{L_1} are set to 1, while β_{CE} and β_{VGG} are set to 10, and β_{TV} is set at 2.

This try-on condition generator network is built and optimised to predict the warped garment to be placed on the target person, as well as the segmentation map of the target person wearing this warped garment. The two outputs of the network are predicted simultaneously through an encoder-decoder network structure which allows for information exchange — this results in more cohesive and realistic results. The two outputs from the try-on condition generator are used as part of the input into the final stage of the model, the try-on image generator, which is discussed in the next section.

[35] Conditional GANs are a variant of GANs where both the generator and discriminator are conditioned on some external information. This makes the generated data not just random but controlled in some way, dependent on the input conditions.

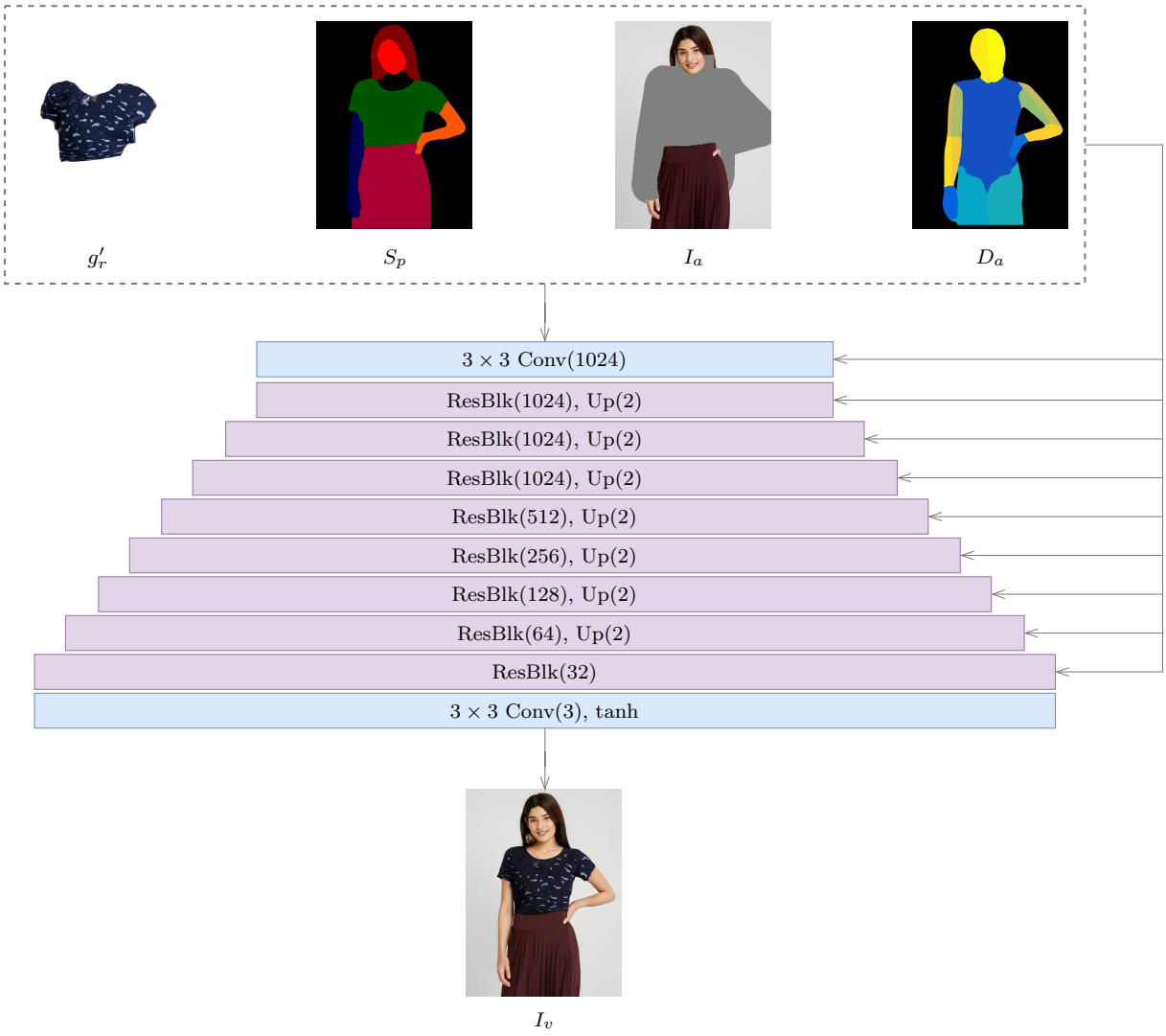

Figure 4.15: Try-on image generator network architecture showing the input images and the resulting virtual try-on image, I_v. The input images are the warped inpainted garment, g'_r, the predicted segmentation map, S_p, the clothing-agnostic person image, I_a, and the dense correspondence, D_a. The notation is as before, with tanh representing the hyperbolic tangent activation function.

4.3.3 Try-on image generator

As a final stage in the virtual try-on network, we take the output from the try-on condition generator, as well as the clothing-agnostic images as inputs into a decoder network to synthesise the virtual try-on image, where the target person is wearing the target garment. Here, in the try-on image generator network, we use a simple decoder architecture made up of two convolutional layers, and eight ResBlks layers with the same structure as before in the try-on condition generator. The convolutional layers in this network use spectral normalisation, and this is done primarily to stabilise the network in training (Lee et al., 2022). The network architecture can be seen in Figure 4.15.

This network has four inputs: the warped inpainted garment, g'_r, the predicted segmentation map, S_p (both of which are the outputs from the try-on condition generator), the clothing-agnostic person image, I_a, and the dense correspondence, D_a. These inputs are injected into every layer of the network, baring the final convolutional layer. The first layer is a 3×3 convolutional layer with 1024 output channels, which is followed by seven ResBlks with varying

numbers of output channels, and each is followed by an upsampling layer where the resolution is doubled. The penultimate layer is a ResBlk without an upsampling layer whose output is parsed to the final 3×3 convolutional layer. The final output layer has three channels, for each colour of RGB, and it is followed by a tanh activation function. The output of this network is a high-resolution image, I_v, where the target person is wearing the target garment.

The tanh activation function is chosen instead of ReLU or similar functions because the convolutional layers in this context use spectral normalisation. ReLU and its variants are non-saturating activations, meaning they do not squash their inputs within a fixed small range. In practice, this often leads to faster convergence because they do not suffer as much from the vanishing gradient problem. However, when using spectral normalisation, the normalisation itself restricts the magnitude of activations, which can make ReLU's non-saturating property less beneficial. With the architecture of the network explained, we shift our focus to how the network is optimised, which is discussed next.

Network optimisation

Similar to the try-on condition generator, the try-on image generator network is trained with multiple loss functions. The training of this network, as before, is done in a pairwise manner, meaning there is a ground truth image to work with. The conditional GAN loss, L_{cGAN}, is used again, along with the perceptual loss, L_{VGG}. The perceptual loss again uses a pre-trained VGG19 network to extract feature maps from the output image at different layers and compares them to those extracted from the ground truth image, $\hat{I}_v$. Formally, this loss can be defined as:

$$L_{\text{VGG}} = \sum_{l=0}^{9} \beta_l \cdot \frac{1}{N_l} \sum_{i=1}^{N_l} (\phi(\hat{I}_v)_i - \phi(I_v)_i)^2. \tag{4.17}$$

This loss is used at every layer in the decoder network, with the weights β_l controlling the relative importance of the l^{th} layer's output. N_l is the number of elements in the feature map of layer l, which is found by multiplying the dimensions of the output by the number of output channels. ϕ is the pre-trained VGG19 function used to extract features from images, and it is applied to both the ground truth image, $\hat{I}_v$, and the predicted image, I_v. Here, this function is used to find the i^{th} element from both $\hat{I}_v$ and I_v.

The conditional GAN loss used here is similar to that used in the try-on condition generator, however, it uses the Hinge loss, which is represented by:

$$\max(0, 1 - D(C_{\text{TOIG}}, \hat{S}_p))$$

for real data, while the loss for fake data can be represented by:

$$\max(0, 1 + D(C_{\text{TOIG}}, S_p)).$$

The Hinge loss aims to maximise the margin of decision between classes, penalising predictions that are either incorrect or not sufficiently confident, thereby encouraging clearer and more decisive classification boundaries. Lee et al. (2022) found that using this loss improved the training stability. Due to its origins in Support Vector Machines (SVMs), the Hinge loss is designed to find the maximum margin separator. This can make the discriminator in GANs more robust to adversarial examples, which could lead to better gradients for the generator.[36]

[36]This refers to the idea that the gradient information the generator receives during training is informative and provides a clear path to improve its sample generation. This is crucial for the convergence and stability of GAN training.

