UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
HONORS PROGRAM

NGUYỄN NGỌC KHÔI NGUYÊN - PHAN NGUYỄN THANH TÙNG

DEEP FASHION WITH OUTFIT RECOMMENDATION AND TRY-ON

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

HO CHI MINH CITY, 2023

UNIVERSITY OF SCIENCE
FACULTY OF INFORMATION TECHNOLOGY
HONORS PROGRAM

NGUYỄN NGỌC KHÔI NGUYÊN 19120106
PHAN NGUYỄN THANH TÙNG 19120424

DEEP FASHION WITH OUTFIT RECOMMENDATION AND TRY-ON

BACHELOR OF SCIENCE IN COMPUTER SCIENCE

THESIS ADVISOR:
DR. LÊ TRUNG NGHĨA
ASSOC. PROF. TRẦN MINH TRIẾT

HO CHI MINH CITY, 2023

COMMENT OF THESIS'S ADVISOR

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

. . . . . .

ACKNOWLEDGEMENTS

First and foremost, we would like to extend our heartfelt appreciation to our advisors, Dr. Le Trung Nghia and Assoc. Prof. Tran Minh Triet, for their invaluable guidance and unwavering support throughout the completion of this thesis. Their expertise, constant encouragement, constructive feedback, and inspirational motivation have played a pivotal role in molding the progress of this thesis. We are deeply grateful for their helpful advice, which not only helps us in this thesis but also in terms of professional work ethic and attitude.

Secondly, we want to express our gratitude to the Faculty of Information Technology members at the University of Science, VNU-HCM. Their knowledge and guidance have profoundly influenced our academic journey and fostered a strong foundation for us to carry out this research.

We are also indebted to our teachers and friends at SELAB and Honors Program Class 19, Faculty of Information Technology at the University of Science, VNU-HCM, for their support, advice, and assistance throughout our process.

Last but not least, we express our deep gratitude to our family, whose belief in us and persistent presence. Their unconditional love, unwavering encouragement, and immeasurable have been a constant source of motivation to help us in the accomplishment of this thesis and throughout our entire learning journey.

This thesis would not have come to fruition without the collective contributions and support of all those mentioned above, and for that, we are truly grateful.

THESIS PROPOSAL

Thesis title: Deep Fashion with Outfit Recommendation and Try-on
Advisor: Dr. Lê Trung Nghĩa - Assoc. Prof. Trần Minh Triết
Duration: December 2022 to June 2023
Students: Nguyễn Ngọc Khôi Nguyên (19120106) - Phan Nguyễn Thanh Tùng (19120424)
Introduction: The fashion industry is increasingly developing in terms of quantity and quality. Prior to purchasing a garment, customers tend to try them on. However, this fitting process has limitations such as time consumption, the risk of damaging the garments, unavailability of stock, and the need for customers to physically visit the store to try them on. To address this issue, researchers have proposed the "Virtual Try-On" concept to enable customers to see the results of trying on garments without physically going to the store. This idea has opened up various research directions, from 3D technology to augmented reality (AR) and visual feature-based methods. This thesis focuses on the "Visual Feature-based Virtual Try-On" approach. By utilizing deep learning networks, the method takes, as input, an image of a person's pose and combines it with the image of an available garment. The result is an image of the person wearing that particular clothing item. This approach allows customers to easily try on garments anywhere, saving time and enabling them to try out a wider range of clothing options.

Objectives:

The research project of the student group focuses on the usability factor while still achieving high effectiveness. Specifically, the input only requires an image of a person and an image of the desired garment to try on. Most existing methods within this category prioritize generating visually appealing try-on images, even if it requires sacrificing processing time. This demands significant hardware capabilities for real-world implementation, thus making widespread application challenging.

Therefore, this thesis aims to develop solutions that improve computational speed while maintaining good output quality. The objective is to achieve fast processing while ensuring satisfactory output. These results enable individuals to easily try on garments from fashion stores using a mobile phone or computer, regardless of location. Additionally, this method can pave the way for new avenues of research in visual feature-based virtual try-on.

Scope:

The research project focuses on the direction of "Virtual Try-On", with input consisting of an upper-body image of a person and an image of a garment. The output is an image of the person in the input image wearing that particular garment. The objective is to achieve fast processing speed while maintaining comparable try-on results to existing methods.

Method:

Research on visual feature-based virtual try-on has attracted significant attention in recent years due to its wide-ranging applications in the fashion industry. Prominent studies such as VITON [1], CP-VTON [2], ClothFlow [3], ACGPN [4] work by segmenting the human body into different parts (such as head, hair, arms, upper body, lower body) and/or predicting body joint connections (pose estimation) to enable deep learning models to transform the input garments to fit the body shape appropriately. However, if the segmentation or pose estimation process encounters issues, it can lead to inaccurate virtual try-on results. Moreover, body segmentation algorithms are highly complex, resulting in a significant increase in computational speed.

To address these limitations, recent studies such as như WUTON [5], PFAFN [6], Flow-Style-VTON [7] have proposed a second approach: utilizing information solely from the person and the clothing images during the virtual try-on image generation process, without the need for segmentation results.

Due to the complex model structure and processing steps involved in the first approach (requiring preprocessing for segmentation and pose estimation), this dissertation primarily adopts the second approach to construct a simpler virtual try-on system. The objective is to provide a solution that balances aesthetic quality and algorithm processing speed, establishing a foundation for practical applications. The proposed solution is expected to utilize a deep learning model, followed by parameter adjustments to achieve the most balanced results in terms of quality and speed.

Expected Results:

The expected outcomes of this thesis include the following:

- Experimental results on the accuracy, runtime as well as size of the model.
- The prototype that allows users to generate try-on photos from models and outfits.

Research timeline:

- **December 2022 - February 2023**: Conduct research for recent methods of using deep learning networks for virtual matching problems.
- **March - April 2023**: Improved algorithm results on images.
- **March - April 2023**: Improved algorithm processing speed.
- **April - May 2023**: Run experiments and compare with recent methods.
- **June 2023**: Develop an virtual try-on application.

Advisors	**Students**
Dr. Lê Trung Nghĩa	Nguyễn Ngọc Khôi Nguyên
Assoc. Prof. Trần Minh Triết	Phan Nguyễn Thanh Tùng

TABLE OF CONTENTS

CHAPTER 5 – FASHION RECOMMENDATION

CHAPTER 6 – APPLICATIONS

LIST OF TABLES

LIST OF FIGURES

ABSTRACT

The fashion e-commerce industry has become an important part of people's lives worldwide. To achieve this, companies have continuously explored and employed technological advancements to provide their customers with positive and engaging shopping experiences. Within this context, the development of recommendation systems has emerged as a pivotal concern, aiming to enhance customer experiences by offering personalized suggestions aligned with individual preferences and styles. Furthermore, addressing the crucial need of online shoppers, virtual try-on technology enables users to visualise how specific garments would look when appearing on them, thereby elevating the customer experience and mitigating damaged item costs for retailers.

This thesis investigates two major challenges in fashion e-commerce: **fashion recommendation** and **virtual try-on**. Regarding recommendation problem, we explore various approaches for three types of retrieval: similar items retrieval within the same category, complementary items retrieval from different categories, and text feedback-guided item retrieval. Notably, our method for retrieving complementary items, based on the transformer architecture, has demonstrated its effectiveness through experimental evaluation. Furthermore, we enhance the overall performance of the search pipeline by integrating approximate algorithms, thereby optimizing the search process. As for the virtual try-on tasks, to address the speed limitations of previous methods, we propose a virtual try-on framework designed to be faster and more memory-efficient while still maintaining realistic output compared to predecessors. The main contribution is that we utilize the knowledge distillation technique, which uses a strong Teacher model to achieve a lightweight Student network.

To exemplify the potential of this thesis, we develop applications in two scenarios: a web-based Smart Fashion Assistant System for online shopping and an AR application for clothing try-on, namely Magic Mirror.

CHAPTER 1
INTRODUCTION

This chapter overviews e-commerce within fashion retail, focusing on the challenges associated with personalized and immersive experiences. We first discuss two significant online shopping problems, which involve virtual try-on technologies and fashion recommendation systems, and investigate various approaches to tackle those problems. Next, we describe the main contributions of this thesis to bridge the gap between physical and online fashion retail. Finally, the organizational structure of the thesis is presented, offering a concise overview of the subsequent chapters and their contents.

1.1 Overview

Nowadays, the fashion industry has become an important part of people's lives. The development of digital globally and the recent pandemic has led to changes in consumer behaviour: e-commerce is increasingly popular and developed. Traditional retail shop owners also expand their stores on online platforms such as Amazon, Zalando, etc. Because of this, it is hardly surprising that the fashion e-commerce market will continue to flourish and diversify at an extraordinary pace. The global fashion e-commerce market is expected to reach a value of over 820 billion U.S. dollars in 2023 and could be 1.2 trillion U.S. dollars in 2027 [8] (details are shown in Figure 1.1). Nevertheless, the transition from traditional in-store shopping to online shopping presents distinctive challenges, such as the time-consuming process of item selection without some recommendations from the sales staff, limitations in trying on clothes before making a purchase, and risks and costs associated with product transportation.

In response to these challenges, intelligent fashion technologies have emerged as

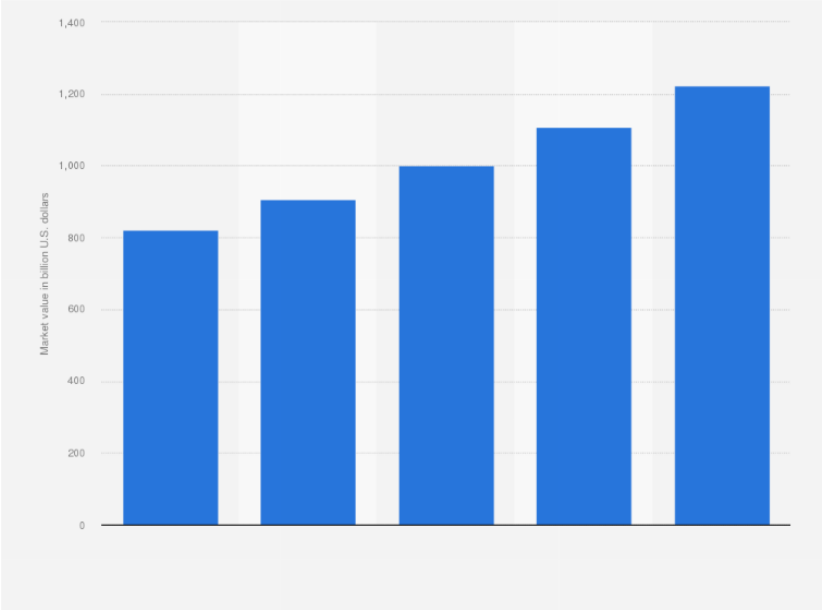


Figure 1.1: Fashion e-commerce market value worldwide forecast 2023-2027 (in billion U.S. dollars) [8].

promising solutions to serve customers with the same support and comfort as in the in-person shopping experience. This thesis focuses on two significant areas of intelligent fashion: fashion recommendation systems and virtual try-on technologies. Both methodologies share a common objective of augmenting the customer experience on online platforms, thereby enabling retailers to bolster profitability. Fashion recommendation systems help users make easier and smarter shopping choices by recommending items that suit their preferences and style. Furthermore, virtual try-on technology has garnered significant attention in recent years by addressing a crucial concern of online shoppers: being concerned about how a particular in-shop garment will look when they wear it and whether it suits them. By allowing users to try on items virtually, this technology enhances the customer experience and saves the cost of damaged items for retailers. These solutions draw a new picture of online shopping and promise to take the customer experience to the next level.

1.2 Motivations

Virtual try-on technology is a method to help customers visualize how a garment might look on their body based on an image of the person and an image of the desired item. This technology addresses a substantial challenge online shoppers encounter: having to visit a store to try on clothing items physically. According to a study of Coresight [29], the period from March 2022 to March 2023 saw a notable 24.4% average return rate for online clothing orders in the United States, which is equivalent to $38 billion in returns. Moreover, 53% of the respondents revealed that their primary reason for returns was the ill-fit of that garment during try-on ([29]).

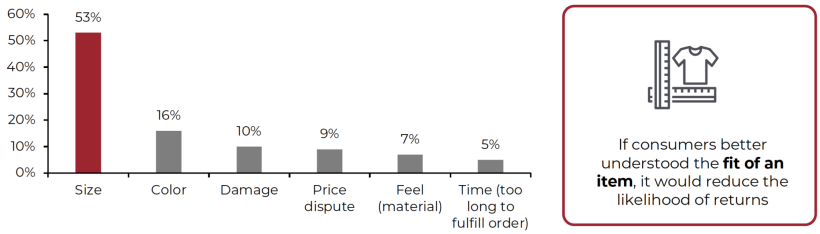


Figure 1.2: Reasons for garment returns from March 2022 to March 2023 (% of Respondents)

As a result, there is a growing interest in virtual try-on, demonstrating the potential to enhance shopping experiences by integrating Augmented Reality (AR). However, existing works on image-based virtual try-on mostly do not put their concern about the complexity of their models, resulting in challenges when applying them in real-time scenarios. This motivates us to investigate the potential of a faster and more resource-efficient virtual try-on method while ensuring the fidelity of the outcomes.

Fashion recommendation has always been crucial to modern fashion e-commerce systems due to its numerous benefits. According to a study of ViSenze [30],

recommendations can increase conversion rates by up to 300% and increase revenue by up to 31%. It saves users valuable time and effort by curating multiple sets of items that align with their preferences and style. Instead of spending hours searching for fashion items online or in physical stores, users can rely on these recommendations to discover new and relevant clothes. Moreover, these recommendations help users explore a wider range of options, introducing them to different styles or brands. This encourages users to try out something they might not have considered before, thus improving the fashion industry's overall revenue. Additionally, as 71% of consumers feel frustrated when their shopping experience is impersonal [30], fashion recommendation can simplify decision-making and increase user satisfaction by presenting curated options tailored to individual preferences. Ultimately, fashion recommendation has proven its usefulness by assisting users in discovering new styles, saving time, and making informed fashion choices.

1.3 Objectives and Main Contributions

The primary objective of this thesis is to investigate and address two significant challenges in the field of fashion e-commerce: virtual try-on and fashion recommendation. By leveraging intelligent systems and cutting-edge method technologies, we aim to develop innovative solutions that enhance the online shopping experience and bridge the gap between the physical and digital realms of fashion retail.

To address the previous virtual try-on method issues, we aim to propose a framework that can achieve faster run time and less memory consumption while producing results of the same quality. Thus, it can pave the potential of integrating image-based virtual try-on into real-time AR scenarios.

Regarding the fashion recommendation problem, we focus on taking a fashion

item as the reference item and recommending based on the visual features of that item. This research explores multiple approaches to produce meaningful results for such a problem. These approaches include intra-category similar item retrieval, inter-category complementary item retrieval, and text feedback-guided item retrieval. We also study incorporating approximate searching methods into these approaches to enhance the system's overall performance.

Another important objective of this thesis is to create an intelligent system for online shopping support, including web and AR applications. We aim to combine the power of fashion recommendation and virtual try-on in this system, offering new possibilities for personalized and immersive interactions between consumers and online fashion platforms.

The main contribution of this thesis can be summarized as follows:

- Propose a virtual try-on framework based on knowledge distillation learning. We can achieve a lightweight network, reducing inference time and memory consumption, thus making it easier to deploy and operate on AR devices.

- Introduce an automatic fashion-pose data generation pipeline designed to enrich existing fashion datasets by synthesizing new person poses from a single image of that person.

- Investigate various ways to recommend fashion items from a reference image. These approaches include retrieving intra-category similar items, inter-category complementary items and text feedback-guided items. We also propose Outfit Retrieval Transformer to address the complementary item retrieval task.

- Incorporate approximate searching for retrieval, which runs significantly faster while producing nearly the same results as exhaustive methods.

- Build an AR application for virtual try-on, which can reduce the risk of damage to real clothes for shops.
- Develop a smart fashion assistant web application tailored for the fashion e-commerce industry, which integrates virtual try-on functionality and fashion recommendation capabilities, enhancing the overall user experience and increasing customer satisfaction in the fashion e-commerce sector.

1.4 Thesis Organization

This thesis is organized into 7 chapters:

Chapter 1: This chapter overviews e-commerce within fashion retail, focusing on the challenges associated with personalized and immersive experiences. We first discuss two significant online shopping problems, which involve virtual try-on technologies and fashion recommendation systems, and investigate various approaches to tackle those problems. Next, we describe the main contributions of this thesis to bridge the gap between physical and online fashion retail. Finally, the organizational structure of the thesis is presented, offering a concise overview of the subsequent chapters and their contents.

Chapter 2: This chapter provides an overview of the fundamental knowledge relevant to our thesis. We present a list of Computer Vision problems that play important roles in our proposed approach.

Chapter 3: This chapter presents an overview of the literature in the virtual try-on and fashion recommendation fields. In the domain of virtual try-on, we introduce recent image virtual try-on approaches. Besides, some commercial products and their technology are discussed to provide a comprehensive view of this field. Moving on to fashion recommendation, we introduce related works centering on visual-based fashion recommendations. Finally, we discuss techniques for similarity search, including exhaustive and approximate methods applied to our

recommendation approaches.

Chapter 4: In this chapter, we propose Distilled Mobile Real-time Virtual Try-On (DM-VTON), which focuses on synthesizing try-on images with increased speed compared to previous methods while ensuring accuracy. Our approach is based on a knowledge distillation scheme that leverages a strong Teacher network as supervision to guide a Student network without relying on human parsing. Notably, we introduce an efficient Mobile Generative Module within the Student network, significantly reducing the runtime while ensuring high-quality output. Additionally, we propose Virtual Try-on-guided Pose for Data Synthesis to address the limited pose variation observed in training images. Finally, we provide the experimental details of our proposed method, and then we present a comparative study of DM-VTON with state-of-the-art methods.

Chapter 5: In this chapter, we address the problem of recommending items given a reference fashion item. We carefully investigate three approaches to retrieving items: similar items within the same category, complementary items from other categories, and items guided by text feedback. In terms of retrieving intra-category similar items and text feedback-guided items, we employ a pretrained CLIP-based model and receive remarkable results. As for the inter-category complementary item retrieval, we consider it a Natural Language Process problem and propose Outfit Retrieval Transformers (ORT), which utilizes the Transformers architecture. Through experiments, ORT proves its effectiveness and can produce reasonable recommendations. Because using an embedding to query items from a dataset plays an important role in recommendation, we analyze various approximate searching methods and compare them with the exhaustive K-Nearest Neighbor algorithm regarding query time and accuracy.

Chapter 6: This chapter presents our applications that assist customers in e-commerce based on the methods presented, including Smart Fashion Assistant, a

system for online shopping support, and Magic Mirror, an application that allows users to try on clothing items virtually in an augmented reality scenario. We present an overview of each application, followed by the details of the conducted experiments, including a pilot study to evaluate their effectiveness and user satisfaction.

Chapter 7: We conclude our work by summarizing the results obtained and discussing the potential for future research directions. In this thesis, we solve two major problems in online fashion shopping: virtual try-on and fashion recommendation. Through intensive experiments, we prove the applicability and efficiency of our solutions and further demonstrate them in demo applications. The feedback we gained from the pilot study shows that our work still has some limitations, which we plan to improve in future works.

CHAPTER 2
BACKGROUND

This chapter provides an overview of the fundamental knowledge relevant to our thesis. We present a list of Computer Vision problems that play important roles in our proposed approach.

2.1 Convolution Neural Network

Convolutional neural network, also known as CNN or ConvNet, was introduced by Yann LeCun et al. [31] to understand and explain visual data such as images. A digital image contains pixels arranged in a grid to represent information about colour, brightness, texture, etc. The CNN network is inspired by how the human visual system processes data on each receptive field. CNN is commonly used for extracting feature maps from images containing information about the relationship between pixels. This information can be used as the input for the downstream tasks: image classification, object detection, etc.

2.1.1 Convolution Layer

The convolution Layer is the core block of a CNN. It extracts information from the input through matrix multiplications. Specifically, it performs multiplications between a set of learnable parameters, known as a kernel, and the corresponding matrix bounded by the receptive fields. In mathematics, the convolution between two functions [32], $f, g : \mathbb{R}^d \to \mathbb{R}$ is defined as:

$$(f * g)(x) = \int f(z)g(x - z)dz, \tag{2.1}$$

For two-dimensional matrices, we have the formula:

$$(f * g)(i, j) = \sum_{a}^{A-1} \sum_{b}^{B-1} f(a, b) g(i - a, j - b), \tag{2.2}$$

Where f is the kernel and g is the input image with size $A \times B$. An illustration of the formula is shown in Figure 2.1.

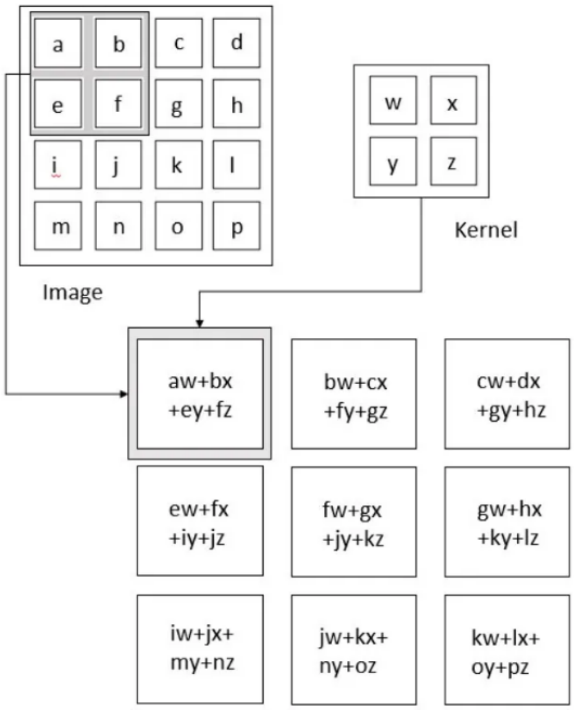


Figure 2.1: Convolution operation (Source: Deep Learning [9]).

Visually, the convolution window (kernel) is slid over all the height and width of images to compute the result at each spatial position. However, with this method, we can only shift the kernel pixel by pixel, leading to high computational costs when the input size is large. Stride and padding are techniques applied in convolution for computational efficiency and to control the downsampling rate. Stride is the number of elements that the window moves at a time. It enables to capture a larger receptive field. Padding is the extra pixels added around

the border of the input before performing the convolution operation to keep the input dimensions after convolution. If we have an input of size $W_{in} \times H_{in}$ and kernel of size F with stride S and padding P, then the size of output is $W_{out} \times H_{out}$ (as illustrated in Figure 2.2):

$$\begin{cases} W_{out} = \dfrac{W_{in} - F + 2P}{S} + 1 \\ H_{out} = \dfrac{H_{in} - F + 2P}{S} + 1 \end{cases} . \tag{2.3}$$

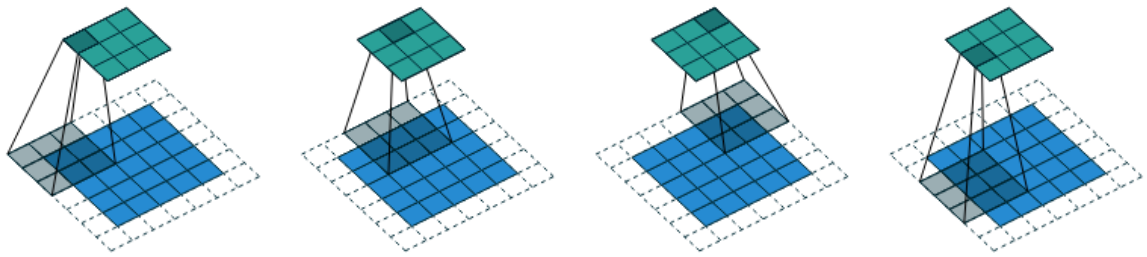


Figure 2.2: Two-dimensional convolution ($F = 3, S = 2, P = 1$) with input of size $W_{in} = H_{in} = 5$. After sliding the kernel over all the input, we get the output with size $W_{out} = H_{out} = 3$ (Source: A guide to convolution arithmetic for deep learning [10]).

2.1.2 Pooling Layer

The pooling layer is commonly inserted between successive convolution layers to reduce the spatial size of the feature maps by aggregating information from local regions. This help decreases the number of parameters and computation and prevents overfitting. The pooling operation uses a window slid over all the input to compute an output for each region. Unlike the convolution layer, the window of the pooling layer contains no parameters. Instead, the output of the regions in the window is determined by a unified function, such as the maximum

or the average of the neighbours in the pooling window area.

The pooling operation is processed independently on every depth slice of the input. If we have an input of size $W_{in} \times H_{in} \times D_{in}$ and a pooling window with size F and stride S, the pooling layer produces an output of size $W_{out} \times H_{out} \times D_{out}$ (D_{in}, D_{out} are the depth size) where:

$$\begin{cases} W_{out} = \dfrac{W_{in} - F}{S} + 1 \\ H_{out} = \dfrac{H_{in} - F}{S} + 1 \\ D_{out} = D_{in} \end{cases} . \tag{2.4}$$

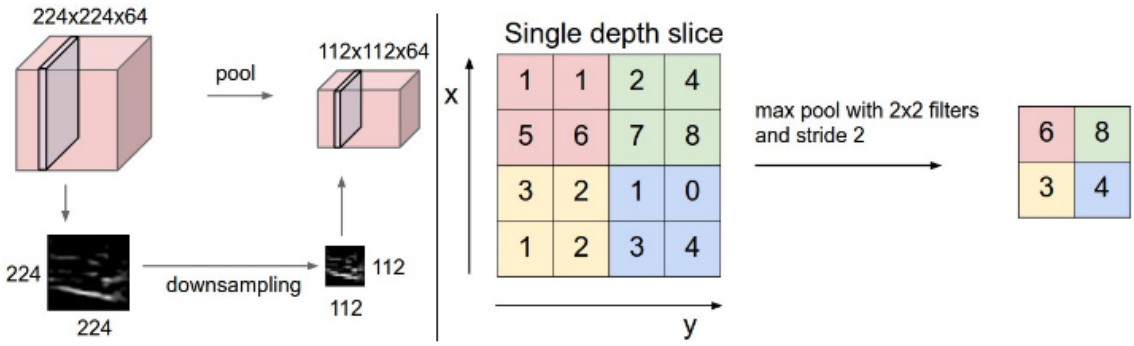


Figure 2.3: Pooling operation downsample by applying the kernel to each slice. **Left**: Appling pooling operation with $F = 2, S = 2$ reduces the input size from $224x \times 224 \times 64$ to $112x \times 112 \times 64$. **Right**: Illustration of the max pooling with $F = 2, S = 2$, which preserves the maximum value in adjacent pixels (Source: CS231n: Deep Learning for Computer Vision [11]).

2.2 Attention and Transformer

2.2.1 Self-attention

Self-attention is a sequence-to-sequence operation: the input is a sequence of vectors that goes in, and the output is also a sequence of vectors. This operation is proposed in the paper "Attention is all you need" [12]. We denote the input vectors as $x_1, x_2, ..., x_t$ and the corresponding output vectors $y_1, y_2, ..., y_t$. All vectors have dimension k.

The self-attention operation takes a weighted average over all the input vectors, as in Equation 2.5. The weight values can be derived from the input vectors themselves, which is why this operation is called self-attention. Equation 2.7 shows the simplest option, the dot product.

$$y_i = \sum_j w_{ij} x_j, \tag{2.5}$$

$$w_{ij} = \frac{exp(w'_{ij})}{\sum_j exp(w'_{ij})}, \tag{2.6}$$

$$w'_{ij} = x_i^T x_j. \tag{2.7}$$

Modern deep-learning architectures further improve this operation by adding three kxk learnable matrices, denoted as W_q, W_k, and W_v. We apply these matrices as the linear transformations of the input vectors x_i. Equation 2.8 shows the specific formula.

$$y_i = \sum_j w_{ij} (W_v x_j), \tag{2.8}$$

$$w_{ij} = \frac{exp(w'_{ij})}{\sum_j exp(w'_{ij})}, \tag{2.9}$$

$$w'_{ij} = (W_q x_i)^T (W_k x_j). \tag{2.10}$$

2.2.2 Multi-head Self-attention

Using a single self-attention operation, each input vector x_j can only influence every output vector y_i in a fixed way, depending on the transformation matrices W. To tackle this problem, Vaswani et al.[12] combine multiple self-attention mechanisms (indexed with r), which involve learning different matrices W_k^r, W_q^r, and W_v^r. These are called attention heads h. Each attention head gives different output vectors y_i^r. These vectors are concatenated along all attention heads and passed through a linear transformation to reduce the dimension to k. Figure 2.4 illustrates how multi-head self-attention works.

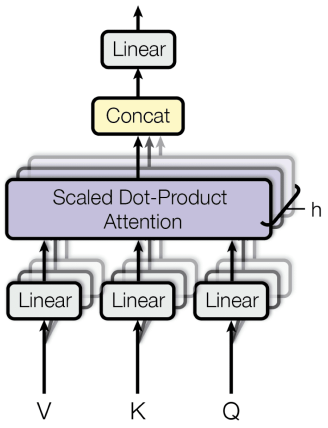


Figure 2.4: Multi-head self-attention mechanism (Source: Attention is all you need [12]).

However, this approach has one drawback, which takes h times slower as the number of heads increases. Multi-head self-attention solves this problem by slic-

ing the input vectors into h chunks and performing the multi-head self-attention on each chunk. For example, if we have 640-dimension input vectors and 16 attention heads, we need to slice each input vector into 16 chunks of 40 dimensions.

2.2.3 Transformers

Each transformer's block is built upon a multi-head self-attention block, followed by a normalization layer, a single multi-layer perception applied independently to each input vector, and another normalisation layer. Residual connections are added before both normalizations. Figure 2.5 shows the architecture of a Transformers block.

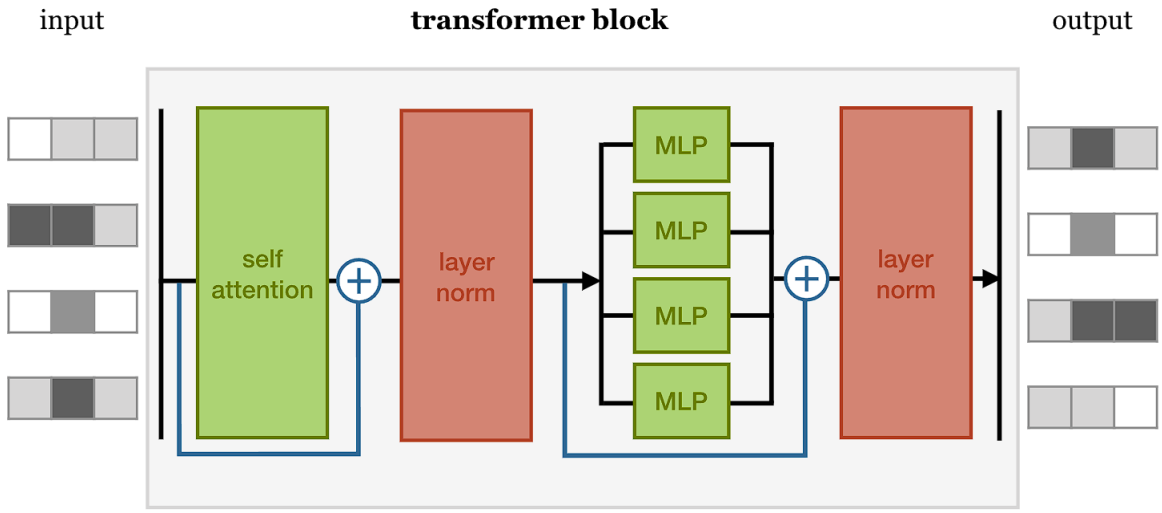


Figure 2.5: Transformers block architecture (Source: Transformers from scratch [13]).

A **Transformers block** can produce a sequence of vectors given a sequence of vectors, so it is also a sequence-to-sequence model. Modern deep-learning architectures [12, 33] utilize this characteristic and apply multiple Transformers blocks sequentially, called **Transformers Encoder**, to improve the overall performance.

Especially, in BERT [33], the author uses a special vector $<CLS>$ as the first input vector. The Transformers Encoder is trained so that the first output vector captures the semantics of the whole sequence of input vectors (Figure 2.6). After being proposed, this approach is widely adopted for downstream tasks, such as classification, which require one embedding representing the whole input.

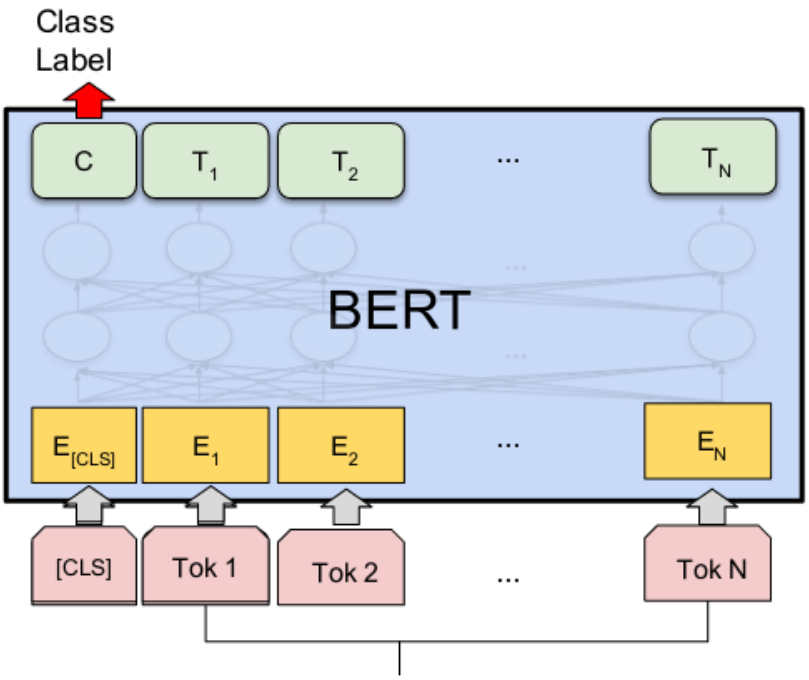


Figure 2.6: BERT $<CLS>$ vector (Source: Text classification using BERT [14]).