As before, where 'TOIG' represents the 'Try-On Image Generator', there is a set of conditions used, which can be represented by:

$$C_{\text{TOIG}} = \{g'_r, S_p, I_a, D_a\}.$$

Now, using the Hinge loss, as opposed to the least squares loss as before, the discriminator loss, $L_{D_{\text{TOIG}}}$, is calculated as:

$$L_{D_{\text{TOIG}}} = E[\max(0, 1 - D(C_{\text{TOIG}}, \hat{S}_p))] + E[\max(0, 1 + D(C_{\text{TOIG}}, S_p))]. \tag{4.18}$$

The set of conditions, C_{TOIG}, and either the ground truth segmentation map, $\hat{S}_p$, or the generated segmentation map, S_p, are used as the inputs for the discriminator. The generator loss, $L_{G_{\text{TOIG}}}$, is found by:

$$L_{G_{\text{TOIG}}} = -E[D(C_{\text{TOIG}}, S_p)]. \tag{4.19}$$

This leads to a final conditional GAN loss, as described below:

$$L_{\text{cGAN}_{\text{TOIG}}} = \max_D \min_G (L_{D_{\text{TOIG}}} + L_{G_{\text{TOIG}}}). \tag{4.20}$$

The last loss function used to optimise this network is the feature-matching loss, developed by Wang et al. (2018b) to reduce the discrepancy between the intermediate features of the generated image and the real image when passed through a discriminator. This loss can be used since an adversarial training methodology is followed. This loss is particularly useful in ensuring the generated image's features align with those of the ground truth image at various layers. This loss also helps to stabilise the GAN training process and pushes the generated images to occupy the same feature space as the ground truth images. This loss can be defined by the following function, which is relevant to the GAN discriminator:

$$L_{\text{FM}} = \sum_{l=0}^{k} \beta_l \cdot \frac{1}{N_l} \sum_{i=1}^{N_l} (D_l(\hat{I}_v)_i - D_l(I_v)_i)^2. \tag{4.21}$$

The feature-matching loss is related to the perceptual loss, which is why the functions for both are similar. Here, D_l denotes the feature map from layer l of the discriminator D, while there are k layers in D. As before, N_l represents the number of elements in the feature map of layer l, and β_l controls the relative importance of the different layers. Applying this function to each layer in the network encourages the generator to produce images that not only fool the discriminator but also have similar intermediate features as the ground truth images.

Finally, these individual loss functions can be combined to form an overall loss function for the try-on image generator network. This is similar to Equation (4.16), however, there are no cross-entropy, L_1, nor total-variation loss functions. The function can be defined as follows:

$$L_{\text{TOIG}} = \beta_{\text{VGG}} L_{\text{VGG}} + \beta_{\text{cGAN}} L_{\text{cGAN}} + \beta_{\text{FM}} L_{\text{FM}}. \tag{4.22}$$

Again, the β coefficients control the relative importance of each of the respective loss functions. Both β_{VGG} and β_{FM} are set to 10, while β_{cGAN} is set to 1.

This try-on image generator network uses the output conditions generated by the try-on condition generator as input, as well as the clothing-agnostic person image and the dense correspondence. It outputs the final, high-resolution, virtual try-on image, where the target person is wearing the target garment, which has been extracted from another person image and inpainted. This is the final phase of the multistage model, as shown in Figure 4.1. With the architectures of all the models having been discussed, the focus shifts to the training methodology followed in our research, which is covered in the next section.

(a) Ground truth. (b) Result with highlighted differences.

Figure 4.16: Pairwise virtual try-on training overview showing the original ground truth image, and the resulting virtual try-on where the original image is recreated, with the differences between the two highlighted.

4.4 Training methodology

As briefly mentioned in the preceding sections, the network is trained in a pairwise manner, which means that the same image is used for the target garment and the target person. Here, the goal is to recreate the image by virtually trying on the same garment as already worn in the image, which allows us to have a ground truth image to work with. Ideally, an unpaired dataset should be used in training, as this will allow the model to better generalise between people that have very different features, however, this is costly to achieve.[37] As a summary of the training methodology, consider Figure 4.16, which shows the original, or ground truth image, and the virtual try-on image, which is a reconstruction of the original image.

In Figure 4.16b, the obvious differences between the resulting virtual try-on and the original are circled. Both earrings have been distorted, while the text on the shirt has been slightly enlarged and blurred, and the label of the shirt has been excluded. Both sleeves have slightly different shapes from the original, and the neckline has been widened as well. There are also minor differences in the person's collarbone and the inside of their left elbow. These are the differences that the loss functions described above seek to minimise, leading to a more coherent and visually appealing virtual try-on result.

This training methodology is followed by Han et al. (2017) in VITON, Choi et al. (2021) in VITON-HD, and by Lee et al. (2022) in HR-VITON. One of the main reasons for following a training methodology like this is that it represents reality in that at test time, only the target garment, and the target person images will be available — this ensures the model will be able to generalise at test time (Han et al., 2017).

With this in mind, when training the model, the problem becomes given the same image, S_p, for both the target person and the target garment, and the clothing-agnostic representations, how can the network synthesise a perceptually convincing virtual try-on image? Moreover, the model is trained so that it should generalise at test time and be able to synthesise a perceptually convincing image of the target person wearing an arbitrary garment. Further, more technical implementation details of the model can be found in Appendix B. Now that we have explained the training methodology used for the networks and delved into their architectures, concluding

[37] Doing this will require at least double the amount of images, which is highly impractical in a retail setting.

remarks follow before implementing and testing the model.

4.5 Conclusion

In this chapter, we provided an overview of the end-to-end virtual try-on model, expanding on each of the subnetworks and their architectures. We provided a high-level overview of the model, delved into the specifics of the garment extraction and inpainting models, and discussed the virtual try-on models. Additionally, we covered the pre-processing steps needed for the different models. The loss functions used to optimise and train each of the models, where applicable, were explained, as well as the general training methodology.

This methodology is unique in the fact that it combines the high-resolution virtual try-on capabilities of HR-VITON with the practical application of garment extraction and inpainting. Adding the inpainting layer to the garment extraction model is a means to supplement lost information due to occlusions in the image. Inpainting also has an application in improving the synthesised image by altering any corrupted pixels; this has been done by Kubo et al. (2019).

Having discussed the research methodology, we now shift our focus to the model's implementation and testing. The subsequent chapter will detail the implementation and evaluation of the model, utilising the remaining 2,032 images from the test set for benchmarking our model's performance against existing models and discussing the outcomes.

5 | Application and results

Using the methodology outlined in the previous chapter, we now implement the virtual try-on with garment extraction model. A detailed view of the code and implementation can be found here. In this chapter, we evaluate the performance of our model using the metrics discussed in Section 2.1.1. We also compare the performance of our model to the state-of-the-art HR-VITON model, as well as ACGPN (Yang et al., 2020) and CP-VTON (Wang et al., 2018a). Additionally, we conduct an ablation study to assess the effectiveness of the inpainting module.[38] The model's performance is showcased in both paired and unpaired settings. We provide qualitative results for the various models, and we showcase some of the resulting images.

As highlighted in Section 2.1.1, evaluating generated images, indeed images in general, poses a non-trivial challenge. Nonetheless, several metrics have been developed to assess different facets of generated images. The metrics we will employ are as follows:

1. **Root Mean Squared Error (RMSE)**: the square root of Mean Squared Error (MSE), provides a scale-sensitive representation of the average pixel-wise error between two images, making it more responsive to larger discrepancies.
2. **Peak Signal-to-Noise Ratio (PSNR)**: measures image quality by comparing the maximum pixel value with the MSE, reflecting the relationship between the signal (original image) and noise (difference from the generated image). A higher PSNR indicates a closer resemblance to the original. The formula is:

$$\text{PSNR} = 10 \times \log_{10} \left(\frac{\text{MAX}_I^2}{MSE} \right),$$

   where MAX_I is 255 for an 8-bit image, representing the maximum pixel value, and MSE is the squared differences' average. The multiplication by 10 ensures representation in decibels, which is a logarithmic unit suitable for capturing the ratio of signal to noise.[39]
3. **Learned Perceptual Image Patch Similarity (LPIPS)**: uses deep networks to measure perceptual differences between images.
4. **Structural Similarity (SSIM)**: quantifies image quality by comparing local patterns of luminance, contrast, and structure between two sets of images.
5. **Inception Score (IS)**: evaluates the quality and diversity of generated images using a pre-trained inception model.
6. **Fréchet Inception Distance (FID)**: assesses the similarity between the distribution of generated images and real images using features extracted from an Inception network.

[38] An ablation study is essentially testing the model with certain components or functionality removed to highlight their effectiveness or lack thereof.

[39] It is also a metric that is aligned with human perception.

Table 5.1: Quantitative performance metrics comparison between HR-VITON, our full model, ACGPN, and CP-VTON in a paired setting. ↑ and ↓ represent the desired result for each metric, with ↑ meaning higher results are better, while ↓ means lower results are better.

Model	RMSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑	IS ↑	FID ↓	KID ↓
HR-VITON	0.096	68.49	0.066	0.883	3.515	9.698	0.0030
Ours	**0.117**	**66.77**	**0.094**	**0.875**	**3.256**	**11.97**	**0.0042**
ACGPN[40]	-	-	0.112	0.850	-	-	-
CP-VTON[40]	-	-	0.158	0.786	-	-	-

7. **Kernel Inception Distance (KID)**: estimates the similarity between the distributions of real and generated images without the need for multiple evaluations or bias correction.