CHAPTER 3
LITERATURE REVIEW

This chapter presents an overview of the literature in the virtual try-on and fashion recommendation fields. In the domain of virtual try-on, we introduce recent image virtual try-on approaches. Besides, some commercial products and their technology are discussed to provide a comprehensive view of this field. Moving on to fashion recommendation, we introduce related works centering on visual-based fashion recommendations. Finally, we discuss techniques for similarity search, including exhaustive and approximate methods applied to our recommendation approaches.

3.1 Virtual Try-on

This section presents the virtual try-on approaches, divided into two main parts. In the first one, we discuss the research on image-based virtual try-on, which aim to warp the garment to fit the target person's pose. After that, we introduce some commercial products that are being applied in practice and their limitations.

3.1.1 Image-based Virtual Try-on

Image-based virtual try-on techniques can be classified into two categories: parser-based and parser-free approaches, in the sense that they use a body-parser map in the inference process or not. Both of them typically involves three steps: extracting the intrinsic input features, warping the input garment to fit the clothing area of the person image, and performing the replacement using a generative model.

Parser-based Methods. As for parser-based virtual try-on methods, they require human representation, including the human segmentation map, human pose, etc., to deform and fit the garment to the corresponding region on the input image. Previous methods [1, 2, 3, 4, 15, 16] utilized pretrained models such as human parsing [34, 35], pose estimation [36], densepose [37] to extract information (i.e. human segmentation map) from the person input used for both training and inference.

The initial approach that laid the foundation for this method was VITON [1], which proposed a coarse-to-fine framework. Firstly, a multi-task encoder-decoder generator is employed to generate a coarse sample and a clothing mask. Subsequently, the garment is warped using a thin plate spline (TPS) transformation with shape context matching. Finally, the coarse results are refined using a refinement network that leverages realistic details from a deformed item. Then it was realized that using image descriptors for warping had shown lower accuracy than deep learning networks. CP-VTON [2], building upon VITON [1], adopted a similar architecture but utilized an end-to-end neural network to learn the transformation parameters of the TPS transformation. This improvement significantly enhanced the accuracy of the virtual try-on outcomes.

Nonetheless, VITON [1] and CP-VTON [2] only focus on deforming the target garment during the virtual try-on process. Consequently, these approaches lead to losing important details related to the human body when try-on. To mitigate this issue, Yang et al. [4] proposed ACGPN, a framework that involves transforming the body-parser map to align with the target garment. Subsequently, the transformed parser map, input person image, and the deformed garment are utilized in the try-on generation. This integrated strategy ensures that the resulting try-on outcome effectively preserves the characteristics of both the human body and the in-shop clothes, enhancing the realism and fidelity of the try-on result.

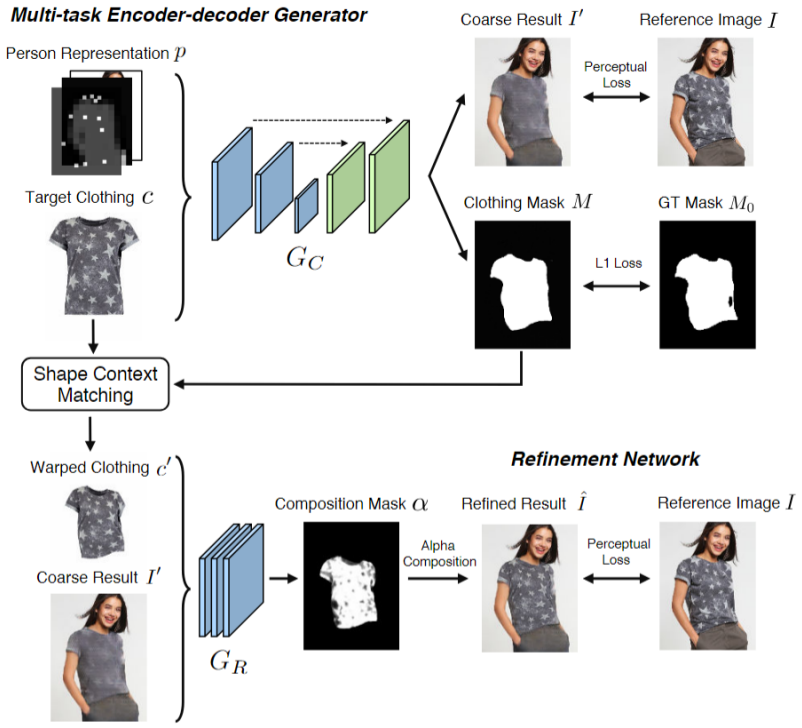


Figure 3.1: Overview architecture of VITON [1].

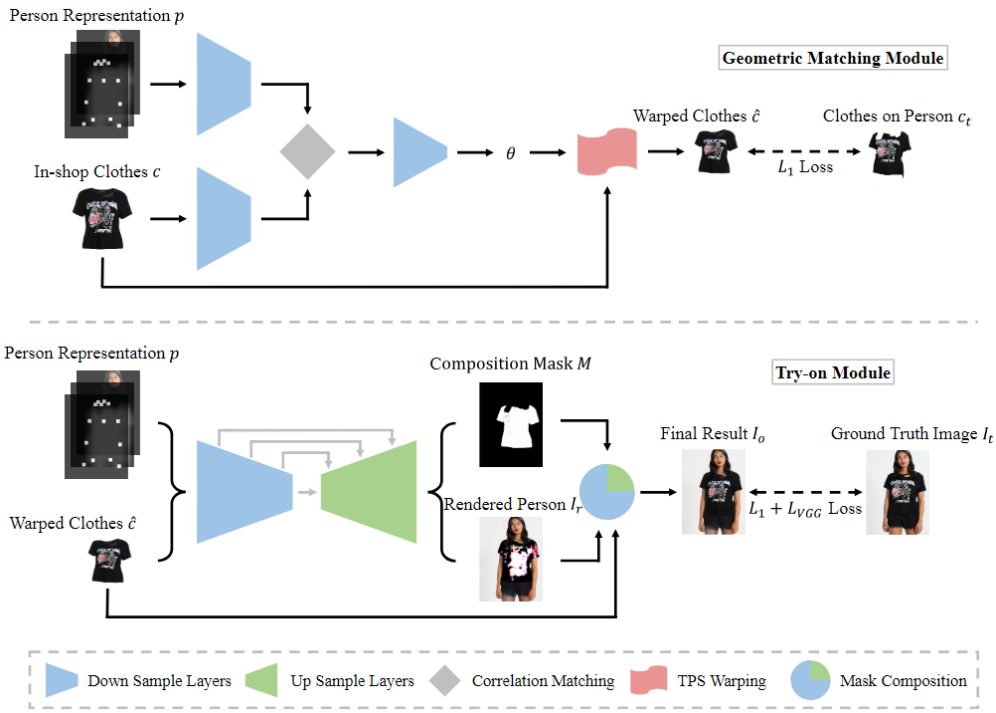


Figure 3.2: Overview architecture of CP-VTON [2].

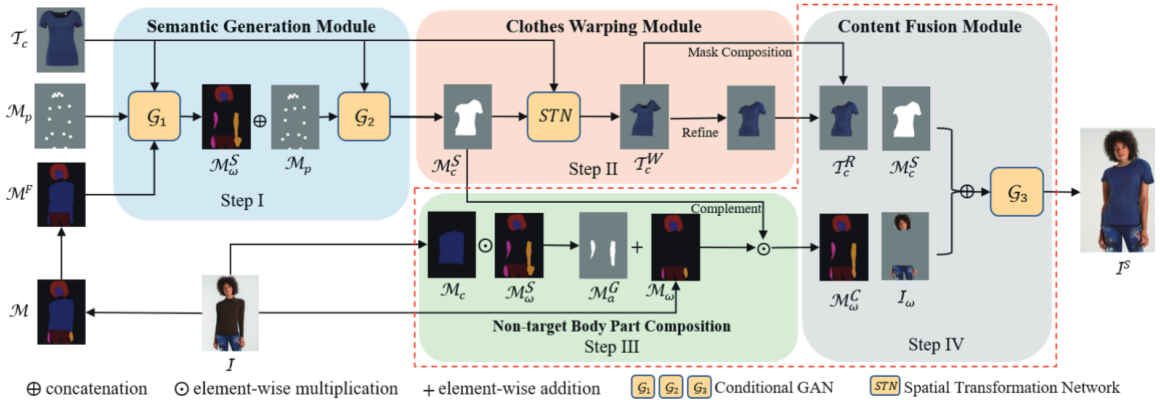


Figure 3.3: Overview architecture of ACGPN [4].

Han et al. [3] introduced ClothFlow, a framework designed for synthesizing clothed person images, serving both posed-guided person image generation and virtual try-on tasks. First, similar to ACGPN [4], ClothFlow generates a segmentation map of the target person. Subsequently, a flow estimation network predicts the appearance flow [27] that facilitates the warping of the source image to the desired target view. This approach yields more natural results compared to traditional TPS transformation. Finally, the final synthesized output is obtained by combining the source image, warped image, target segmentation map, and target pose. The ClothFlow framework for pose-guided person image generation is illustrated in Figure 3.4. For the virtual try-on tasks, the authors treated the garment image as the source image, and the input person pose as the target pose. The garment is then deformed by leveraging the predicted flow between the garment and the corresponding region of the person. The idea of using appearance flow instead of TPS transformation for deformation has been further developed and adopted by recent state-of-the-art methods [6, 7].

Following the direction of multi-stage approaches, Fele et al. [15] introduced a Context-driven Virtual Try-on Network (C-VTON). This framework comprises two fundamental components: a Body-Part Geometric Matcher, designed to deform the garment, and a Context-Aware Generator, aiming at generating the

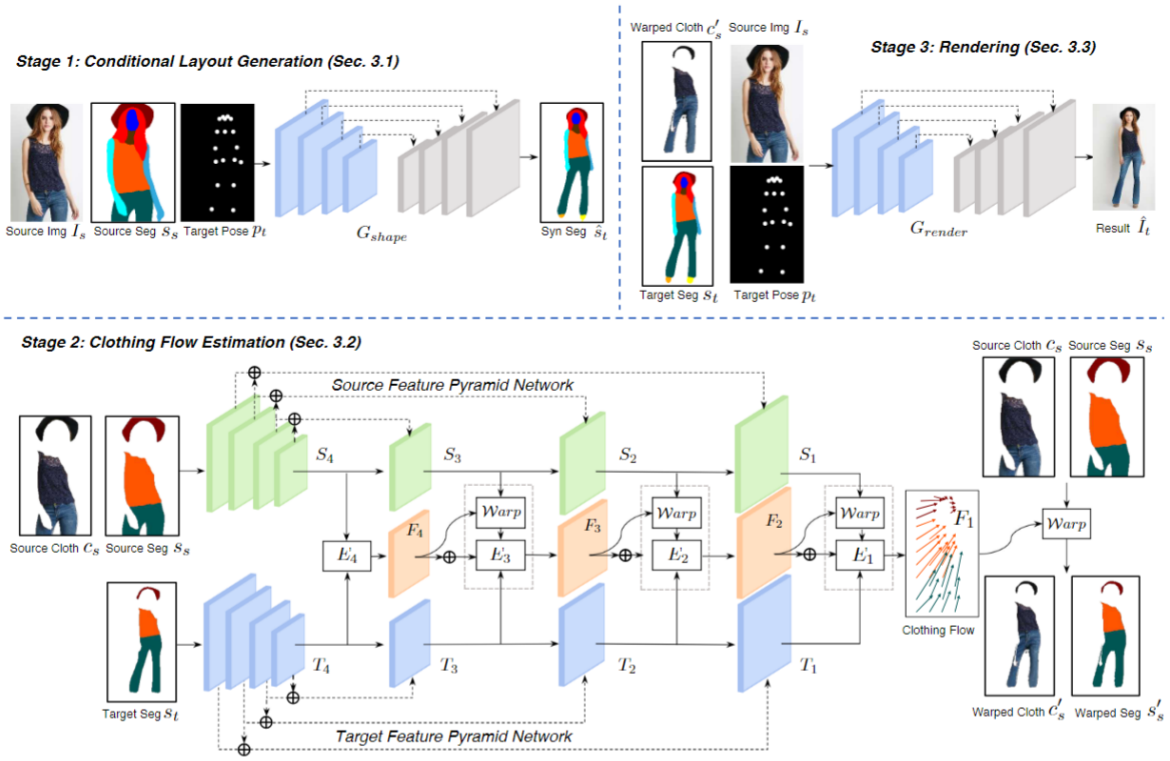


Figure 3.4: Overview architecture of CLothFlow for pose-guided person image generation [3].

try-on image with contextual information.

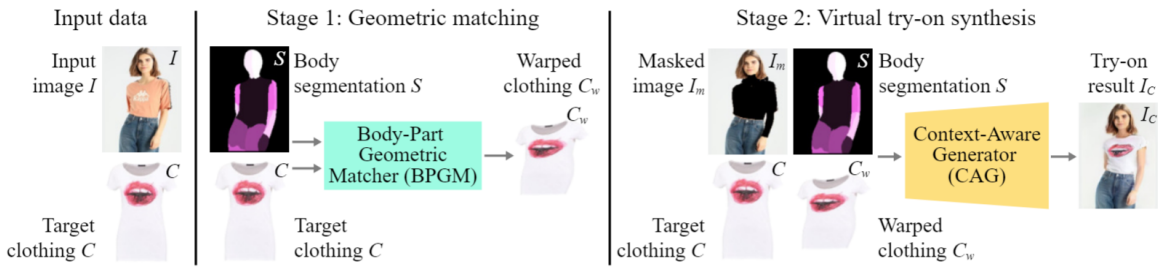


Figure 3.5: Overview architecture of C-VTON [15].

However, the multi-stage approaches generate a try-on image relying on intermediate prediction results, which may be inaccurate, ultimately giving rise to noticeable artifacts in the final output. To address this issue, Bai et al. [16] introduced a novel Single Stage Virtual Try-on framework (SDAFN), which only

uses a target pose map as guidance. By developing the deformable attention flow network and the shallow encoder and decoder, SDAFN allows the generation of try-on images end-to-end. The overall framework is shown in Figure 3.6.

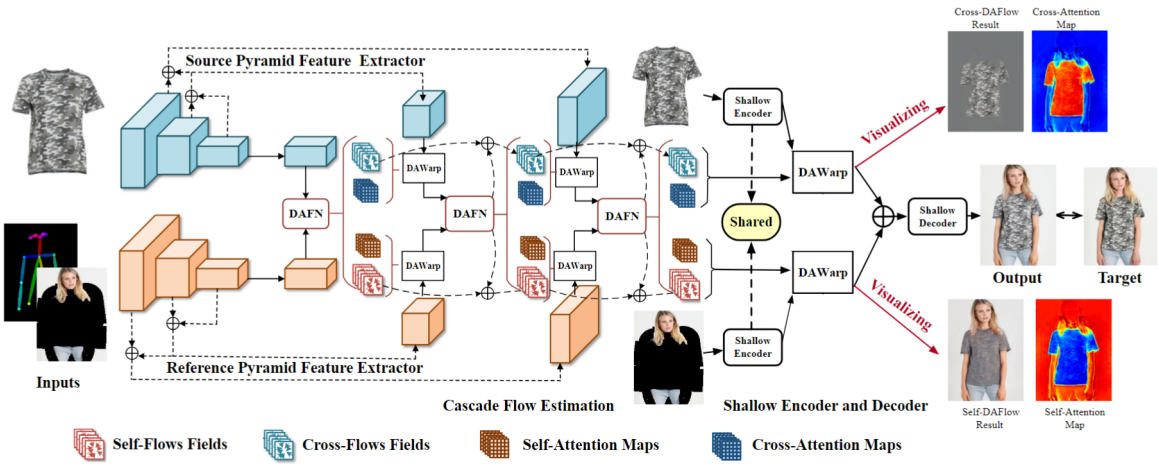


Figure 3.6: Overview architecture of SDAFN [16].

Parser-free Methods. In contrast, the parser-free approach only requires a person image and an input garment for inference. Consequently, this speeds up the process and eliminates reliance on human representation estimation models during inference. WUTON [5], as the pioneering parser-free method, introduced a Teacher-Student framework. It leverages a pretrained parser-based Teacher network to synthesize fake images acting as ground truth for guiding the training of the parser-free Student network.