It is important to note that only the IS, FID and the KID metrics can be used in both a paired, and unpaired setting, while the others can only be used in a paired setting. For a more comprehensive explanation of the different metrics, see Section 2.1.1. We add additional metrics, the RMSE and the PSNR, not included by either Lee et al. (2022) or Choi et al. (2021), as a measure of similarity between paired images. We start by looking at the results in a paired setting, which follows in the next section.

5.1 Paired setting results

Using the metrics discussed above to evaluate model performance, we look at the virtual try-on images generated in a paired setting — meaning the original image is recreated by the network. This process is shown in Figure 4.16 in Section 4.4. The advantage of using a paired setting is having a ground truth image, from which we can make direct comparisons between the real and the generated images. These comparisons can be across the image in general — LPIPS and SSIM are used for that — or at a pixel level — RMSE and PSNR are used here.

There are four models of interest: the first is HR-VITON, which is the benchmark we are using, secondly, the results of our full model, as described in Chapter 4, are shown. We also include the results of CP-VTON and ACGPN, whose results are taken from Lee et al. (2022); this is done to show our model's improved performance over other virtual try-on networks. These models have previously been discussed in Section 2.2 in the literature review. The results for all of the metrics can be seen in Table 5.1, which shows the different metrics across each of the models. Also shown are the desired results for each metric, with ↑ meaning higher values are more desirable, while ↓ means lower values are more desirable.

From this table, we can see that HR-VITON performs the best across each of the metrics, followed by our model. Both ACGPN and CP-VTON perform worse than our model, however, only LPIPS and SSIM are measured in the paired setting for these models, so a complete comparison cannot be made to them. It is important to note the distinction between our model and the other models; our model first extracts the garment from another image, while HR-VITON, ACGPN, and CP-VTON all use a standalone garment image. To see the results of a paired setting, and a comparison between our model and HR-VITON, consider Figure 5.1, which shows the resulting virtual try-on image derived by the two models, the original image, and the garment used for the try-on. More images can be seen in Appendix C.

[40]These results are taken from the work of Lee et al. (2022). Only LPIPS, SSIM, FID, and KID metrics are given in their paper as they did not measure RMSE, PSNR, or IS. These metrics are taken 'as-is', we did not derive them ourselves. FID and KID are not measured in a paired setting for these models and are thus excluded from the metrics.

(a) HR-VITON's try-on result.

(b) Our model's try-on result.

Figure 5.1: Comparison between HR-VITON and our model in a paired setting. Shown are the original image (top left), the garment used (bottom left), which is the inpainted garment extracted from the original image in our model's case, and the standalone garment image in HR-VITON's case. The resulting virtual try-on is the larger image on the right in each figure.

The results shown in Figure 5.1 and in Table 5.1 show that our model performs well in synthesising high-resolution virtual try-on images. However, the results are not as good as those of HR-VITON, but this is somewhat expected given we are first extracting and then inpainting the garment, while HR-VITON uses a high-quality standalone garment image. To understand the impact of this, consider Figure 5.2. In this figure, three garments are shown: the first is the standalone garment image used by HR-VITON and the like. Secondly, the garment used by our full model, which has been extracted from another image and inpainted to supplement lost information. Lastly, the garment extracted from another image, but without inpainting applied to it is shown, which is discussed further in Section 5.3.

From these images, it is clear why HR-VITON performs better than our model because of the quality and detail of the standalone garment image. There are three main reasons for the standalone garment having better quality and detail than our extracted and inpainted garments: the first is the size of the standalone garment in the image; it is larger and thus has more pixels to represent the details, whereas the extracted garment is kept at the same size as it is in the original image from where it is extracted. Although the extracted and standalone images are the same size, 768×1024, the size of the garment in each image is different. The standalone garment takes up almost all of the image, whereas the extracted garment is much smaller. To highlight the size difference, consider the garment images in Figure 5.1, which are shown in the bottom left of each image. The size difference can also be seen in Figure 5.2. Upsampling could potentially be used to increase the size of the extracted garment, however, it does not add any new or genuine detail to the image. Secondly, any inaccuracies or oversights stemming from the garment extraction and inpainting modules are not only propagated but further magnified within the subsequent virtual try-on model. Thirdly, the garment is already 'warped' when it is extracted, meaning it is extracted according to the body shape of the person in the target garment image — this presents a unique problem as the garment is not 'un-warped' before being used in the virtual try-on model, and again, it is carried over and exacerbated in the try-on image. These effects can be seen in Figure 5.2, particularly between Figure 5.2a and Figure 5.2b.

Notwithstanding this, our model still performs relatively well, with the added practicality of not needing a standalone image. Our model outperforms both ACGPN and CP-VTON in a paired setting, as evident in Table 5.1. Because HR-VITON is the benchmark model, and the best-performing model from the comparison, we use that as our baseline. Our model has an RMSE

(a) Standalone garment. (b) Inpainted garment. (c) Non-inpainted garment.

Figure 5.2: Comparison between the garments used by the different models. Shown, from left to right, is the standalone garment image used by HR-VITON, the extracted and inpainted garment, used by our full model, and lastly, the extracted garment without inpainting applied to it.

that is 21.88% worse than the benchmark, which represents an increase of 0.021 from 0.096 to 0.117. Because PSNR is related to MSE, and by extension RMSE, our model has a slightly lower PSNR value. Our model's PSNR value is 2.511% lower (worse) than the benchmark's, which is a decrease of $1.72dB$ from $68.49dB$ to $66.77dB$. This means that there is more deviance and noise at a pixel level between our model's images and the ground truth when compared to the images produced by HR-VITON. These metrics also show that HR-VITON produces images that better resemble the ground truth when compared to our model.

The LPIPS score for the images produced by our model is 42.42% higher than that of HR-VITON, increasing from 0.066 to 0.094. This implies a larger perceptual difference between the images generated by our model and the ground truth compared to HR-VITON. Conversely, our model produces images that are perceptually closer to the ground truth than both ACGPN and CP-VTON. Their LPIPS scores are 19.15% and 68.09% higher than ours, respectively. For SSIM, which evaluates structural differences between images, our model slightly underperforms compared to HR-VITON. Our model's SSIM stands at 0.875, which is 0.906% lower than HR-VITON's 0.883. Nonetheless, when compared to ACGPN and CP-VTON, our model is superior, with scores 2.857% and 10.29% higher respectively.

IS is a measure of a set of images' diversity and quality, with a higher value meaning images are more diverse and are of higher quality. Our model's IS value is 3.256, which is 7.37% lower than that of HR-VITON, which is 3.515. The fact that these metrics are close to each other, and both are relatively high, indicates that our model and HR-VITON can synthesise high-quality and diverse images. For perspective, the set of real images (i.e. the non-generated images, or the ground truth) has an IS value of 3.725 — meaning our generated images have a quality and diversity that approaches that of real images, as reflected in the IS values.

Both FID and KID are metrics primarily used to evaluate the quality of generated images compared to real images, with lower values indicating better similarity between the distributions of the generated and real datasets. Although these metrics can be applied in both paired and unpaired settings, Lee et al. (2022) did not test ACGPN and CP-VTON in a paired setting, hence we do not report on them. Our model yields an FID of 11.97, which is 23.43% higher than HR-VITON's value of 9.698. Meanwhile, HR-VITON registers a KID score of 0.003, outperforming our model's score of 0.0042 by 40%. These results suggest that the distribution

of images generated by our model is less similar to real images compared to those produced by HR-VITON.

The results indicate that while our model may not surpass HR-VITON in performance, it still outperforms other models in the comparison. It is important to note that the metrics discussed are somewhat subjective. Instead of aiming for specific target values, we often look for higher or lower scores, introducing a degree of subjectivity to the evaluations. With this being said, our model produces relatively good results in a paired setting.

Our model consistently demonstrates strong performance characteristics through thorough testing and evaluations. While our model may slightly trail HR-VITON in most aspects, it significantly surpasses many other models within the domain. The consistently good scores across diverse metrics underscore its robustness and the fidelity of the images it generates. Furthermore, the perceptual closeness of the generated images to the ground truth, as indicated by the LPIPS and SSIM scores, confirms the model's capability to produce perceptually and structurally convincing results. Although the evaluation metrics inherently bear a level of subjectivity, the data presents a compelling case for the efficacy and reliability of our model in image generation tasks.

In a paired setting, we have the distinct advantage of a ground truth image, allowing us to make direct, pixel-wise comparisons between images. However, this does not accurately represent reality or the model's intended use. We now turn our attention from the paired setting to the unpaired one, which more closely mirrors reality. This is discussed in the next section.

5.2 Unpaired setting

In our evaluations, our model demonstrated promising results in a paired setting. Yet, for real-world applicability, this performance must carry over to an unpaired setting. This unpaired context mirrors the model's true intended application, where users would wish to virtually try on garments previously worn by someone else. Quantifying success in this scenario is more challenging compared to the paired setting, as pixel-wise comparisons with a ground truth image are not feasible.[41] Assessing models in an unpaired setting parallels the challenges of evaluating unsupervised learning problems: there is no definitive right answer and evaluations can be somewhat subjective. However, relative comparisons among various models can still provide valuable insights into our model's performance.