WUTON [5] faces a specific limitation in which the optimization of the Student network is aimed at reconstructing synthetic images produced by the Teacher network. However, the result of the Teacher network has noticeable artifacts, resulting in a decline in the quality of the Student network. Ge et al. [6] presented a novel knowledge distillation-based approach (PF-AFN) to overcome this challenge. In this approach, the parser-free Student network uses the synthetic

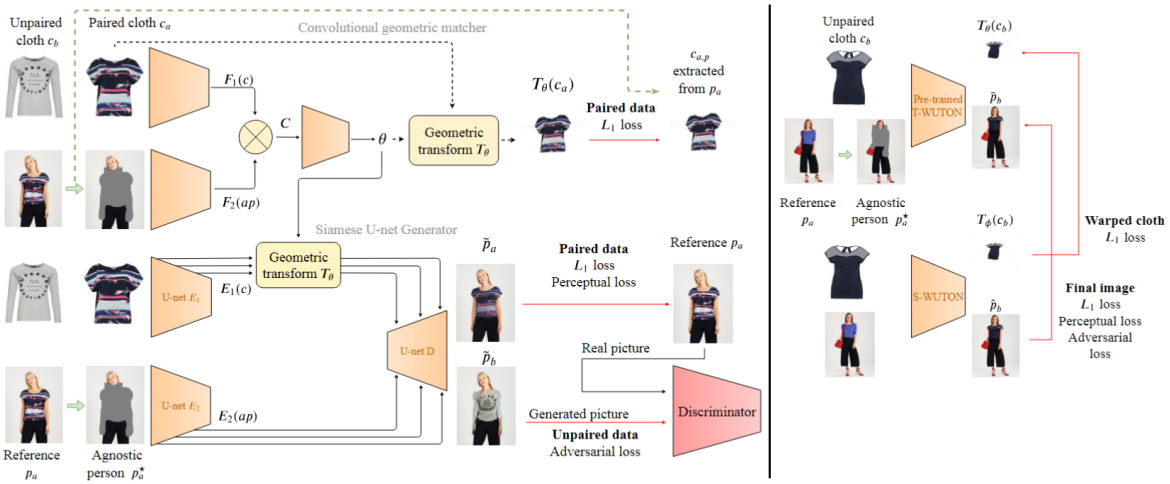


Figure 3.7: Overview architecture of WUTON [5].

images generated by a parser-based network as input and learns to reconstruct the original images. By doing so, the training process of the Student network is supervised by real image data. This novel training method has since become the standard for subsequent parser-free methods.

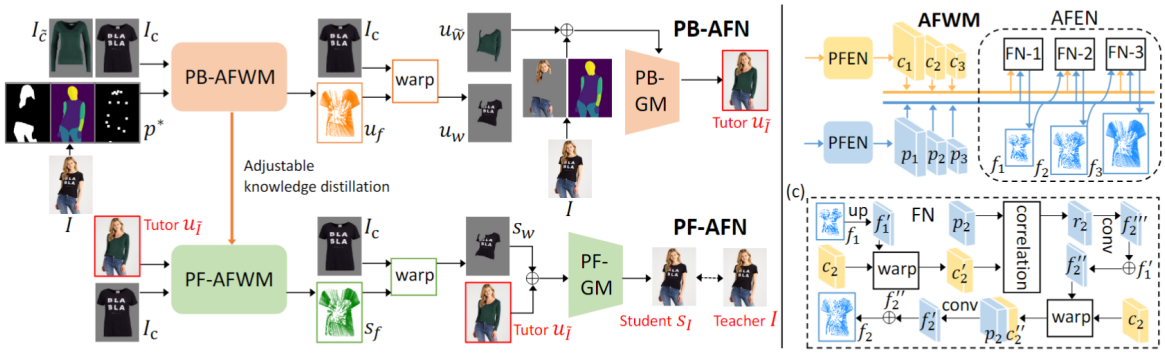


Figure 3.8: Overview architecture of PF-AFN [6].

He et al. [7] proposed FS-VTON, an enhanced framework derived from PF-AFN [6] with specific improvements in the warping component. The authors introduced a Style-based Global Appearance Flow Estimation using StyleGan blocks [38]. It enables the model to capture global and local correspondences between clothes and person images, enhancing garment deformation. In contrast,

RMGN [17] directed its focus towards the generation module through the development of a Regional Mask-Guided Generator, built upon SPADE blocks [39] and residual connection.

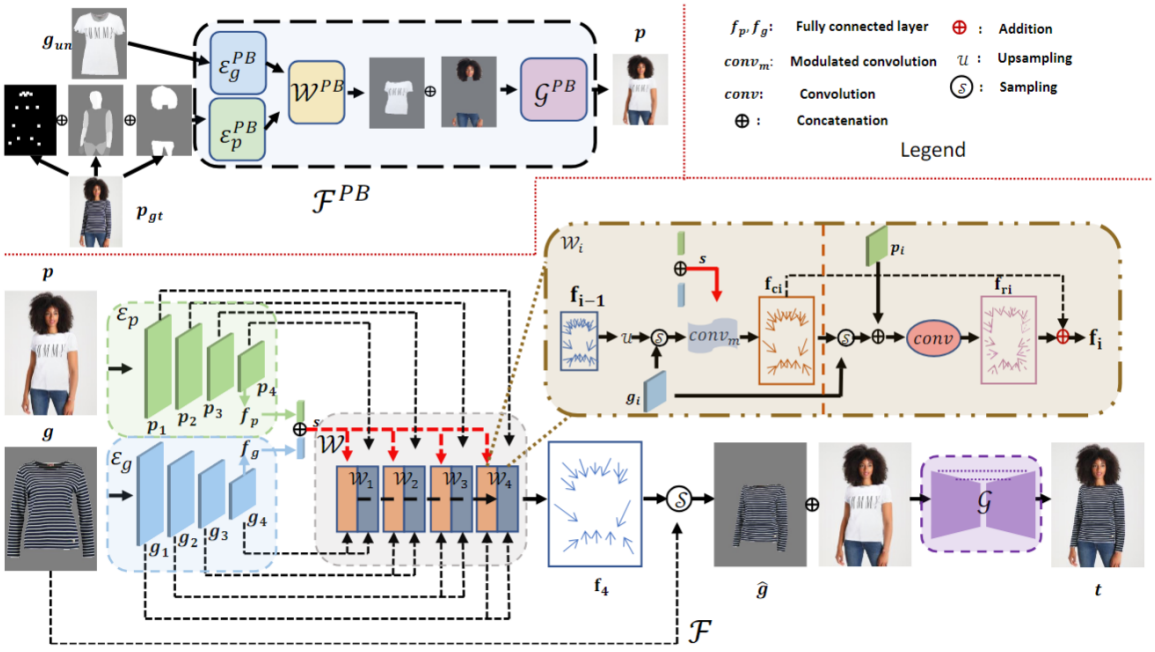


Figure 3.9: Overview architecture of FS-VTON [7].

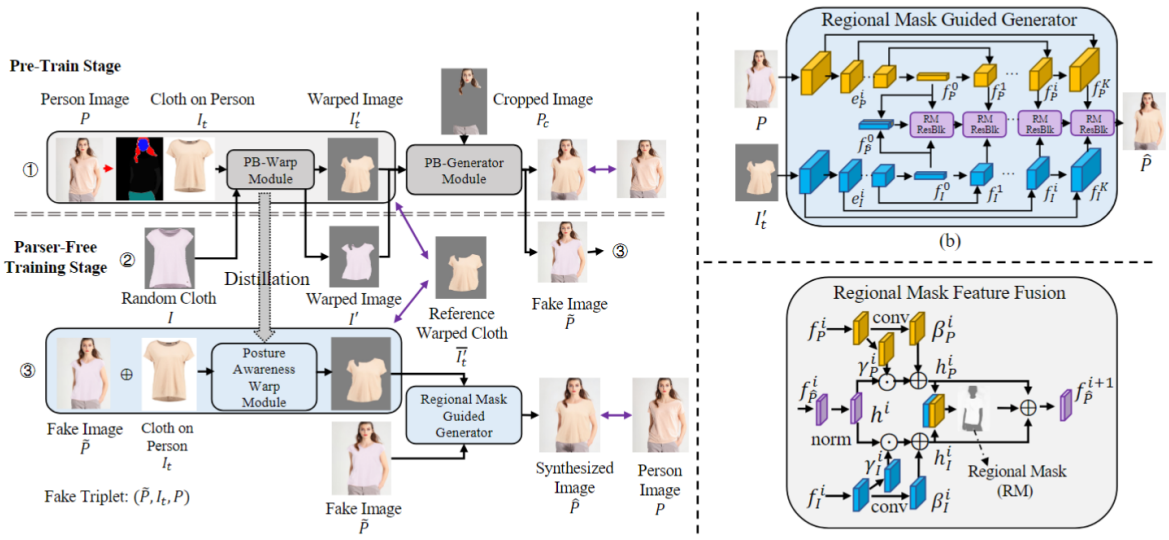


Figure 3.10: Overview architecture of RMGN [17].

In this thesis, we adopt the parser-free approach to prioritize speed, as calculating human representation is a time bottleneck in the try-on process. However, we take a distinct step from existing methods by modifying the Student network for improved speed and reduced memory consumption while keeping the parser-based approach in the Teacher network to preserve the output quality.

3.1.2 Commercial Products

Companies are experiencing significant advancements in virtual try-on, offering many competitive advantages. Artificial intelligence (AI) and/or augmented reality (AR) are the technologies behind this technology. Such products are not limited to online platforms; even offline store owners can set up on-site kiosks or virtual mirrors, enabling customers to try on items without risking damage to the actual product. In this part, we explore several commercial products already used by companies.

One example of a virtual try-on product is the virtual try-on for fashion accessories, exemplified by Warby Parker's pioneering initiative to provide customers with a virtual eyewear try-on experience [40]. Offering this feature on their website allows customers to experiment with different glasses styles before purchasing. This demand becomes even more important for high-end and costly products. Recognizing this, Baume & Mercier, a luxury watch company, has enhanced their digital sales experience by incorporating a virtual try-on system [18]. Their method involves creating 3D watch models in various wrist sizes and striving to replicate transparency and lighting effects accurately [41].

Another awesome application is the virtual try-on for shoes, which leverages AR technologies to visualize how shoes appear on the user's feet. Some of the products that can be mentioned are Amazon virtual try-on for shoes [42] and Artlabs [43], which can fit 3D shoe models onto the customer's feet. Conse-

Figure 3.11: Virtual try-on with 3D watch models (Source: Baume & Mercier Virtual Try-On System [18]).

quently, customers can assess the appearance of the shoes from multiple angles by moving their feet.

The virtual try-on for clothes is another challenge. The methods employed in this domain aim to fit the clothes to suit the target human pose. Compared to the 3D-based solutions, which wear the 3D model of the item to the customer's body [19] or allow users to create 3D models that represent them to try on [44], the 2d-based virtual try-on product (e.g. Google Shopping virtual try-on [20]), which use the real models and direct processing on the image, is intensively developing and applying over the years. However, Google Shopping's virtual try-on feature [20] only allows users to select from the available model images, which makes it difficult for customers who wish to evaluate the fit of items on their bodies.

Figure 3.12: Tracking human body to wear 3D clothes models (Source: Geenee AR [19]).

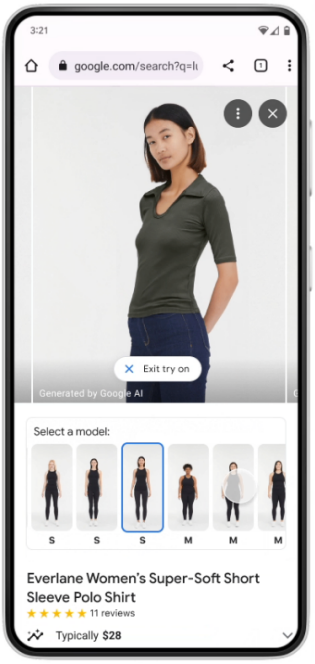


Figure 3.13: Google AI virtual try-on feature for shopping (Source: Google introduces new AI virtual try-on feature [20]).

3.2 Fashion Recommendation

With the explosion of data and deep learning, a growing number of studies have focused on visual-based fashion recommendations. In 2018, Mariya et al. [22] provided the PolyvoreOutfits dataset to support the fill-in-the-blank task, where we need to find an item of a specific category to complete a partial outfit.

Figure 3.14: Outfit fill-in-the-blank task, where the highlighted item is the missing item (Source: Fashion Outfit Complementary Item Retrieval [21]).

They also propose a Type-Aware network (Figure 3.15) as the baseline framework. Initially, the framework learns a shared embedding space where all fashion items are represented. Subsequently, this shared embedding space is projected into subspaces based on pairs of item types. For example, in the top-shoes subspace, the shoes that match a specific top must be close, even if they differ significantly in the overall embedding space.

After the Self-Attention mechanism and Transformers architecture [12] were introduced and gained success in the natural language processing task [33], CSA-Net [21] and OutfitTransformer [23] incorporated these ideas by utilizing the textual embeddings of the categories and significantly improved the recommendation results.

Specifically, CSA-Net represents each category by an embedding vector to capture their textual semantics. The CSA-Net framework (illustrated in Figure 3.16) takes a reference image, its category vector, and a target category vector. A

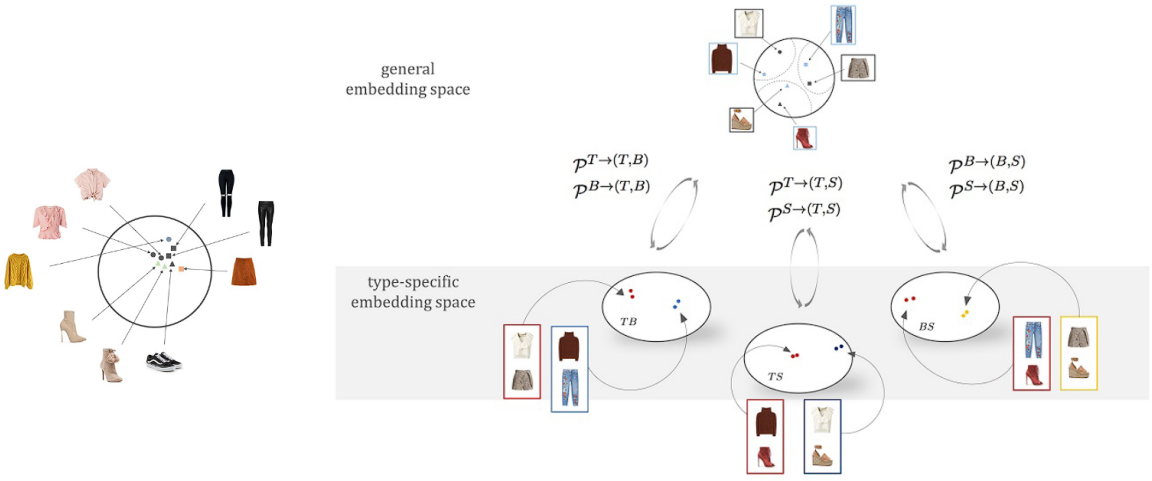


Figure 3.15: Type-Aware learning strategies (Source: Learning Type-Aware Embeddings for Fashion Compatibility [22]).

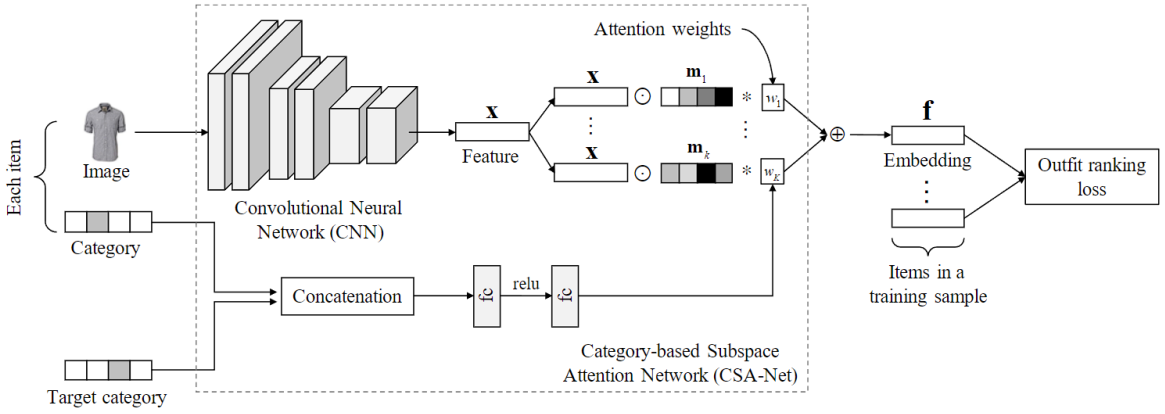


Figure 3.16: Overview of CSA-Net framework (Source: Fashion Outfit Complementary Item Retrieval [21]).

Convolution Neural Network is employed to extract the image embedding. This vector is then multiplied by a set of masks to create multiple subspace embeddings (multi-attention head). The concatenated category vector is used to predict the attention weights, which determine the contribution of each subspace embeddings in the final output embedding.

To eliminate the learning of pair-wise embeddings in the above methods, Out-

fitTransformer [23] learns the embedding of an entire outfit by utilizing the Transformer Encoder block and a special token at the first position (same as $<CLS>$ token in BERT [33]). This token not only captures the embeddings of all items in the outfit but also specifies the target category for the missing item (as shown in Figure 3.17).