As mentioned in the previous section, we utilise three primary quantitative metrics to assess performance: FID, KID, and IS. The IS is unique in that it can be derived solely from a set of generated images, serving to evaluate both their quality and diversity without the need for reference to a real-image distribution. Conversely, FID and KID necessitate a reference set of real (non-generated) images. They function by measuring the similarity between two distributions; specifically the distribution of generated images versus that of real images. In essence, these metrics, FID and KID, provide insights into the realism of the generated images by gauging the closeness of their distribution to authentic images. The results of these metrics in an unpaired setting can be seen in Table 5.2, which again shows the same four models, but this time only the three metrics discussed here are measured. It is important to note that IS is not measured for ACGPN and CP-VTON by Lee et al. (2022).

From this table, we can see that once again HR-VITON is the top-performing model with the best scores across each of the three metrics. All the models perform slightly worse in the

[41] Ideally, a dataset with multiple individuals wearing the same garment would be available for model training, but such a collection is both impractical and expensive.

Table 5.2: Quantitative performance metrics comparison between HR-VITON, our full model, ACGPN, and CP-VTON in an unpaired setting. The arrow notation is as before.

Model	IS ↑	FID ↓	KID ↓
HR-VITON	3.398	11.93	0.0034
Ours	**3.152**	**15.30**	**0.0063**
ACGPN	-	43.29	0.0373
CP-VTON	-	43.28	0.0376

unpaired setting across all the metrics, however, this is to be expected as the shapes and styles of garments transferred from one image to another are now different.[42]

Our model has a smaller IS value in the unpaired setting when compared to the paired one, and it is 7.24% lower than that of HR-VITON. Perhaps the most notable decrease in performance lies in the FID and KID metrics. Our model has an larger FID score of 15.3, which is up 27.82% from the paired setting, and it is 28.25% higher than HR-VITON's score in the same setting. However, our model far outperforms both ACGPN and CP-VTON in terms of FID. Our model has an FID score that is 182.94% better than ACGPN, and 182.88% better than CP-VTON.

Much like the FID value, our KID value has worsened in the unpaired setting compared to the paired setting, up 50% from 0.0042 to 0.0063. In relative terms, our model has also performed worse than HR-VITON with an 85.29% increase in the KID score; this is represented by an increase from 0.0034 to 0.0063. Nonetheless, this is still significantly better than both ACGPN and CP-VTON — ACGPN has a KID score of 0.0373, which is 492.06% worse than ours, with CP-VTON's KID metric being 496.83% worse than ours with a score of 0.0376.

Overall, HR-VITON produces the most perceptually convincing images out of the models compared, but again, it is important to note that it, and ACGPN and CP-VTON use standalone garment images, which have the advantages as discussed in the previous section. Despite this, our model still performs relatively well, especially when compared to ACGPN and CP-VTON (both considered state-of-the-art). The subjective nature of the metrics used makes it difficult to draw absolute conclusions about the model performance, however, they do give us a good guide. Based on the metrics, our model can synthesise perceptually convincing, high-resolution images by first extracting and then inpainting the garment before trying it on the target person. The images generated by our model are diverse and of high quality, as shown by the IS metric. It also generates images that have a similar distribution to that of real images, meaning the model produces very plausible results.

As an example of the results generated by both our model and that of HR-VITON, consider Figure 5.3. We can see that the results produced by both models are similar, and are both perceptually convincing. Our model first extracts the garment from the image in the bottom left of Figure 5.3b and then adds the previously occluded detail back onto the garment. Although not perfect, the detail added resembles the original garment, and the inpainted part can be seen around the woman's right shoulder and neckline area. Both models produce images at high resolution, and the details of the woman in the image are preserved.

Based on these results, we can conclude that our model not only meets but also surpasses the requirements for a virtual try-on system as outlined by Han et al. (2017) and discussed in Section 2.2. In addition to these requirements, our model has the added functionality and practicality of garment extraction, which means we do not require a standalone garment image to be used; this is an aspect where our model surpasses the likes of HR-VITON and VITON-HD. The

[42]This means previously occluded body parts must be synthesised, which adds to the complexity of the model.

(a) HR-VITON's try-on result.

(b) Our model's try-on result.

Figure 5.3: Comparison between HR-VITON and our model in an unpaired setting. Shown are the target person image (top left), the target garment image (bottom left), which is an image of a person wearing the garment in our case, and the standalone garment image in HR-VITON's case. The resulting virtual try-on is the larger image on the right in each figure.

Table 5.3: Ablation study results comparing our model both with and without the inpainting module in a paired setting. The arrow notation is as before.

Model	RMSE ↓	PSNR ↑	LPIPS ↓	SSIM ↑	IS ↑	FID ↓	KID ↓
Full model	0.117	66.77	0.094	0.875	3.256	11.97	0.0042
Without inpainting	0.120	66.55	0.098	0.873	3.116	12.25	0.0049

inpainting module in our garment extraction network allows us to supplement lost information, a shortfall of previous methods outlined by Ishikawa and Ikenaga (2022). The efficacy of this inpainting module is discussed as an ablation study in the section that follows.

5.3 Ablation study

To the best of the author's knowledge, incorporating an inpainting module into the garment extraction network represents a novel approach to virtual try-on systems that also include garment extraction. Ishikawa and Ikenaga (2022) acknowledge in their paper that information loss due to occlusions is a limiting factor in their methodology, which resembles ours by initially extracting a garment from an image for use in the virtual try-on process. In contrast, context-aware image inpainting reconstructs missing or previously occluded pixels by leveraging the information from the surrounding pixel context. The effectiveness of this technique in our virtual try-on task is examined in this section.

To evaluate the performance of our model without inpainting, we simply bypass the inpainting layer, using the extracted garment directly from the target garment image. This approach, without the inpainting, mirrors how HR-VITON would function if provided with a target garment image. In such a scenario, the garment would first need extraction rather than being given a standalone garment. We perform the ablation study in both paired and unpaired settings, with the same metrics as before. The results of the paired setting can be seen in Table 5.3, while the unpaired setting results can be seen in Table 5.4.

From these results, we can see that the inpainting module has a clear, positive effect on the overall virtual try-on network. In the paired setting, our full model outperforms the model without inpainting in each of the seven metrics. The full model's reduced RMSE illustrates its superior

Table 5.4: Ablation study results comparing our model both with and without the inpainting module in an unpaired setting. The arrow notation is as before.

Model	IS ↑	FID ↓	KID ↓
Full model	3.152	15.30	0.0063
Without inpainting	3.128	15.34	0.0064

accuracy with fewer pixel-level discrepancies, while its elevated PSNR value signals a noticeable improvement in image quality. This superiority is further reinforced by a diminished LPIPS score, suggesting that the full model's images bear a closer perceptual likeness to the ground truth. Though both models are almost on par in terms of structural similarity, as indicated by their SSIM scores, the full model marginally outperforms its counterpart. Additionally, its superior IS denotes a commendable blend of quality and diversity in its generated images. Finally, the full model's lower FID and KID scores drive home the conclusion that it crafts images that more closely mimic the authenticity of real images when juxtaposed with the model that operates without inpainting.

The results in the unpaired setting further affirm the advantage the inpainting module brings the virtual try-on network. The full model, which incorporates inpainting, exhibits a slightly superior IS score of 3.152 in comparison to the 3.128 of the model without inpainting, indicating that the full model produces images with slightly better quality and diversity. Even though the FID scores of both models are closely aligned, the full model has a marginally lower value of 15.30 compared to 15.34. This suggests that images generated by the full model have a distribution marginally more resemblant to real images. This is further reinforced by the KID score, with the full model registering an improved result of 0.0063 against the 0.0064 of the model without inpainting, underscoring that the full model's image distribution more closely mirrors that of genuine images. While these differences are slight, they collectively emphasise the positive influence of the inpainting module.

Although the results affirm the inpainting module's positive impact on the virtual try-on network and its role in aligning our generated images more closely with those of HR-VITON compared to other methods, the inpainting effect is most effectively assessed through visual analysis. Consider Figure 5.4, which shows the results generated by the model with inpainting, and without. The target garment image is the same for both of these images, and it can be seen in the bottom left of Figure 5.3b.

The model that uses inpainting first extracts the garment, and then inpaints, or predicts, previously occluded pixels, such as those that are behind hair or an arm, for example. This garment can be seen in Figure 5.2b, while the non-inpainted garment can be seen in Figure 5.2c. The differences between the two garments are evident and seem to be magnified in the try-on image, which diminishes the plausibility and perceptual credibility of the generated image that does not use inpainting.

From this, it is clear to see the added benefit the inpainting layer brings to the virtual try-on with garment extraction network. Although, the quantitative results may not be overly convincing, this is attributed to the fact that not every image needs inpainting, and when inpainting is applied, it is only applied for a small portion of the image. To put this into perspective, on average, 3 386 pixels need to be inpainted per image, however, in a 768×1024 image, which compromises of 786 432 pixels, this only accounts for 0.43% of pixels needing inpainting. This may skew the results slightly in favour of the model without inpainting. To test the efficacy of the inpainting module completely, only images with a certain minimum number of pixels to be

(a) With garment inpainting. (b) Without garment inpainting.

Figure 5.4: Comparison between an image generated by our full model which contains the inpainting layer (a) and one generated by the model without inpainting (b). The difference is evident around the neckline in the second image, with the previously occluded part of the garment being inpainted in the first image, and left as is in the second.

inpainted (occluded pixels) could be tested.[43] However, from the results presented here, and with the aid of the images, we can conclude that the inpainting module is highly effective in improving the performance of the virtual try-on with garment extraction network.

While we have discussed the advantages of using inpainting for enhancing the garment image in a virtual try-on setting, it is essential to recognise the broader horizons of inpainting. Beyond garment enhancement, inpainting techniques offer significant value in the realm of image restoration and enhancement, especially when adapting to the surrounding context of the image is paramount. This context-aware approach not only ensures image integrity but also delivers a more seamless and natural restoration. In the section that follows, we will dive deeper into context-aware image inpainting for image enhancement.

5.4 Inpainting for image enhancement

Context-aware inpainting has long had success in object removal, regional editing, and image restoration and enhancement Kınlı et al. (2020). And, following the work of Kubo et al. (2019), we can apply inpainting to correct any corrupted pixels generated by our virtual try-on network after the fact. To understand how this works, consider Figure 5.5, which shows the output from our model in an unpaired setting (shown in Figure 5.3b) and the resulting image after inpainting has been applied to correct the corrupted pixels. In our original output image, there are pixels near the woman's right ear that are corrupted via our network,[44] circled in red in Figure 5.5a. These are the pixels we are looking to inpaint based on their surrounding context.

We can see the result after inpainting is applied to those pixels in Figure 5.5b. This image, compared to the original one, is more perceptually convincing given that there are now fewer corrupted pixels. This highlights the efficacy of applying context-aware inpainting as a post-processing step to our generated images. More examples showing this benefit can be seen in Figure C.3.

[43] This is not tested as part of this research, but it should be an interesting investigation to carry out.

[44] Pixels in the hair region are often corrupted given that hair detail is fine and very difficult to replicate and restore.

(a) Image needing enhancement.

(b) Enhanced result after inpainting.

Figure 5.5: An example demonstrating how context-aware image inpainting can be used for image enhancement. Shown is the result from our virtual try-on model, highlighting an area where there are corrupted pixels (Figure 5.5a), along with the result after inpainting has been applied (Figure 5.5b). The corrupted pixels have been inpainted to match their surrounding context.

Although this method shows promise in improving our generated images even further, it has the shortfall of not being applied programmatically at this stage — a user will have to manually mask the pixels they wish to inpaint. It is for this reason that we do not include this benefit in measuring our model's performance, we merely provide visual evidence of its efficacy.[45] Furthering this capability will be an interesting study to complete, and it is noted as a recommendation for any future work involving this research.

While our virtual try-on with garment extraction model has showcased promising results in synthesising images, it is imperative to adopt a holistic view of its performance. No model is infallible, and ours is no exception. In the interest of transparency and further refinement, the next section will delve into some notable failure cases of our model. We will explore the reasons behind these shortcomings, shedding light on the inherent limitations and potential avenues for improvement.

5.5 Limitations and failure cases

Although our method has been proven effective, it is equally important to discuss the limitations of our model. Analysing failure cases is not solely an act of transparency but serves several critical purposes. Initially, it provides a comprehensive perspective, ensuring that the research is considered in its totality, rather than solely by its successes. Moreover, by examining the scenarios where our model underperforms, we can identify precise areas of weakness and potential enhancement. This analysis is instrumental in informing future developments and research trajectories, ultimately improving the model's robustness and utility. Furthermore, detailing failure cases establishes realistic expectations and defines the model's constraints for stakeholders and the research community. With this approach of thorough examination and dedication to

[45] We are not able to apply this 'after the fact' inpainting to all the images as it requires the author manually drawing the masks for over 2 000 images, and as such, we cannot test them against the others.

ongoing refinement, we will now explore the specific failure cases of our model.

One limitation, as discussed in Section 4.2.2, is the generation of the inpainting mask. When doing that, we want to only mask the occluded pixels of an image, however, our current, and somewhat rudimentary method of generating a mask oftentimes masks pixels that are not necessarily occluded, or vice versa. This leads to misguided masks which may impact the final virtual try-on image. However, because the number of pixels needing to be inpainted is generally very low, this is not seen as a priority to overcome.

The second, and possibly the largest limiting factor in the model is the prediction of the segmentation map. Because this is arguably the most important factor in getting the virtual try-on image correct, any error in the prediction is exacerbated even further in the try-on image. As an example of the effect incorrect segmentation map prediction can have on the virtual try-on, consider Figure 5.6, which shows both the target person and target garment images, the ground truth image, the extracted target garment image, the predicted segmentation map, and the result.

Figure 5.6: An example highlighting incorrect segmentation map prediction. Shown at the top left is the target person, the target garment image at the top middle, with the ground truth garment at the top right. The extracted and inpainted garment is shown at the bottom left. The predicted segmentation map is shown at the bottom middle, with the resulting virtual try-on shown at the bottom right.

From this image, it is evident that incorrectly predicting the segmentation map hampers the virtual try-on image. Numerous factors affect the segmentation map prediction, including the extracted garment and the pose of the person in both the target garment image and the target person image. Here, in this example, the extracted garment, based on the pose of the target person, is not representative of the full garment — we can see that the extracted garment seemingly has one sleeve missing, which is not true according to the ground truth garment image. This error is introduced based on the fact that we cannot 'unwarp' the garment, and the pose of the person in the target garment image greatly affects that. This error is then carried over into the predicted segmentation map, and subsequently the result, with only one sleeve, albeit partially, being predicted.

Other limitations, similar to the segmentation map prediction, lie in the ability of the 'off-the-

shelf' models to correctly predict the initial segmentation map and the dense correspondence. PGN (Gong et al., 2018) and Densepose (Güler et al., 2018) are responsible for these, respectively. There are inherently different factors that impact how well these models work; for example, if the person's arms are crossed over their torso, the models fail to accurately predict what the segmentation map and dense correspondence should be. As an example of this, consider Figure 5.7, which shows the target person image and its resulting segmentation map and dense correspondence, along with the final try-on result.

Figure 5.7: An example highlighting the limitations experienced by the off-the-shelf models when dealing with complicated poses. Shown at the top left is the target person image, with the resulting segmentation map found by PGN shown at the top right. The dense correspondence, found with Densepose, can be seen at the bottom left, while the resulting virtual try-on can be seen at the bottom right.

From this example, one can see that the complicated pose of the woman impacts the final try-on result. This is underpinned by the segmentation map and the dense correspondence representation, which both fail to accurately capture the position and shape of the woman's arms. These limitations of the models lead to the suboptimal result, which shows the woman with incomplete and distorted arms, and with an overall mismatch of the garment.

Having candidly explored the failures and limitations of our model, we have set a foundation for an informed understanding of its current capabilities and areas of needed improvement. This transparent analysis not only underscores the challenges faced but also maps out potential pathways for future enhancements. While the shortcomings highlighted may pose constraints in certain scenarios, they also drive the quest for innovation and refinement in subsequent iterations. As we transition to the concluding remarks of this chapter, it is crucial to contextualise these limitations within the broader achievements and potential of our work.

5.6 Conclusion

The findings presented in this chapter indicate that the model contributes positively to the development of virtual try-on technology. It has performed well against established benchmarks in both paired and unpaired settings and has shown notable results when compared with models

such as ACGPN and CP-VTON. While the model is competitive with the state-of-the-art HR-VITON and VITON-HD, it does not always produce as perceptually convincing images as they do. Nonetheless, our model integrates garment extraction with context-aware image inpainting, a method not seen in HR-VITON and VITON-HD, which may enhance practical application.

Our ablation study demonstrated the importance of the inpainting module, confirming its central contribution to the model's success. In keeping with a comprehensive approach to our research, we also pointed out areas where the model fell short or faced limitations. Recognising these shortcomings not only offers transparency but also points us towards opportunities for further refinement.

The combination of garment extraction with context-aware image inpainting distinguishes our model and suggests potential for practical applications in real-world settings. The outcomes presented in this chapter demonstrate our model's capacity to meet the research objectives established in Chapter 1. With this being said, we transition into our final chapter, which discusses our research holistically and makes final remarks.

6 | Conclusion

As we conclude this research, we restate the objectives and distinct aspects of our virtual try-on model that uses garment extraction. We clearly present the key findings, highlighting our model's performance compared to other leading models in the field. This comparison highlights our achievements and areas for improvement. We examine the practical implications of our model before revisiting the initial research objectives. We then conclude with a reflective discussion on our research journey and its wider importance.

This study was initiated to develop an image-based virtual try-on network that includes the capability of garment extraction. A pivotal distinction of our model is its ability to extract the target garment directly from an image, eliminating the need for a standalone image — a unique feature in contrast to other state-of-the-art high-resolution models. Our proposed solution meets and often exceeds the exacting requirements for a virtual try-on system, as put forward by Han et al. (2017), which are:

1. The preservation of the target person's body parts and pose, inclusive of hair and accessories.
2. Ensuring that the target garment conforms naturally to the pose and physique of the target individual.
3. Maintaining the integrity of patterns, hues, and nuances of the target garment, encompassing textures, embroideries, and emblems.