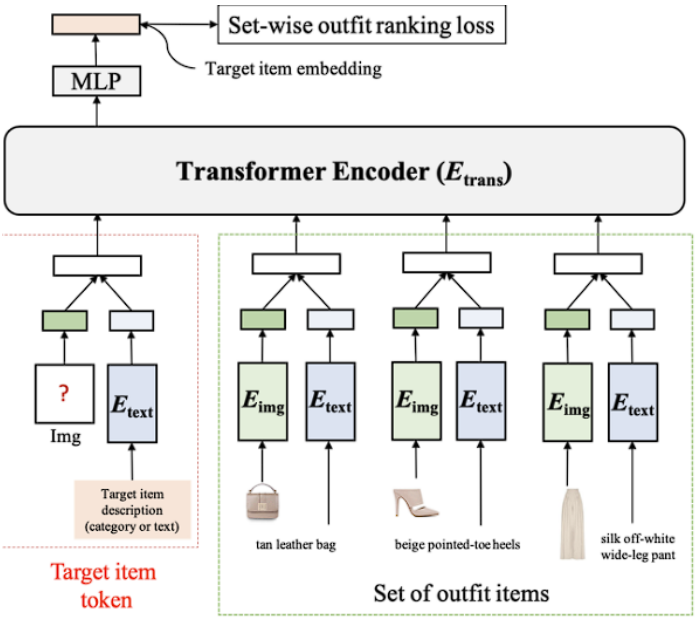


Figure 3.17: Overview of OutfitTransformer framework (Source: OutfitTransformer: Outfit Representations for Fashion Recommendation [23]).

Wu et al. [24] further expanded the research scope by introducing the FashionIQ dataset for the text feedback-guided item retrieval task, where we need to find the desired item given a reference input item and its relative attribute feedback in the natural language. Figure 3.18 demonstrates the retrieval pipeline based on the user's feedback.

By utilizing CLIP [45], a deep learning model that learns the joint representations of both images and textual descriptions, Baldrati et al. [46, 47] achieved state-of-the-art performance on the FashionIQ dataset, proving its applicability

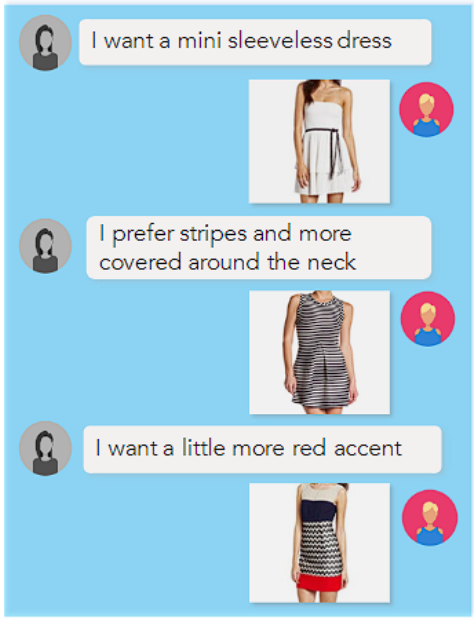


Figure 3.18: Demonstration of feedback-guided retrieval (Source: Fashion IQ: A New Dataset Towards Retrieving Images by Natural Language Feedback [24]).

to the fashion domain for text feedback-guided image retrieval.

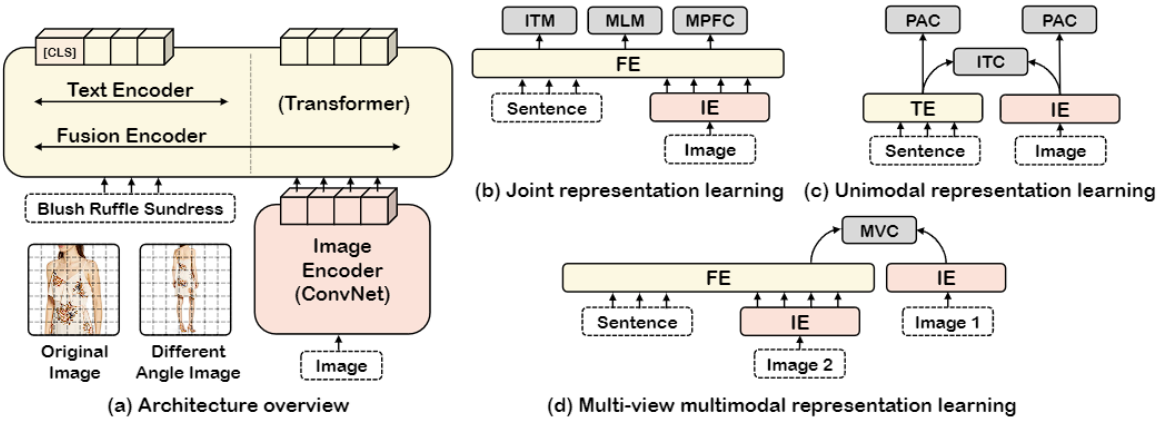


Figure 3.19: Overview of FashionViL architecture (Source: FashionViL: Fashion-Focused Vision-and-Language Representation Learning [25]).

Han et al. [25] bridged the gap between the text feedback-guided image retrieval problem and the fill-in-the-blank problem by proposing a general vision-

and-language network framework. The FashionViL overall architecture is shown in Figure 3.19. The framework optimizes its text and image encoders by learning multiple tasks such as Multi-view contrastive learning (MVC), Pseudo-attribute classification (PAC), Masked patch feature classification (MPFC), Image-text contrastive learning (ITC), Image-text matching (ITM), and Masked language modelling (MLM). Subsequently, the framework encoders can produce universal embeddings for downstream tasks such as text feedback-guided image retrieval or fill-in-the-blank problem.

3.3 Similarity Search

Similarity search has become a hot topic in recent years due to the explosion of data. There are two main types of similarity search: exhaustive and approximate. Exhaustive searching, such as the K-Nearest Neighbor (KNN) algorithm, involves calculating the distances between the query point and all items in a dataset to find the nearest neighbours. This approach provides accurate results but can be computationally expensive, especially in large datasets or high dimensional data points.

Approximate searching rises as the solution for the time complexity of exhaustive searching. These methods either reduce the dataset dimension or limit the search scope with the trade-off for accuracy. Production Quantization (PQ) [48] clusters each dimension or group of dimensions into buckets and indexes the dataset by those buckets instead, thus reducing the number of bytes required to represent the dataset (as illustrated in Figure 3.20).

In terms of limiting the search scope, Inverted File Index (IVF) [49] divides the search space into subspaces and only searches in a subset of these subspaces. Some methods leverage the tree [50] or graph [51] data structure to speed up the search process with the cost of more memory consumption. Specifically,

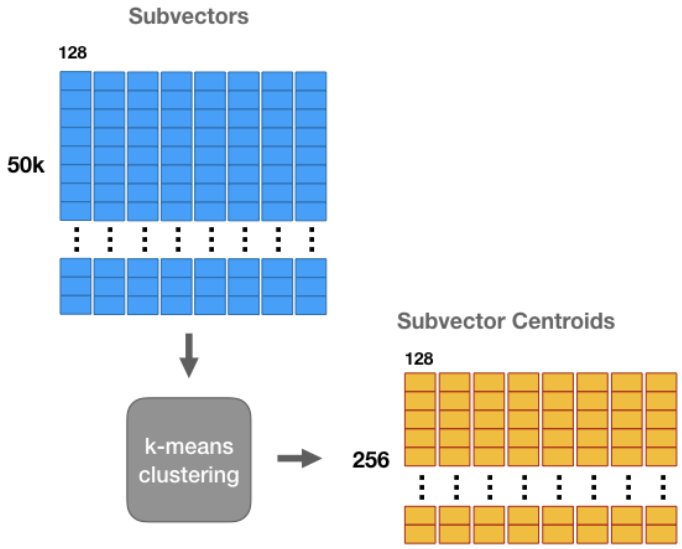


Figure 3.20: Overview of Production Quantization method (Source: Product Quantizers for k-NN Tutorial Part 1 [26]).

Hierarchical Navigable Small Worlds [51] constructs multiple subgraphs from all items; the lower the level is, the denser the subgraph becomes, and the searching process is performed like in a Skip List. Approximate Nearest Neighbors Oh Yeah [50] builds a binary tree by repeatedly choosing two random items and splitting the space into two subspaces; Spotify's recommendation system uses this algorithm.

CHAPTER 4
VIRTUAL TRY-ON

In this chapter, we propose Distilled Mobile Real-time Virtual Try-On (DM-VTON), which focuses on synthesizing try-on images with increased speed compared to previous methods while ensuring accuracy. Our approach is based on a knowledge distillation scheme that leverages a strong Teacher network as supervision to guide a Student network without relying on human parsing. Notably, we introduce an efficient Mobile Generative Module within the Student network, significantly reducing the runtime while ensuring high-quality output. Additionally, we propose Virtual Try-on-guided Pose for Data Synthesis to address the limited pose variation observed in training images. Finally, we provide the experimental details of our proposed method, and then we present a comparative study of DM-VTON with state-of-the-art methods.

4.1 Overview

Our objective is to generate an image of a person wearing a specific garment while preserving the rest of the image. To achieve this goal, we adopt the knowledge distillation training pipeline [52, 5, 6, 7, 17] to develop a Distilled Mobile Real-time Virtual Try-On (DM-VTON) framework (see Figure 4.1). Our proposed DM-VTON consists of two networks: Teacher and Student networks. Both include several main components: feature extractor, clothes-warping module, and generator.

The Teacher network aims to produce the virtual try-on result using the parser-based training process. The Student network then utilizes the Teacher network to generate synthetic input images, enabling the Student network to be supervised

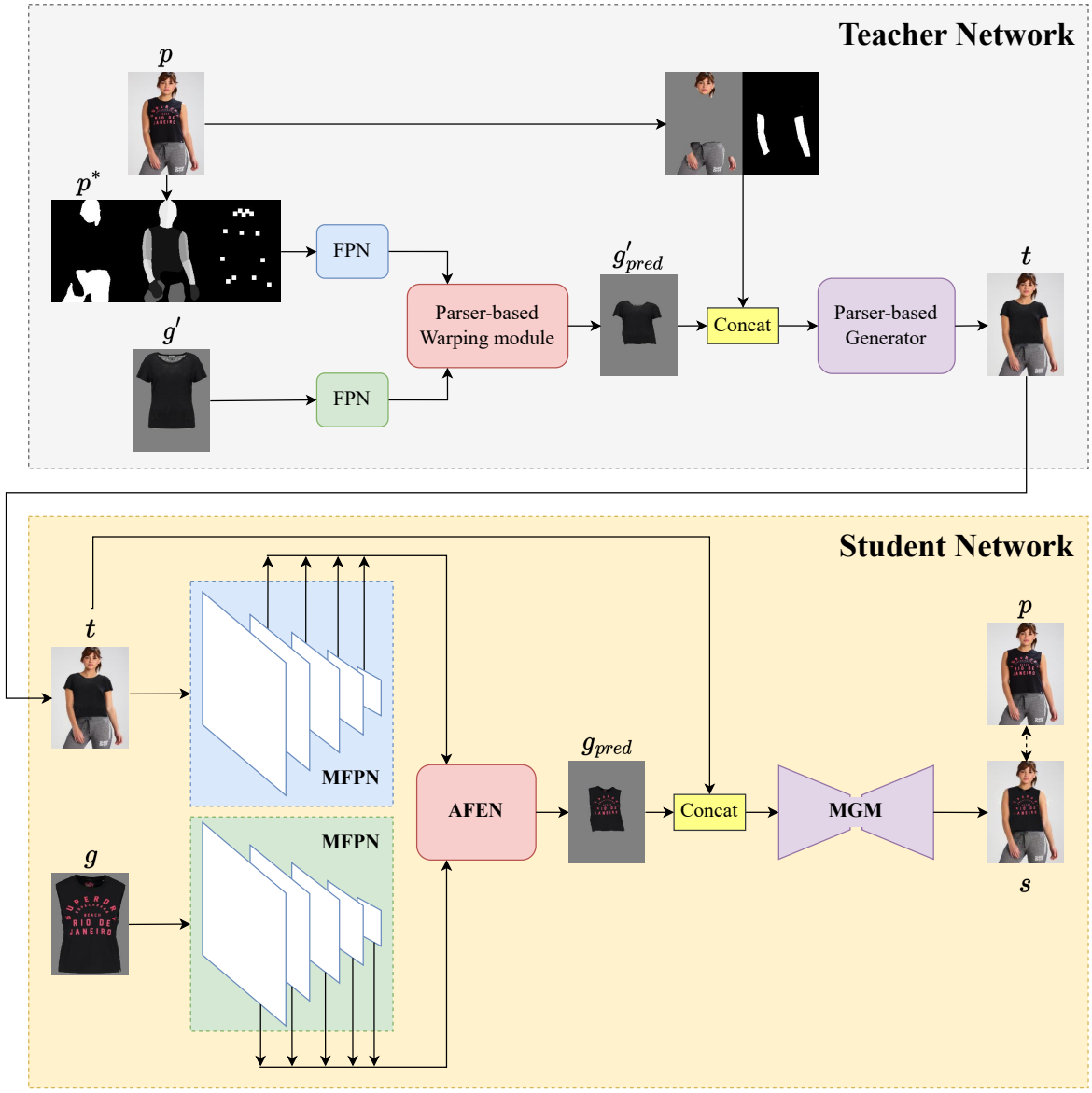


Figure 4.1: Overview architecture of the proposed Distilled Mobile Real-time Virtual Try-On (DM-VTON) framework. The parser-based Teacher network generates a synthetic image as the input for training the Student network.

by the original images without relying on human representation. The Teacher network is built upon SOTA virtual try-on models to ensure high-quality output. Focusing on inference speed, we propose lightweight components for the Student network.

4.2 Appearance Flow

The concept of appearance flow begins with a method for synthesizing images of the same object observed from arbitrary viewpoints introduced by Zhou et al. [27]. From the observation that the appearance (texture, shape, colour, etc.) of different views of an object are highly correlated, research suggests that the information of an input view can be used to generate images for various views. Appearance flow refers to 2-D coordinate vectors specifying which pixels in the input view could be used to synthesize the target view. Specifically, with the pixel i of the target image, the appearance flow vector $f^i \in \mathbb{R}^2$ indicates the coordinate of the input pixel sampled to reconstruct it (as illustrated in Figure 4.2). This idea is also applied to solve many problems, such as visual tracking [53], pose transfer [54], image inpainting [55], virtual try-on [6, 7]

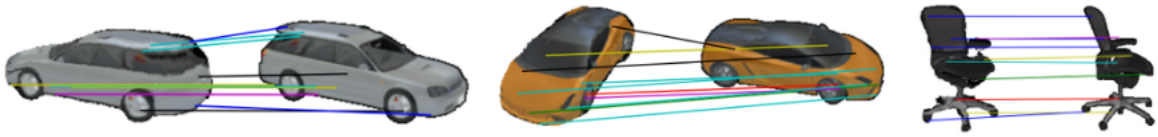


Figure 4.2: Illustration of appearance flow vectors (Source: View Synthesis by Appearance Flow [27]).

4.3 Teacher Network

The main purpose of this network is to generate a synthetic person image that serves as the input for the Student training process. Furthermore, the Teacher also helps this process through a knowledge distillation scheme. In particular, we take advantage of the SOTA method of virtual try-on task: FS-VTON [7]. As shown in Figure 4.1, it incorporates two feature pyramid networks (FPN) [56] constructed from residual blocks, enabling the extraction of features from the

human representation p^* and garment image g'. To achieve the garment deformation functionality, the Teacher network utilizes a style-based global appearance flow estimation network that uses modulated convolution [38]. This network first predicts a coarse appearance flow via extracted global style vector and then refines it locally. The last flow thus can capture the global and local correspondence between the garment and the target person. This makes the Teacher network more robust against the problems of detail-preserving and large misalignment. Finally, the warped clothes and the preserved region on the human body are concatenated as the generator input for try-on result generation. The generator of our Teacher network follows the encoder-decoder architecture with skip connections, which have been proven effective in detail preservation.

Because the inputs of the parser-based model (i.e., human representation) contain more semantic information when compared to those in the parser-free model, we employ an adjustable knowledge distillation learning scheme [6] with a distillation loss to guide the Student network. When training the Student network with fake image t and garment g, we also pass $p*$ and g through the pretrained Teacher network. The distillation loss is formulated as follows:

$$L_{dis} = \lambda_{fea} L_{fea} + \lambda_{flow} L_{flow}, \tag{4.1}$$

$$L_{fea} = \psi \sum_{i=1}^{N} (p_i^{pb} - t_i^{pf})^2 + \psi \sum_{i=1}^{N} (g_i^{pb} - g_i^{pf})^2, \tag{4.2}$$

$$L_{flow} = \psi \sum_{i=1}^{N} \| f_i^{pb} - f_i^{pf} \|_2, \tag{4.3}$$

$$\psi = \begin{cases} 1, \ if \ \|t-p\|_1 < \|s-p\|_1 \\ 0, \ otherwise \end{cases}, \tag{4.4}$$

where t, and s are the try-on result of the Teacher and Student, respectively; p is the person image ground truth. p_i^{pb} and t_i^{pf} denote the output feature maps at the i-th scale extracted from $p*$ and fake image t; similarly, g_i^{pb} and g_i^{pf} are the i-th

scale feature maps extracted from garment image g by the feature extractor of parser-based and parser-free network, respectively. f_i^{pb} and f_i^{pf} are the predicted appearance flows from the Teacher and Student warping modules at the i-th scale. ψ is the adjustable factor used to adjust so that the distilling process takes place only if the quality of the generated image of the parser-based network is better than that of the parser-free network.