Through the application of context-aware image inpainting, our model refines extracted garments to more closely align with ground truth images. This capability is further demonstrated in post-processing, where the inpainting technique enhances the quality of the generated images.

When benchmarked against leading contenders in the field, our model was rigorously validated in both paired and unpaired settings, and evaluated using an array of metrics that are standard for images engendered with GANs. These metrics focus on image quality, structure, diversity, and similarity to real images. In the comparative analysis, the then top-performing model, HR-VITON, marginally outperformed our model in all metrics. However, our model demonstrated notable strengths in various other aspects, highlighting its capability and potential. In the metrics evaluated, our model outperformed both ACGPN and CP-VTON.

To understand the role of the garment inpainting module, an ablation study was carried out. The results of omitting this module underscored its significant value, particularly in unpaired scenarios, and confirmed its effectiveness within our comprehensive model. A visual illustration further demonstrated the ability of inpainting to enhance image generation, corroborating the findings presented by Kubo et al. (2019).

Beyond the outcomes of our model, the study also examined its limitations, identifying areas requiring further development. The model's occasional difficulties with accurately 'unwarping' extracted garments highlighted a need for improved garment fidelity and more accurate seg-

mentation map predictions. Additionally, the occasional inaccuracies in segmentation maps and dense correspondence images generated by off-the-shelf models, particularly in complex poses, were noted as areas needing enhancement. The method employed for inpainting mask generation, although basic, presents opportunities for refinement. Recognising these aspects not only underscores the comprehensiveness of our research but also charts a path for future advancements. Addressing these areas could significantly bolster the model's performance and its practical deployment.

The integration of garment extraction and inpainting modules contributes to the versatility of our model, enhancing its potential for real-world application. The model is designed to be suitable for various deployment contexts, accommodating both standalone garment images and those requiring extraction from composite images, or both. As a result, it is capable of integration into a diverse array of online retail environments, including those lacking a consistent collection of standalone garment images.[46] The functionality and performance of our model make it a candidate for application within the online retail sector, which is valued at $6.31 trillion.

Our research addresses identified gaps in the field and establishes a basis for future studies. The model introduces capabilities with the potential to support advancements in the industry and contribute to the evolution of virtual try-on systems. With this, we showcased how our work answered the set of research objectives as set out in Chapter 1. Throughout our research, we sought to:

1. Design and implement an efficient garment extraction algorithm capable of accurately identifying and extracting clothing items from a variety of images, including those with complex patterns, textures, and colours, and that handles occlusions.
2. Investigate and incorporate pose estimation and body shape modelling techniques for better alignment and adaptation of extracted garments to the target person's body in the virtual try-on process.
3. Develop a robust image-based high-resolution virtual try-on system that accurately and realistically represents garments on target individuals, taking into account various body shapes, sizes, and poses, and that handles occlusions.
4. Evaluate the performance of the proposed high-resolution virtual try-on system through quantitative comparisons with existing state-of-the-art methods.
5. Explore potential applications and extensions of the high-resolution virtual try-on system.

The methodology we employed, combined with the practical outcomes and results demonstrated by our model, effectively addresses our research objectives. It aligns with, and in certain areas surpasses, the initial goals and requirements set for this study. As we conclude this research, we do so with a profound sense of achievement and anticipation. We look forward with enthusiasm to future developments in the field of virtual try-ons that may be inspired by or built upon our work.

[46] For perspective, a notable South African online retailer, Superbalist, displays an inconsistent mix of product imagery: some items are shown as standalone garments, while others are modelled by individuals.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). TensorFlow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

Agustsson, E. and Timofte, R. (2017). NTIRE 2017 challenge on single image super-resolution: Dataset and study. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*.

Azadi, S., Olsson, C., Darrell, T., Goodfellow, I., and Odena, A. (2019). Discriminator rejection sampling. arXiv preprint. https://arxiv.org/abs/1810.06758.

Belongie, S., Malik, J., and Puzicha, J. (2002). Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(4):509–522.

Bińkowski, M., Sutherland, D. J., Arbel, M., and Gretton, A. (2021). Demystifying MMD GANs. arXiv preprint. https://arxiv.org/abs/1801.01401.

Cao, Z., Hidalgo, G., Simon, T., Wei, S. E., and Sheikh, Y. (2019). OpenPose: Realtime multi-person 2D pose estimation using part affinity fields. arXiv preprint. https://arxiv.org/abs/1812.08008.

Cao, Z., Simon, T., Wei, S. E., and Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. arXiv preprint. https://arxiv.org/abs/1611.08050.

Chen, L. C., Papandreou, G., Kokkinos, I., Murphy, K., and Yuille, A. L. (2017). DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected CRFs. arXiv preprint. https://arxiv.org/abs/1606.00915.

Choi, S., Park, S., Lee, M., and Choo, J. (2021). VITON-HD: High-resolution virtual try-on via misalignment-aware normalization. arXiv preprint. https://arxiv.org/abs/2103.16874.

Clevert, D. A., Unterthiner, T., and Hochreiter, S. (2016). Fast and accurate deep network learning by Exponential Linear Units (ELUs). arXiv preprint. https://arxiv.org/abs/1511.07289.

Collins, E., Bala, R., Price, B., and Süsstrunk, S. (2020). Editing in style: Uncovering the local semantics of GANs. arXiv preprint. https://arxiv.org/abs/2004.14367.

Cramer-Flood, E. (2022). Worldwide ecommerce forecast update 2022. https://www.insiderintelligence.com/content/worldwide-ecommerce-forecast-update-2022. Accessed 14 March 2023.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*, 29(6):141–142.

Dong, H., Liang, X., Wang, B., Lai, H., Zhu, J., and Yin, J. (2019). Towards multi-pose guided virtual try-on network. arXiv preprint. https://arxiv.org/abs/1902.11026.

Dosovitskiy, A. and Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. arXiv preprint. https://arxiv.org/abs/1602.02644.

Elgaard, H. (2023). Return on ad spend (ROAS): The first (but not only) step in analyzing e-commerce results. https://www.tangiblee.com/blog/return-on-ad-spend-roas-the-first-but-not-only-step-in-analyzing-e-commerce-results. Accessed 14 March 2023.

Gong, K., Liang, X., Li, Y., Chen, Y., Yang, M., and Lin, L. (2018). Instance-level human parsing via part grouping network. arXiv preprint. https://arxiv.org/abs/1808.00157.

Gong, K., Liang, X., Zhang, D., Shen, X., and Lin, L. (2017). Look into Person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. arXiv preprint. https://arxiv.org/abs/1703.05446.

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014). Generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1406.2661.

Güler, R. A., Neverova, N., and Kokkinos, I. (2018). DensePose: Dense human pose estimation in the wild. arXiv preprint. https://arxiv.org/abs/1802.00434.

Gulrajani, I., Ahmed, F., Arjovsky, M., Dumoulin, V., and Courville, A. (2017). Improved training of wasserstein gans. arXiv preprint. https://arxiv.org/abs/1704.00028.

Han, X., Wu, Z., Wu, Z., Yu, R., and Davis, L. (2017). VITON: An image-based virtual try-on network. arXiv preprint. https://arxiv.org/abs/1711.08447.

He, K., Zhang, X., Ren, S., and Sun, J. (2015). Deep residual learning for image recognition. arXiv preprint. https://arxiv.org/abs/1512.03385.

Heusel, M., Ramsauer, H., Unterthiner, T., Nessler, B., and Hochreiter, S. (2018). GANs trained by a two time-scale update rule converge to a local nash equilibrium. arXiv preprint. https://arxiv.org/abs/1706.08500.

Huang, K., Hong, W., and Ma, Y. (2005). Symmetry-based photo editing. *Pattern Recognition*, 38(6):825–834.

Iizuka, S., Simo-Serra, E., and Ishikawa, H. (2017). Globally and locally consistent image completion. *ACM Transactions on Graphics*, 36:1–14.

Ishikawa, S. and Ikenaga, T. (2022). Image-based virtual try-on system with clothing extraction module that adapts to any posture. *Computers & Graphics*, 106(C):161–173.

Isola, P., Zhu, J. Y., Zhou, T., and Efros, A. A. (2018). Image-to-image translation with conditional adversarial networks. arXiv preprint. https://arxiv.org/abs/1611.07004.

Jablonski, N. G. and Chaplin, G. (2000). The evolution of human skin coloration. *Journal of Human Evolution*, 39(1):57–106.

Jetchev, N. and Bergmann, U. (2017). The conditional analogy GAN: Swapping fashion articles on people images. arXiv preprint. https://arxiv.org/abs/1709.04695.

Johnson, J., Alahi, A., and Fei-Fei, L. (2016). Perceptual losses for real-time style transfer and super-resolution. arXiv preprint. https://arxiv.org/abs/1603.08155.

Karras, T., Aila, T., Laine, S., and Lehtinen, J. (2018). Progressive growing of gans for improved quality, stability, and variation. arXiv preprint. https://arxiv.org/abs/1710.10196.

Karras, T., Laine, S., Aittala, M., Hellsten, J., Lehtinen, J., and Aila, T. (2020). Analyzing and improving the image quality of StyleGAN. arXiv preprint. https://arxiv.org/abs/1912.04958.

Kendall, A., Grimes, M., and Cipolla, R. (2016). PoseNet: A convolutional network for real-time 6-dof camera relocalization. arXiv preprint. https://arxiv.org/abs/1505.07427.

Kingma, D. P. and Ba, J. (2014). Adam: A method for stochastic optimization. arXiv preprint. https://arxiv.org/abs/1412.6980.

Kubo, S., Iwasawa, Y., Suzuki, M., and Matsuo, Y. (2019). UVTON: UV mapping to consider the 3D structure of a human in image-based virtual try-on network. *2019 IEEE/CVF International Conference on Computer Vision Workshop (ICCVW)*, 1:3105–3108.

Kumar, A. (2020). Semantic image segmentation using fully convolutional networks. https://towardsdatascience.com/semantic-image-segmentation-using-fully-convolutional-networks-bf0189fa3eb8. Accessed 17 June 2023.

Kınlı, F., Özcan, B., and Kıraç, F. (2020). A benchmark for inpainting of clothing images with irregular holes. arXiv preprint. https://arxiv.org/abs/2007.05080.

Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., and Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. arXiv preprint. https://arxiv.org/abs/1609.04802.

Lee, S., Gu, G., Park, S., Choi, S., and Choo, J. (2022). High-resolution virtual try-on with misalignment and occlusion-handled conditions. arXiv preprint. https://arxiv.org/abs/2206.14180.

Lewis, K. M., Varadharajan, S., and Kemelmacher-Shlizerman, I. (2021). TryOnGAN: Body-aware try-on via layered interpolation. arXiv preprint. https://arxiv.org/abs/2101.02285.

Li, P., Xu, Y., Wei, Y., and Yang, Y. (2022). Self-correction for human parsing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(6):3260–3271.

Lin, T. Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. arXiv preprint. https://arxiv.org/abs/1612.03144.

Liu, G., Reda, F. A., Shih, K. J., Wang, T.-C., Tao, A., and Catanzaro, B. (2018). Image inpainting for irregular holes using partial convolutions. arXiv preprint. https://arxiv.org/abs/1804.07723.

Ma, L., Jia, X., Sun, Q., Schiele, B., Tuytelaars, T., and Gool, L. V. (2018). Pose guided person image generation. arXiv preprint. https://arxiv.org/abs/1705.09368.

Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., and Smolley, S. P. (2017). Least squares generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1611.04076.

Miyato, T., Kataoka, T., Koyama, M., and Yoshida, Y. (2018). Spectral normalization for generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1802.05957.

Northcutt, C. G., Jiang, L., and Chuang, I. L. (2021). Confident learning: Estimating uncertainty in dataset labels. *Journal of Artificial Intelligence Research (JAIR)*, 70:1373–1411.

NVIDIA, Vingelmann, P., and Fitzek, F. H. (2020). CUDA, release: 10.2.89. Retrieved from https://developer.nvidia.com/cuda-toolkit.

Odena, A., Olah, C., and Shlens, J. (2017). Conditional image synthesis with auxiliary classifier gans. arXiv preprint. https://arxiv.org/abs/1610.09585.

Park, T., Liu, M. Y., Wang, T. C., and Zhu, J. Y. (2019). Semantic image synthesis with spatially-adaptive normalization. arXiv preprint. https://arxiv.org/abs/1903.07291.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). PyTorch: An imperative style, high-performance deep learning library. arXiv preprint. https://arxiv.org/abs/1912.01703.

Radford, A., Metz, L., and Chintala, S. (2016a). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1511.06434.

Radford, A., Metz, L., and Chintala, S. (2016b). Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1511.06434.

Razavi, A., van den Oord, A., and Vinyals, O. (2019). Generating diverse high-fidelity images with VQ-VAE-2. arXiv preprint. https://arxiv.org/abs/1906.00446.

Ronneberger, O., Fischer, P., and Brox, T. (2015). U-Net: Convolutional networks for biomedical image segmentation. arXiv preprint. https://arxiv.org/abs/1505.04597.

Salimans, T., Goodfellow, I., Zaremba, W., Cheung, V., Radford, A., and Chen, X. (2016). Improved techniques for training GANs. arXiv preprint. https://arxiv.org/abs/1606.03498.

Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. arXiv preprint. https://arxiv.org/abs/1409.1556.

Song, Y., Li, Y., Wu, B., Chen, C. Y., Zhang, X., and Adam, H. (2017). Learning unified embedding for apparel recognition. arXiv preprint. https://arxiv.org/abs/1707.05929.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. arXiv preprint. https://arxiv.org/abs/1409.3215.

von Neumann, J. and Morgenstern, O. (1944). *Theory of Games and Economic Behavior*. Princeton University Press.

Wang, B., Zheng, H., Liang, X., Chen, Y., Lin, L., and Yang, M. (2018a). Toward characteristic-preserving image-based virtual try-on network. arXiv preprint. https://arxiv.org/abs/1807.07688.

Wang, T. C., Liu, M. Y., Zhu, J. Y., Tao, A., Kautz, J., and Catanzaro, B. (2018b). High-resolution image synthesis and semantic manipulation with conditional gans. arXiv preprint. https://arxiv.org/abs/1711.11585.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Weatherall, I. L. and Coombs, B. D. (1992). Skin color measurements in terms of cielab color space values. *The Journal of Investigative Dermatology*, 99(4):468–473.

Yang, H., Zhang, R., Guo, X., Liu, W., Zuo, W., and Luo, P. (2020). Towards photo-realistic virtual try-on by adaptively generating↔preserving image content. arXiv preprint. https://arxiv.org/abs/2003.05863.

Yi, Z., Tang, Q., Azizi, S., Jang, D., and Xu, Z. (2020). Contextual residual aggregation for ultra high-resolution image inpainting. arXiv preprint. https://arxiv.org/abs/2005.09704.

Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? arXiv preprint. https://arxiv.org/abs/1411.1792.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. (2019a). Free-form image inpainting with gated convolution. arXiv preprint. https://arxiv.org/abs/1806.03589.

Yu, J., Lin, Z., Yang, J., Shen, X., Lu, X., and Huang, T. S. (2018). Generative image inpainting with contextual attention. arXiv preprint. https://arxiv.org/abs/1801.07892.

Yu, R., Wang, X., and Xie, X. (2019b). VTNFP: An image-based virtual try-on network with body and clothing feature preservation. *2019 IEEE/CVF International Conference on Computer Vision (ICCV)*, 1:10510–10519.

Zhang, H., Xu, T., Li, H., Zhang, S., Wang, X., Huang, X., and Metaxas, D. (2017). StackGAN: Text to photo-realistic image synthesis with stacked generative adversarial networks. arXiv preprint. https://arxiv.org/abs/1612.03242.

Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

Zhang, R., Isola, P., Efros, A. A., Shechtman, E., and Wang, O. (2018). The unreasonable effectiveness of deep features as a perceptual metric. arXiv preprint. https://arxiv.org/abs/1801.03924.

Zhou, B., Khosla, A., Lapedriza, A., Torralba, A., and Oliva, A. (2016). Places: An image database for deep scene understanding. arXiv preprint. https://arxiv.org/abs/1610.02055.

A | Additional EDA information

In this Appendix, there is additional information relating to Chapter 3. It contains information that should bring clarity to any misunderstanding. Because colour is used extensively throughout the Exploratory Data Analysis (EDA) section, it would be amiss to not include any tabular information for completeness' sake. Colourblind-friendly colour palettes cannot be used due to the nature of the data; the colours represented in the visualisations are the actual colours represented in the data, thus, this tabular information will cover the shortfall in information.

The data shown in Table A.1 shows the information which was used to build Figure 3.2 in Subsection 3.1.1. Shown in the table is the bar number (from left to right in the plot), with its associated count of garments, a sample of the colour, and the HEX colour code. The bars in Figure 3.2 are arranged by order of decreasing count in the dataset, and so too is the table.

Table A.1: Distribution of average garment colours ordered by count.

Bar No.	Count	Colour sample	HEX colour
1	2359		ECE9E8
2	1720		1A191D
3	1260		0F0F10
4	955		212229
5	576		DDDEDE
6	540		252B38
7	316		EDD6CC
8	286		CFCBC9
9	240		3F3B39
10	226		283A52
11	207		555450
12	170		441C22
13	167		6A1E2B
14	157		6F6B5C
15	152		B3B6B7
16	145		8E2A32
17	144		CB1629
18	141		E8B6AF
19	137		E9BD3A
20	130		E83334
21	127		959B99
22	126		7E8379
23	123		AD4935
24	110		ECD266
25	109		E9554F

Table A.1: Distribution of average garment colours ordered by count.

Bar No.	Count	Colour sample	HEX colour
26	101		CC9C89
27	95		E18F24
28	87		B05C5E
29	82		B0CADF
30	79		274793
31	79		5F728F
32	78		EAE99C
33	76		ED8A72
34	70		C18A55
35	69		1A6968
36	51		7C9DCD
37	49		179E8D
38	40		2E73C8
39	39		ED5E9E
40	29		59C5C2

The data shown in Table A.2 shows the information which was used to build Figure 3.6 in Subsection 3.1.3. The bars in Figure 3.6 are arranged by order of decreasing brightness of the colours, as is the table.