4.4 Student Network

We propose a parser-based approach for synthesizing try-on images with increased speed compared to previous methods while ensuring accuracy. As shown in Figure 4.1, our Student network consists of three key components: Mobile Feature Pyramid Network (MFPN), Appearance Flow Estimation Network (AFEN), and Mobile Generative Module (MGM). These components synergistically collaborate to extract features, manipulate garments through deformation, and generate try-on images. The AFEN introduced by Ge et al. [6] proved effective in deforming garments by using appearance flow estimates from pyramid features. The MFPN and MGM are built upon the architecture of MobileNetV2 [57] with Inverted Residual blocks specifically designed to optimize computational efficiency and model size.

4.4.1 Mobile Feature Pyramid Network

As shown in Figure 4.3, MFPN incorporates the architecture of MobileNetV2 with Inverted Residual blocks [57] to a Feature Pyramid Network. By leveraging the capabilities of two MFPN blocks, we extract two-branch N-level feature maps from person and garment images within a parser-free network. These features are fed into the Appearance Flow Estimation Network (AFEN) to predict the appearance flow map for garment deformation.

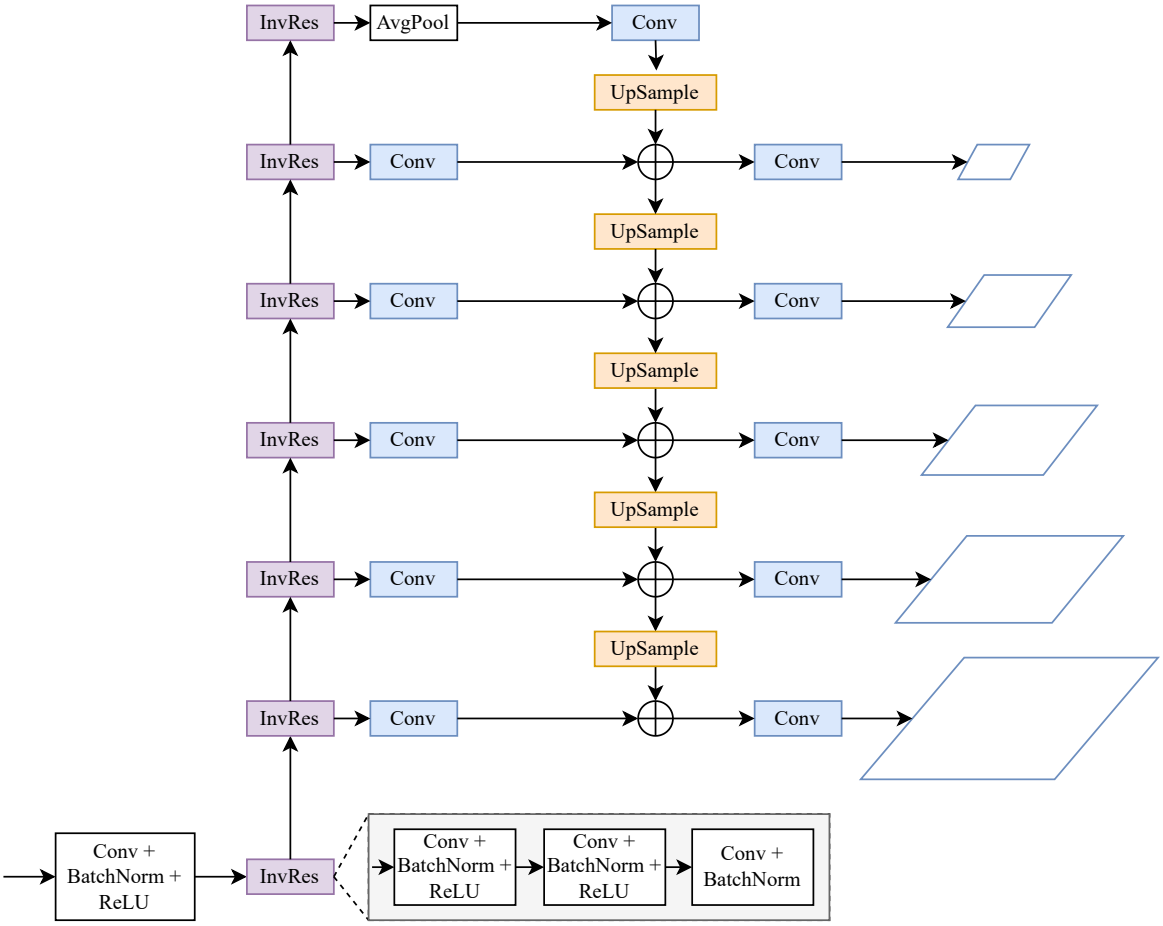


Figure 4.3: Mobile Feature Pyramid Network architecture

4.4.2 Appearance Flow Estimation Network

This component aims to deform the garment to fit the human pose while preserving the texture. Following the work of Ge et al. [6], we adopt an appearance flow estimation network (AFEN) comprising subnetworks equipped with varying sizes of convolution layers. These subnetworks are responsible for estimating flows based on extracted multi-level feature maps. The outcome of this network can capture the long-range correspondence between the garment image and the person image, effectively minimizing issues related to misalignment. To enhance the preservation of clothing characteristics, this module is optimized with the

second-order constraint:

$$L_{sec} = \sum_{i=1}^{N} \sum_{t} \sum_{\pi \in N_t} CharLoss\left(f_i^{t-\pi} + f_i^{t+\pi} - 2f_i^t\right), \tag{4.5}$$

where f_i^t denotes the t-th point on the i-th scale flow map; N_t is the set of horizontal, vertical, and diagonal neighborhoods around the t-th point; and $CharLoss$ denotes generalized Charbonnier loss [58].

4.4.3 Mobile Generative Module

To synthesize the entire try-on image from the warped image and target person image, we develop a Mobile Generative Module, the integration of the architectural principles of UNet [59] and MobileNetV2 [57] as illustrated in Figure 4.4. The primary objective behind the design of this generator is to reduce both the computational burden and the model's overall size.

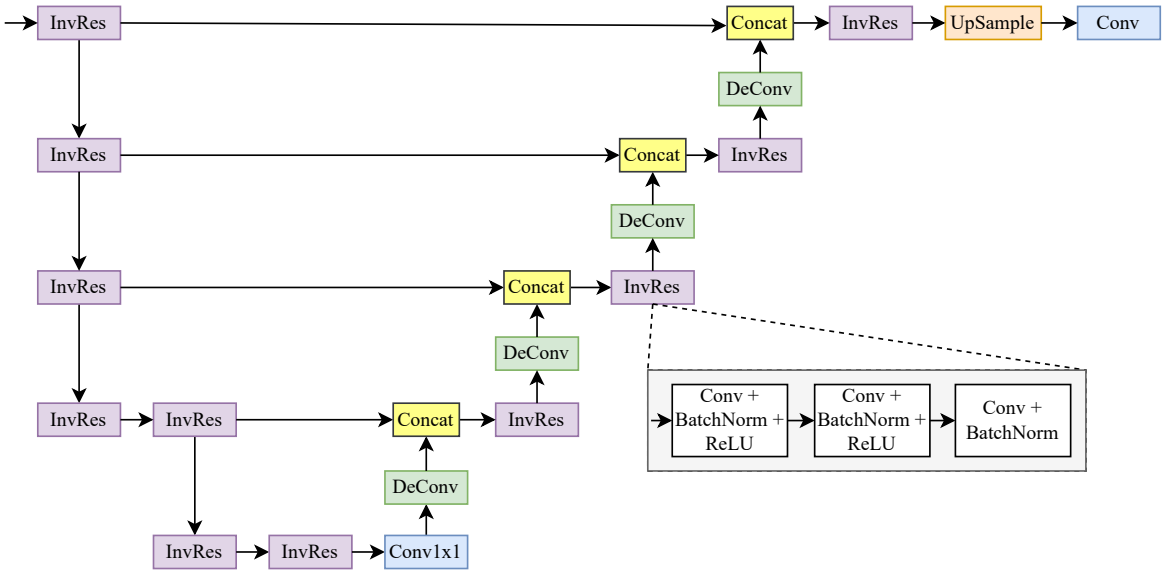


Figure 4.4: Mobile Generative Module architecture

4.4.4 Loss Function

During training, we optimize the warping module separately in the first stage and then train together with the generator in the last stage. The loss function used in the first stage is defined as:

$$L^{warp} = \lambda_l^{warp} L_l^{warp} + \lambda_{per}^{warp} L_{per}^{warp} + \lambda_{sec} L_{sec} + \lambda_{dis} L_{dis}, \tag{4.6}$$

$$L_l^{warp} = \|g_{pred} - p \odot m_{gt}\|, \tag{4.7}$$

$$L_{per}^{warp} = \sum_i \|\Phi_i(g_{pred}) - \Phi_i(p \odot m_{gt})\|, \tag{4.8}$$

where L_l^{warp} denotes pixel-wise L1 loss, L_{per}^{warp} is the perceptual loss [60], L_{sec}^{warp} is the smooth loss (second-order constrain), L_{dis}^{warp} is the distillation loss, g_{pred} is warped garment. p is the person image ground truth with the garment mask m_{gt}; Φ_i denotes the i-th block of pre-trained VGG19 [61].

With the generative module, we also apply L1 loss perceptual loss [60] between the synthesized image and the ground truth image to supervise the training process of MGM:

$$L^{gen} = \lambda_l^{gen} L_l^{gen} + \lambda_{per}^{gen} L_{per}^{gen}, \tag{4.9}$$

$$L_l^{gen} = \|s - p\|, \tag{4.10}$$

$$L_{per}^{gen} = \sum_i \|\Phi_i(s) - \Phi_i(p)\|, \tag{4.11}$$

where L_l^{gen} is L1 loss and L_{per}^{gen} is the perceptual loss [60]. s and p are the generated output of the Student network and the person image ground truth, respectively.

In practice, we empirically set $\lambda_l^{warp} = 1$, $\lambda_{per}^{warp} = 0.2$, $L_{sec}^{warp} = 6$, $L_{dis}^{warp} = 0.04$, $\lambda_l^{gen} = 5$, $\lambda_{per}^{gen} = 1$. The overall loss function when training the whole model in

the last stage is:

$$L = 0.25 * L^{warp} + L^{gen}. \tag{4.12}$$

4.5 Virtual Try-on-guided Pose for Data Synthesis

By using the K-Means clustering algorithm, we observe that the original VITON dataset [1] is mainly composed of images with straight-arm poses (as in Figure 4.6(a)). This bias creates a challenge as models trained on such data are prone to overfit and perform poorly on images with different upper-body poses. To tackle this problem, we propose the Virtual Try-on-guided Pose for Data Synthesis (VTP-DS) pipeline. Intending to improve the existing virtual try-on framework, the pipeline incorporates two key ideas: automatically detecting poorly performed poses using the Object Keypoint Similarity (OKS) metric [62] and synthesizing new training data specifically targeting those poses. The overview of the VTP-DS pipeline is illustrated in Figure 4.5.

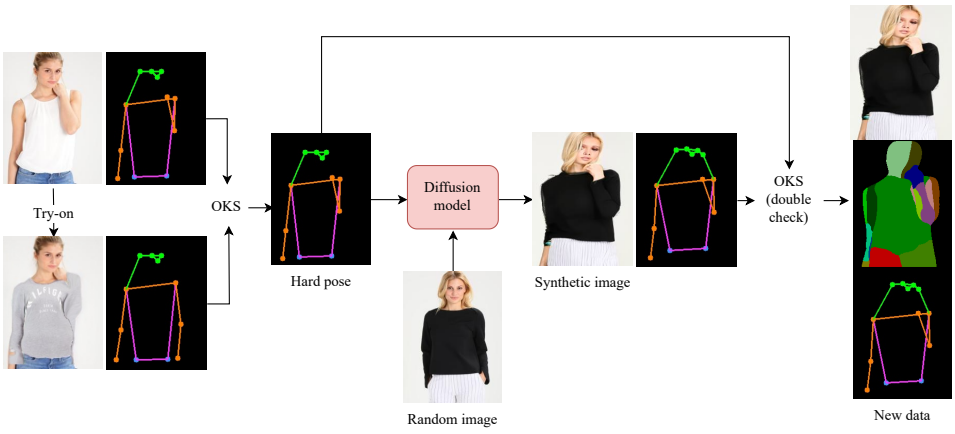
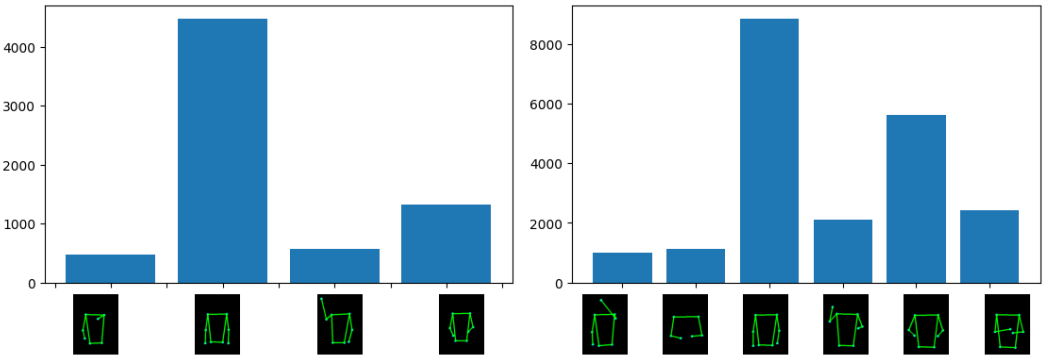


Figure 4.5: Overview of Virtual Try-on-guided Pose for Data Synthesis pipeline.

Given an input image containing a person, we extract that person's pose by using the YOLOv7 pose estimation method [63]. Then, we utilize our trained DM-VTON model to perform virtual try-on on the input image. The extracted



(a) VITON (b) VITON + synthesized data

Figure 4.6: Pose distribution in VITON dataset [1].

pose from the resulting image is compared with the pose of the input image using a customized OKS metric Equation 4.13.

$$\frac{1}{|P|}\sum_{i\in P}\exp(\frac{-d_i^2}{2s^2k_i^2}), \tag{4.13}$$

where P denotes the set of arm and hand keypoints, while the original formula uses all keypoints; d_i denotes the Euclidean distance between the keypoint i of two poses; s denotes the total area containing the pose; k_i is the constant provided by Lin et al. [62] to represent the standard deviation for keypoint i. Because we focus on distinguishing different upper-body poses, only the arm and hand keypoints contribute to the formula. If the OKS score falls below a specified threshold $t = 0.9$, it is identified as a hard pose.

Once identifying a hard pose, we randomly pick a person image from the VITON dataset. Then it leverages Bhunia's Diffusion model [64] to synthesize a new image of the person in the corresponding pose. To ensure the accuracy of the synthesized image, we perform a double-check using the OKS metric to verify the correctness of the output pose. Finally, DensePose [37] is utilized to generate the body-parser map of the synthesized image.

We initially collected videos from Youtube to synthesize additional data for training networks, specifically focusing on posing or catwalk videos. These videos had varying durations, ranging from 1 to 10 minutes. Subsequently, we extracted individual frames from these videos, which served as the input for our VTP-DS pipeline. After that, we manually removed low-quality results, resulting in 14,314 high-quality synthesized images for training the networks.

To access the quality of synthesized images, we use the K-means algorithm combined with our modified OKS metric. As details of the pose clustering results shown in Figure 4.6, data in the VITON training set mainly focuses on poses with simple poses (i.e. arms are less covered, low rotation amplitude). Meanwhile, when combined with our synthesized images, new pose clusters appear, and the concentration of data in groups is less imbalanced, which can help to train robust virtual try-on models.

4.6 Experiments

We conducted experiments comparing our DM-VTON framework with other state-of-the-art (SOTA) methods in terms of inference speed, memory usage, and the realisticness of the output. During the experimentation, we carefully evaluated the trade-off between those factors. As depicted in Figure 4.7, our DM-VTON framework outperforms all existing state-of-the-art methods regarding inference speed and memory usage while maintaining an equal quality of results.

4.6.1 Detailed Implementation

The Teacher and Student network training process follows the same strategy with two stages: the first stage only trains the warping module, while the latter trains the entire network. Both were under the same setting and carried out on a single Nvidia A100 GPU. We trained the model for 100 epochs with the initial

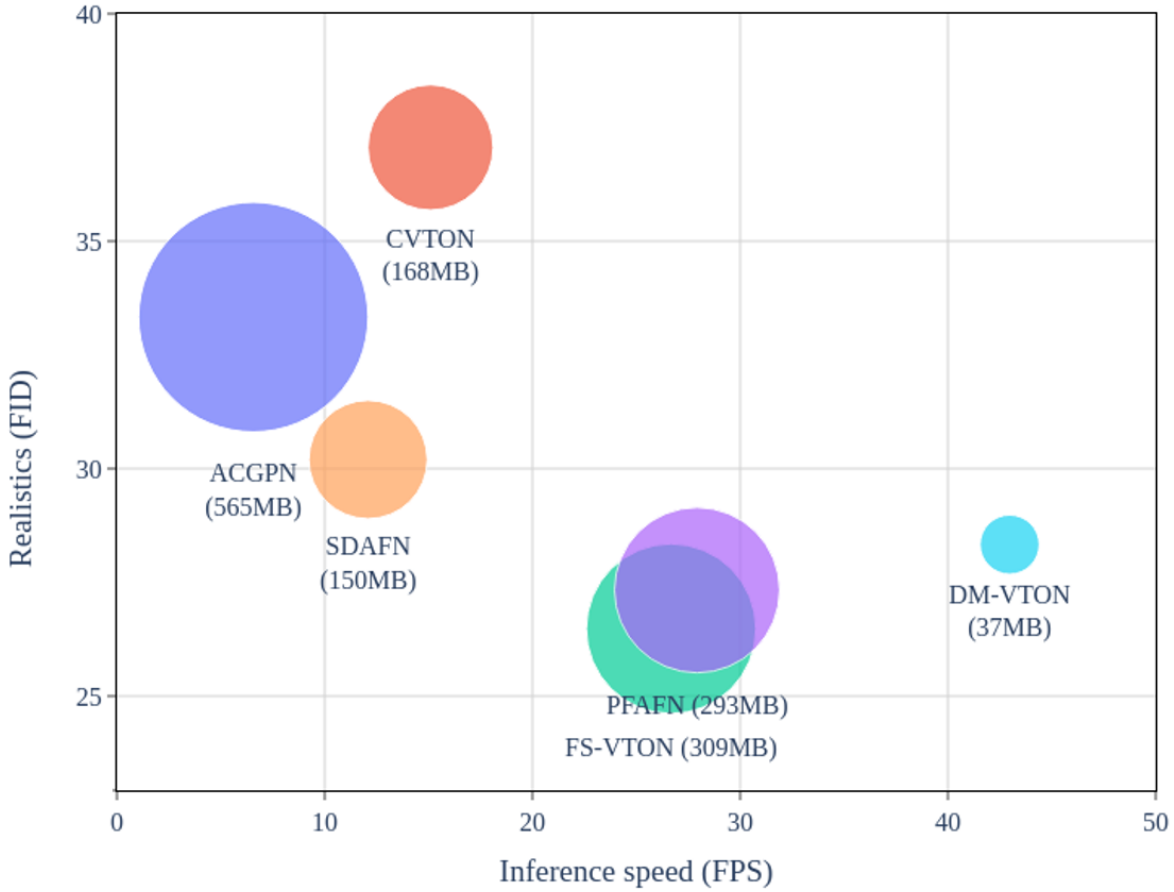


Figure 4.7: The comparison of our method (DM-VTON) and SOTA methods on VITON test set [1] in terms of realistic results (FID [28], lower is better), inference speed (FPS, higher is better), and memory usage. The size of each bubble represents the memory footprint. FPS is measured using a single Nvidia Tesla T4 GPU.

learning rate is 5×10^{-5}, which decays linearly after the first 50 epochs.

4.6.2 Experimental Settings

VITON [1], the most popular dataset for evaluating virtual try-on, was used to evaluate methods. It contains 16,253 frontal-view upper-body woman and top clothing image pairs with 256×192 resolution. However, we followed the work of Han et al. [3] to filter out duplicates and ensure no data leakage happens, remain-

ing 6,824 training image pairs and 416 testing image pairs in the cleaned VITON dataset, denoted by VTION-Clean. We combine the VTION-Clean training set and our synthesized images to train our proposed method.

Fréchet Inception Distance (FID) [28] and Learned Perceptual Image Patch Similarities (LPIPS) [65] metrics were used to evaluate the similarity of try-on results to real images.

4.6.3 Experimental Results

We compared the performance of our proposed MD-VTON with SOTA methods in virtual try-on, such as ACGPN [4], PF-AFN [6], C-VTON [15], SDAFN [16], FS-VTON [7]. Comparison of MD-VTON against those methods in terms of image quality (i.e., FID and LPIPS), inference speed (i.e., ms), FLOPs (Floating point operations), and memory usage (MB) is shown in Table 4.1. Our proposed method outperforms all other SOTAs in terms of runtime, FLOPs, and memory consumption. On the other hand, our DM-VTON achieves slightly higher FID and LPIPS scores than those of PF-AFN [6] and FS-VTON [7]. The experimental results prove that the proposed DM-VTON can run in real-time (i.e., 43 frames per second) with small memory consumption but still retains high-quality virtual try-on results. The visualization of compared methods is illustrated in Figure 4.8.

Table 4.1: Quantitative results between DM-VTON and SOTA virtual try-on methods. The † marker indicates the results measured by the generated images provided by the authors. The speed was evaluated on a single Nvidia T4 GPU.

Method	**Published**	Parser	Pose	**FID ↓**	**LPIPS ↓**	**Runtime (ms) ↓**	**FLOPs (B) ↓**	**Memory usage (MB) ↓**
ACGPN [4]	CVPR 2020	✓	✓	33.33	0.231	153.64	399.08	565.86
PF-AFN [6]	CVPR 2021			27.33	0.216	35.80	137.85	293.25
C-VTON† [15]	CVPRW 2022	✓		37.06	0.241	66.90	108.47	168.60
SDAFN [16]	ECCV 2022		✓	30.20	0.245	83.42	149.40	150.87
FS-VTON [7]	CVPR 2022			**26.48**	**0.200**	37.49	132.98	309.25
DM-VTON	Ours			28.24	0.215	**23.27**	**69.82**	**37.79**

Figure 4.8: Qualitative comparison on VITON-Clean dataset [1].

CHAPTER 5
FASHION RECOMMENDATION

In this chapter, we address the problem of recommending items given a reference fashion item. We carefully investigate three approaches to retrieving items: similar items within the same category, complementary items from other categories, and items guided by text feedback. In terms of retrieving intra-category similar items and text feedback-guided items, we employ a pretrained CLIP-based model and receive remarkable results. As for the inter-category complementary item retrieval, we consider it a Natural Language Process problem and propose Outfit Retrieval Transformers (ORT), which utilizes the Transformers architecture. Through experiments, ORT proves its effectiveness and can produce reasonable recommendations. Because using an embedding to query items from a dataset plays an important role in recommendation, we analyze various approximate searching methods and compare them with the exhaustive K-Nearest Neighbor algorithm regarding query time and accuracy.

5.1 Overview

Fashion recommendation involves retrieving items from a fashion dataset based on some given information and proposing them to users. To limit the scope, in this thesis, we take an input garment as the reference item and investigate three approaches to retrieving items based on the visual feature of that item.

The first approach is **intra-category similar item retrieval**, where we search for items within the same category and with similar visual representations to the reference item.

The second approach, called **inter-category complementary item retrieval**,

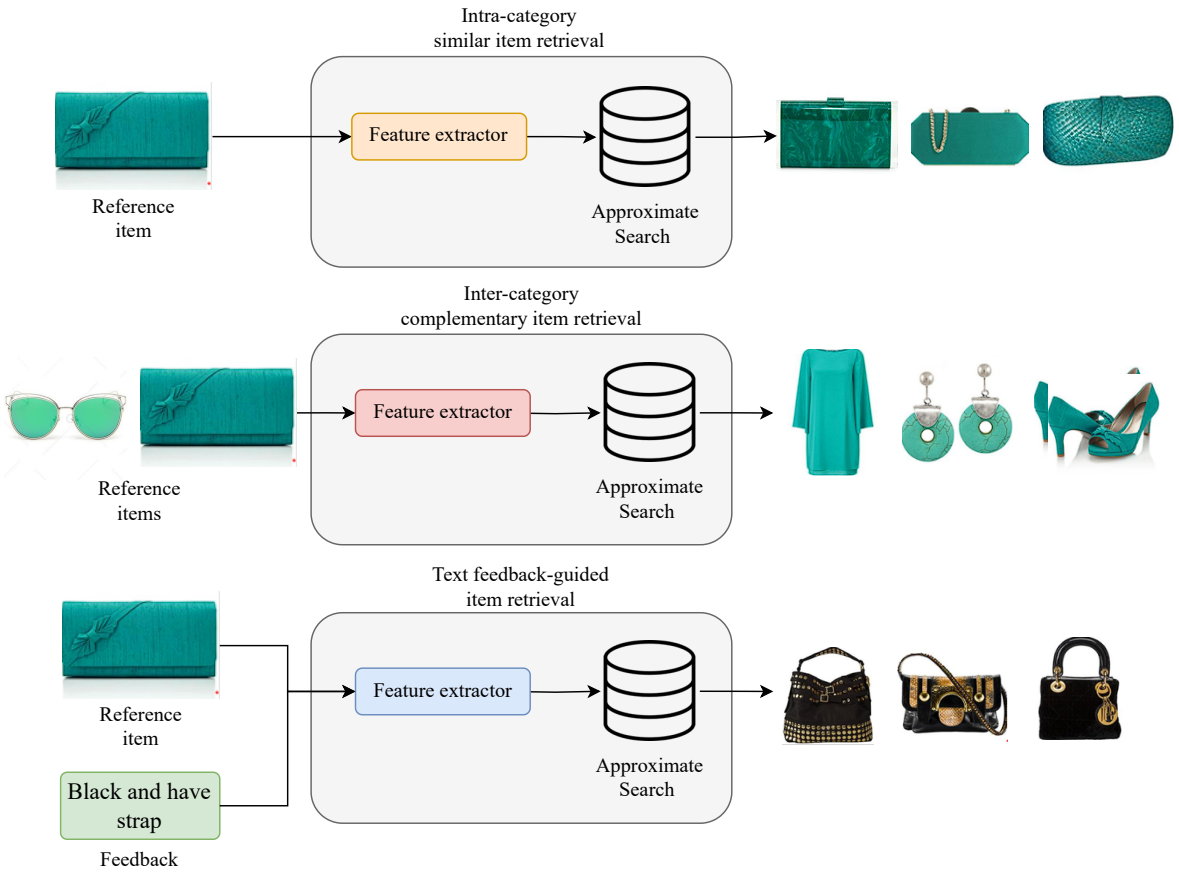


Figure 5.1: Our investigation scope on fashion recommendation

broadens the scope by considering items from different categories that complement the reference item to make a complete outfit. We define a complete outfit as an outfit that contains at most one item of each category, and all items share a similar style or are visually compatible. This approach differs from the fill-in-the-blank task described in PolyvoreOutfits [22], where each outfit has only one missing item. Our problem is more generalized, allowing for varying numbers of missing items in an outfit. The last approach we investigate in this thesis is **text feedback-guided item retrieval**. In this approach, we aim to find fashion items that satisfy the constraints expressed in the user's feedback, provided in natural language. The feedback may describe relative attributes such as being more formal or having longer sleeves.

In all mentioned approaches, after extracting the visual features of the desired items, we utilize the PolyvoreOutfits dataset [22] and search through the embedded features of all items to find the most similar ones. This dataset contains metadata for 251,008 fashion items, and 35,140 outfits, and covers 11 categories: bags, tops, outerwear, hats, bottoms, scarves, jewelry, accessories, shoes, and sunglasses. Due to the dataset's scale, we explore the use of approximate search methods [49, 50, 51] instead of the commonly used exhaustive K-Nearest Neighbor method for more efficient searching. Figure 5.1 illustrates our investigation scope on fashion recommendation.

5.2 Dataset Preprocessing

As we use the visual features of items in the dataset in every search, we extract the visual embedding of all items beforehand and use them directly in later searches. In this thesis, we utilize CLIP [45], which learns joint representations of images and their corresponding textual descriptions (Figure 5.2), to encode all items. Both the Image and Text Encoder of CLIP are built upon the Transformer Encoder [12] block. By passing an image through the CLIP model, we obtain a visual embedding that captures its visual features and semantic information.

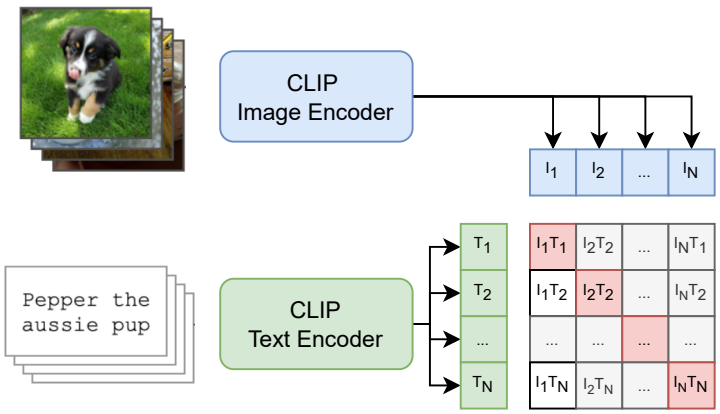


Figure 5.2: CLIP learning process

5.3 Intra-category Similar Item Retrieval

In terms of intra-category similar item retrieval, we first extract the visual embeddings of the reference image using the same CLIP model that we used to extract the embeddings of all items in our dataset. Subsequently, the approximate searching algorithm uses the extracted features to retrieve similar items from the PolyvoreOutfits dataset. Figure 5.3 illustrates our similar item retrieval pipeline.

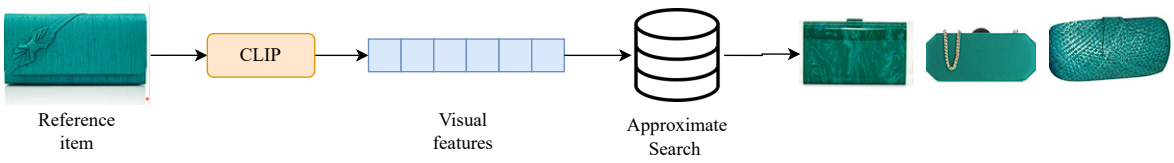


Figure 5.3: Similar item retrieval pipeline

5.4 Inter-category Complementary Item Retrieval

We address the inter-category complementary item retrieval task as the sentence generation task in the natural language domain. Transformers [12], a deep learning architecture for natural language processing (NLP) tasks, has proven its efficiency in generating a complete sentence from some given keywords [66] or a partial sentence [33].

In this thesis, we investigate using the Transformers architecture to retrieve complementary items and make a complete outfit. We name this method **O**utfit **R**etrieval **T**ransformers (ORT). Specifically, we consider a complete outfit as a sequence of fashion items, where the order is determined by their categories: bags, tops, outerwear, hats, bottoms, scarves, jewelry, accessories, shoes, and sunglasses. Not all categories are required. All items in an outfit are divided into three groups: *Input*, *Output*, and *Unavailable*. The division of items among these

groups depends on whether it's the training or inference phase.

PolyvoreOutfits [22] dataset provides us with the metadata of over 35,000 complete outfits, including which items belong to each outfit. Thus, in the training phase, we divide items into an outfit as follows:

- *Input* items are selected randomly from the available items.
- The remaining items within available items are *Outfit* items.
- Items in categories that the outfit lacks are denoted as *Unavailable*.

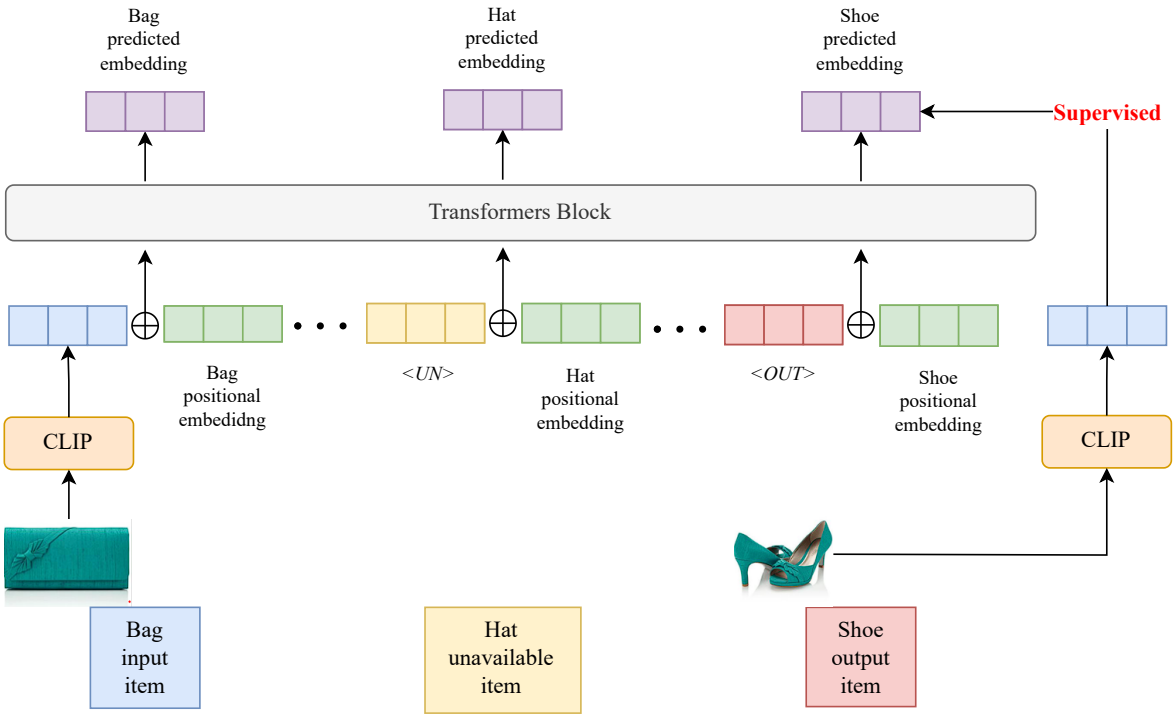
During the inference phase, as we generate complementary items for the reference item, we use the following division:

- *Input* item is the reference item only.
- *Output* items belong to categories that we aim to propose to users. These items are our target complementary items.
- Items in the remaining categories are denoted as *Unavailable*.