Table A.2: Distribution of average skin colours ordered by brightness.

Bar No.	Count	Colour sample	HEX colour	Brightness
1	224		D9B6A2	187.9
2	642		D0AB96	177.3
3	855		CAA38D	169.7
4	537		C89C83	163.5
5	590		C09B87	161.4
6	897		C0957D	156.4
7	774		B7907B	150.7
8	680		BC8D72	149.0
9	910		B08974	143.7
10	727		B3846A	140.1
11	876		A7806C	134.8
12	635		AB7B60	131.2
13	684		9E7763	125.8
14	552		A17157	121.3
15	464		936D5A	115.7
16	442		976750	111.5
17	533		8B5F4C	102.9
18	349		815644	93.84
19	212		714B3C	81.99
20	49		5A3f34	67.94

The data shown in Table A.3 shows the information which was used to build Figure 3.7 in Subsection 3.1.4. The bars in Figure 3.7 are arranged by order of decreasing brightness of the colours, and so too is the table.

Table A.3: Distribution of average hair colours ordered by brightness.

Bar No.	Count	Colour sample	HEX colour	Brightness
1	111		A69079	147.0
2	216		997F66	130.7
3	353		90745B	120.1
4	324		876B52	111.1
5	337		80624A	102.6
6	380		765A44	94.36
7	390		70523C	86.78
8	422		674A36	78.72
9	530		5E4331	71.44
10	728		553D2E	65.01
11	666		503727	59.15
12	484		47352C	56.17
13	746		462F23	51.02
14	664		3C2E27	48.47
15	673		3C291E	44.24
16	1141		332720	41.04
17	802		2A231E	36.12
18	849		2E2119	35.18
19	1169		241D18	30.12
20	649		1D1814	24.77

B | Implementation details

The details in this appendix relate to Section 4.4, which outlines the training methodology followed in building the models. The details here cover the technical implementation of each of the different networks, and the details of how the model was used at test time. Technical specifications, such as the hardware and programs used, as well as how long the models are trained for, and what hyperparameters are used are covered. Part Grouping Network (PGN) (Gong et al., 2018), Densepose (Güler et al., 2018), and OpenPose (Cao et al., 2019) are utilised as off-the-shelf models, with no specific emphasis placed on their training and hyperparameters. Such an approach is common practice in the field of computer vision. We start by explaining the implementation details of the context-aware image inpainting model.

B.1 Inpainting model

The context-aware inpainting model, as developed by Yi et al. (2020), underwent a meticulous training process specifically tailored for high accuracy and efficiency. It benefited from the computational capabilities of two NVIDIA 1080 Ti Graphics Processing Units (GPUs), devices acclaimed for their proficiency in intricate neural network computations.

Maintaining uniformity in the training dataset was paramount. Thus, an image resolution of 512×512 pixels was chosen, alongside a batch size of 8 and 300 000 iterations. This configuration effectively balanced computational efficiency with gradient accuracy.

To achieve dataset consistency, images from DIV2K (Agustsson and Timofte, 2017) and CelebA-HQ (Karras et al., 2018) collections were systematically down-sampled to 512×512 pixels. For the Places2 dataset (Zhou et al., 2016), random cropping was the chosen method to achieve the same pixel dimensions, ensuring a streamlined training process across diverse datasets.

Following the training phase, the model underwent testing across various image resolutions, ranging from 512 pixels to 8K. This comprehensive testing, executed on a dedicated GPU, aimed to simulate diverse real-world scenarios, validating the model's adaptability and preparedness.

With a total of 2.7 million parameters, the final model version showcases compact efficiency. TensorFlow version 1.13 (Abadi et al., 2015) supported its implementation on the software front. Enhanced by CUDNN version 7.6 and CUDA version 10.0 (NVIDIA et al., 2020), this software framework ensured optimised performance, notably during training and testing. We now explore the implementation specifics of the virtual try-on model.

B.2 Virtual try-on model

During the training phase of the try-on condition generator, predictions were made at a resolution of 256×192. However, in the inference stage, both the segmentation map and the

appearance flow derived from the try-on condition generator underwent upscaling to achieve a resolution of 768×1024.

The discriminator within the try-on condition generator employed a down-sampling technique, with inputs down-sampled by a factor of 2. This approach expanded the receptive field, allowing the discriminator to process a wider range of contextual information from the images.

To enhance the stability of the training process, a dropout technique was introduced to the discriminator, providing a safeguard against potential overfitting. Different modules within the architecture had specific batch sizes: a batch size of 8 for the try-on condition generator and a batch size of 4 for the image generator. Both modules underwent 100 000 iterations of training. The learning rates for both the generator and the discriminator of the try-on condition generator were set at 0.0002.

The training phase spanned approximately four days, facilitated by the use of two RTX 3090 GPUs. This entire process utilised the PyTorch version 1.8.2+cu111 environment (Paszke et al., 2019), ensuring consistent and efficient training outcomes.

During the training process of the discriminator for the try-on image generator, particular attention was given to the learning rates, ensuring stable convergence. Specifically, the learning rate for the generator was set at 0.0001, whereas the discriminator benefitted from a slightly higher rate of 0.0004.

Choosing the right optimisation algorithm can be instrumental to both the model's performance and the speed of its convergence. With this in mind, the acAdam optimiser was employed for both modules, acknowledged for its efficiency and robustness. The hyperparameters β_1 and β_2 play pivotal roles within the The Adaptive Moment Estimation (Adam) optimiser, dictating the moment estimates. For this specific training phase, β_1 was designated a value of 0.5, and β_2 was set at 0.999. These selected values strike a balance between the immediate past gradient and the accumulation of past gradients, leading to a consistent training progression. Finally, the implementation details for the inference phase of research are discussed in the section that follows.

B.3 Inference implementation details

The inference of the developed model was carried out on a 2021 Apple MacBook with the M1 chip, utilising the Central Processing Unit (CPU) for the computation.[47] The choice of this environment provides insights into the model's performance on standard, consumer-grade hardware, rather than specialised computational platforms.

For the software infrastructure, Python 3.10 served as the programming language of choice. This version of Python has seen improvements in both performance and library support over the years, ensuring the seamless execution of high-performance tasks. Moreover, the computational framework supporting this endeavour was PyTorch 2.0. As one of the leading deep learning libraries, PyTorch offers a dynamic computation graph, which provides flexibility during model development and is highly beneficial when executing forward-only computation during inference.

A key metric of interest was the inference time. For the generation of a single image, the model took approximately 1 minute and 45 seconds on the aforementioned setup. This duration can be considered when gauging the model's feasibility for real-time or near-real-time applications.

[47] It was interesting to note that, when both the built-in CPU and GPU were tested, the CPU outperformed the GPU in this particular setup. Nonetheless, this should not be directly compared to the performance of a dedicated NVIDIA GPU, or the like, in similar setups.

It is pertinent to mention that while the Apple M1 chip provides a commendable computing prowess for general tasks, dedicated GPUs are inherently designed to handle parallel computations that deep learning models often require. Thus, in a productionised setting, to achieve optimal speeds, especially for applications necessitating quick turnarounds or handling a large volume of requests, the utilisation of a dedicated GPU would be the preferred approach. The parallel processing capabilities of GPUs make them ideally suited for the matrix and tensor operations which underpin deep learning algorithms. By offloading these computations to a GPU, one can expect a substantial reduction in inference times, leading to a more efficient and responsive system.

C | Additional results

In this appendix, we show additional results for the different models to supplement those shown in Chapter 5. Paired results are shown in Figure C.1, and unpaired results can be seen in Figure C.2. All the images here are generated by our model. In addition to these results, we include images showing the benefit of using inpainting for enhancement; these are shown in Figure C.3.

Figure C.1: Additional virtual try-on results in a paired setting. Shown in each figure are the original image (left), and the generated result (right).

Figure C.2: Additional virtual try-on results in an unpaired setting. Shown in each figure are the target person (top left), the target garment (bottom left), and the result (right).

Figure C.3: Additional images showcasing the application of context-aware image inpainting for enhancing the virtual try-on results. Shown in each figure are the original virtual try-on result (left), and the result after inpainting is applied to remove unwanted artefacts (right). The areas to be inpainted are circled in red.

D | Plagiarism declaration

I, Daniel John Charters, a student at the University of Cape Town in the Department of Statistical Sciences, with student number CHRDAN015, declare that:

1. I know that plagiarism is wrong. Plagiarism is using another's work and pretending that it is one's own.
2. I have used a generally accepted citation and referencing style. Each contribution to, and quotation in, this report from the work(s) of other people has been attributed and has been cited and referenced.
3. This dissertation is my work.
4. I have not allowed, and will not allow, anyone, to copy my work to pass it off as their work.
5. I acknowledge that copying someone else's work, or part of it, is wrong, and declare that this is my work.

Signed on April 30, 2024:

Signed by candidate

Daniel John Charters