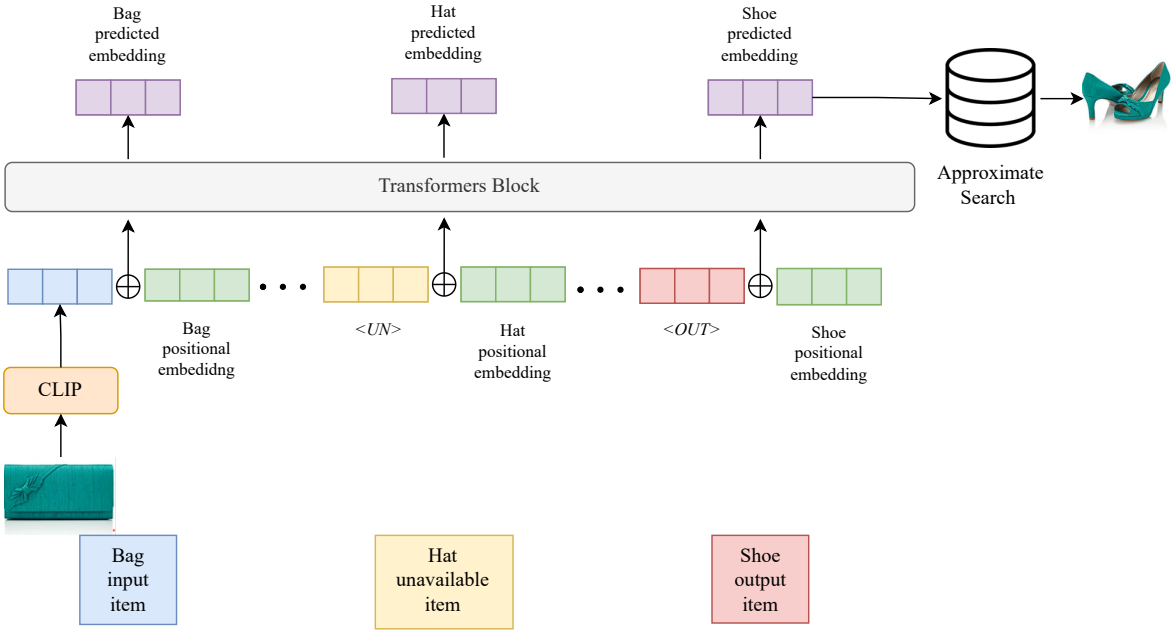
Each group of items has different ways to represent their embedding:

- *Input*: the item's visual embedding as-is.
- *Output*: a shared, learnable embedding, denoted as $<OUT>$.
- *Unavailable*: a shared, learnable embedding, denoted as $<UN>$.

All embeddings are fed into a Transformers Encoder block [12] to produce cross-semantic-aware output embeddings (as shown in Figure 5.4). ORT also use positional embeddings technique [12], where each position specifies the item's category. During the training phase, the network minimizes the noise contrastive



(a) Train phase



(b) Inference phase

Figure 5.4: Outfit Retrieval Transformers pipeline.

loss [67] so the Transformers block learns to maximize the similarity between its output embeddings and the *Output* items' original visual embeddings (as shown

in Figure 5.4(a)). We formulate the loss L as follows:

$$L = -\frac{1}{N} * \Sigma_{i=1}^{N} log \frac{exp(S_{i_P})}{exp(S_{i_P}) + exp(S_{i_N})}, \tag{5.1}$$

$$S_{i_P} = cos(pred_i, pos_i), \tag{5.2}$$

$$S_{i_N} = \Sigma_{j \in neg_i} cos(pred_i, j), \tag{5.3}$$

where N is the number of *Output* items in all outfits, $pred_i$ is the i^{th} predicted output embeddings, pos_i is the ground-truth visual embeddings of that *Output* item, neg_i is a set of negative samples which contains the embeddings of other items of the same category.

During the inference phase, we utilize the approximate searching algorithm to retrieve similar items from ORT's output embeddings (as shown in Figure 5.4(b)).

5.5 Text Feedback-guided Item Retrieval

To address this task, we adopt the CLIP4Cir architecture proposed by Baldrati et al [47]. The architecture proves efficiency despite its simplicity. It first utilizes CLIP model [45] to extract embeddings from both the reference image and relative feedback. Then, both embeddings are fed into a Combiner network, a small trainable network specifically designed to merge the two embeddings into a single output embedding (as shown in Figure 5.5).

During the training phase, we utilize the FashionIQ dataset [24], which consists of triplet pairs comprising a reference image, relative feedback, and a target image. The Combiner is trained to maximize the similarity between its output embeddings and the target images' embeddings produced by CLIP (Figure 5.6(a)),

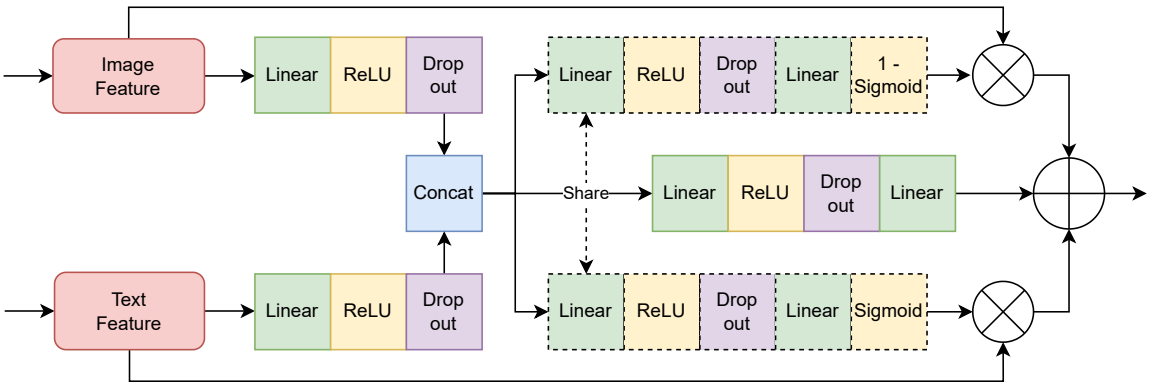
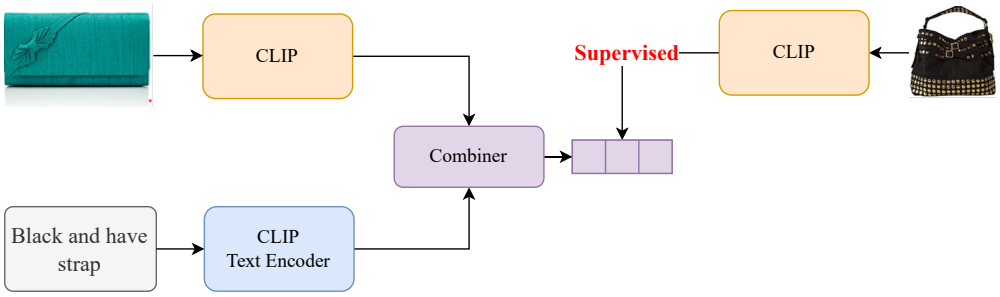
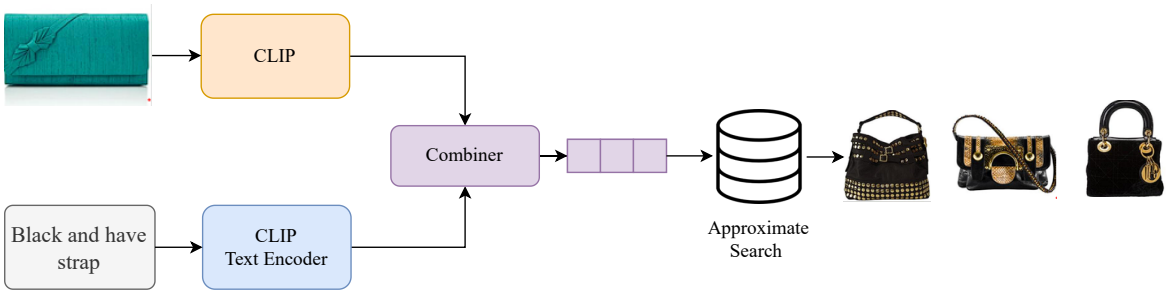


Figure 5.5: Combiner architecture



(a) Train phase



(b) Inference phase

Figure 5.6: CLIP4Cir pipeline.

specified by the following symmetric contrastive loss L:

$$L = \frac{1}{2}(L_c + L_i), \tag{5.4}$$

$$L_c = \text{cross\_entropy\_loss}(E_c E_i, logits), \tag{5.5}$$

$$L_i = \text{cross\_entropy\_loss}(E_i E_c, logits), \tag{5.6}$$

where *logits* denotes the list $\{1, 2, ..., N\}$ with N is the batch size, E_c denotes a batch of Combiner's output embeddings and E_i denotes a batch of CLIP's output embeddings. This shared representation space allows us to effectively compare the similarity between the embeddings produced by the Combiner and CLIP.

To run inference, we also leverage the approximate searching algorithm to retrieve similar items for the target output embeddings (as shown in Figure 5.6(b)).

5.6 Approximate Searching

In all of the three recommendation approaches we have investigated so far, the final involves retrieving the items from the dataset that have visual embeddings most similar to a given input embedding.

For such nearest neighbor problem, K-Nearest Neighbor (KNN) algorithm are widely used due to its simplicity and giving exact results([21, 47, 46, 23]. The KNN algorithm exhaustively searches through every dimension of all items in the dataset. As a result, this algorithm has the runtime complexity of $O(kNd)$ where k is the number of items to retrieve, N is the size of the dataset, and d is the number of dimensions to represent one item, in our case, it is the length of the visual embedding produced by CLIP [45].

Approximate searching algorithms solve this problem by lowering either N or d, thus allowing the search faster with the accuracy trade-off. Approximate methods also help compress the dataset due to the decline of d [68, 48]. However, as mentioned in Jegou's work [69], these methods significantly decrease the result accuracy. In this thesis, our primary goal is to find the fastest approximate methods while keeping a reasonable accuracy. Thus, we focus on investigating three common algorithms that lower the search space N: Inverted File Index [49] (IVF), Hierarchical Navigable Small Worlds [51] (HNSW), and Approximate

Nearest Neighbors Oh Yeah [50] (ANNOY). We use the Euclidean metric as our distance metric while retrieving nearest neighbors.

5.6.1 Inverted File Index (IVF)

The basic idea behind the Inverted File Index [49] is to partition the dataset into multiple subsets, usually done by employing K-Means. We denote the number of subsets as *nlist*. Subsequently, all embeddings in the dataset are stored separately in their corresponding subset. Each subset also contains a centroid, which is the mean of all embeddings in that subset.

We calculate the distance between the input embedding and all centroids during the query process and take the closest one. Then, we only search through all items in the corresponding subset. This means the larger *nlist*, the smaller subset we need to search through.

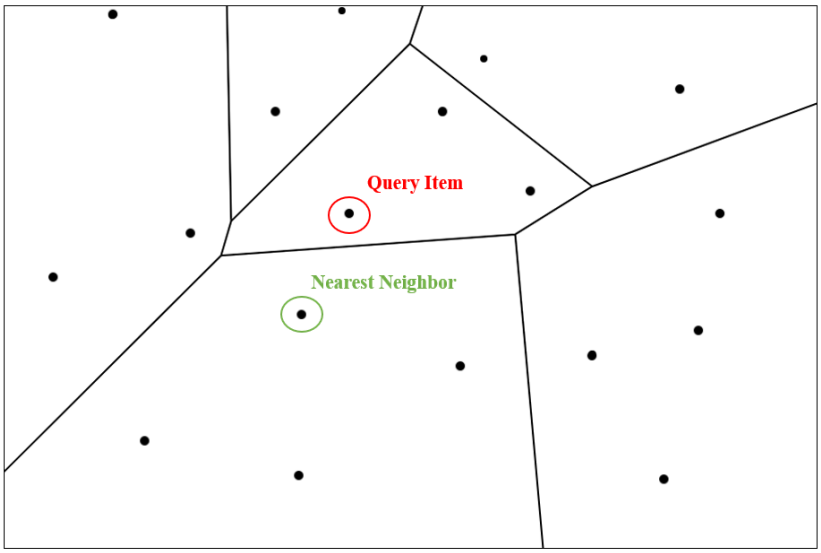


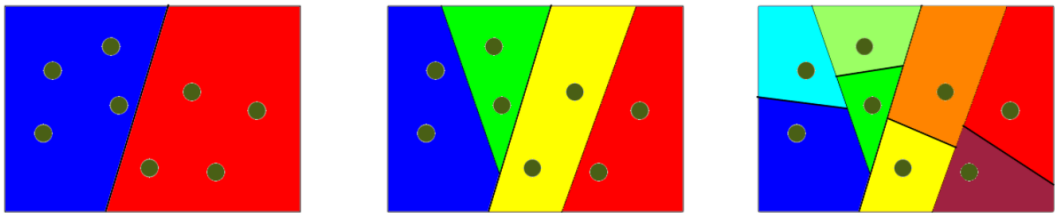
Figure 5.7: The case where the query embedding falls on the outskirts of the closest subset in IVF

However, if the query embedding falls on the outskirts of the closest subset,

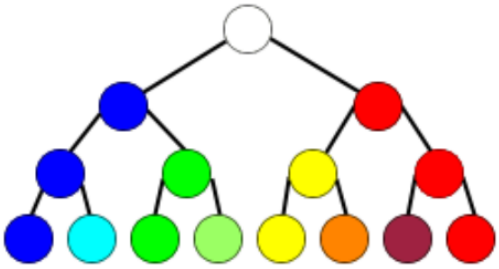
its nearest neighbors may belong to nearby clusters (Figure 5.7). This is what makes this method approximate and not exact. The impact can be reduced by searching for multiple nearby subsets; we denote the number of subsets to search as *nprobe*. A larger *nprobe* means taking more time but giving better accuracy. The number of subsets *nlist* and the number of subsets to search *nprobe* can be tuned to find the time/accuracy reasonable trade-off.

5.6.2 Approximate Nearest Neighbors Oh Yeah (ANNOY)

ANNOY [50] is the search algorithm used by Spotify in their music recommendation system [50]. The key idea of ANNOY is to build a binary tree, where each leaf node is an item in the dataset. ANNOY selects a random hyperplane to partition the dataset into two subsets at each intermediate node in the tree construction process. This hyperplane is determined by sampling two random points from the corresponding subset and taking the hyperplane equidistant from them. The construction process is shown in Figure 5.8. The constructed tree structure is stored alongside the dataset.



(a) Construction process



(b) Constructed tree

Figure 5.8: ANNOY index construction process

We traverse the binary tree from the root during the query process by repeatedly checking which side of the corresponding hyperplane the query embedding belongs to. This method also faces the same issue as IVF; if the query item falls on the outskirts of a subspace, the nearest neighbors may appear in sibling leaf nodes instead. To tackle this problem, ANNOY allows searching for multiple leaf nodes, which can be done by using a priority queue to keep track of the closest hyperplane so we can backtrack to that hyperplane and go to the other side. ANNOY also proposes building multiple trees at once to minimize the effect of randomness.

We denote the number of trees to build as n_trees and the total number of leaf nodes to search in all trees as $search\_k$. A larger n_trees gives more accurate results but larger memory usage; it does not affect the query time. A larger $search\_k$ leads to more accurate results but longer query time.

5.6.3 Hierarchical Navigable Small Worlds (HNSW)

The key idea of Hierarchical Navigable Small Worlds [51] (HNSW) is taken from the Skip List data structure. Skip List is a data structure that consists of multiple sorted linked lists. The bottom level (or level 0) is the ordinary linked list containing all the elements. The higher levels contain fewer elements and act as shortcuts to traverse the list quickly. Figure 5.9 illustrates the Skip List data structure.

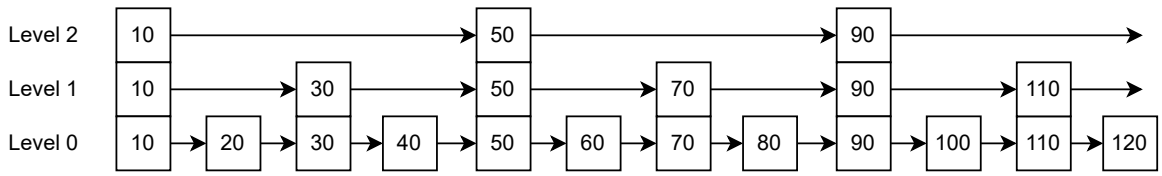


Figure 5.9: Skip list data structure

To search in a Skip List, we start at the highest level, which has the longest step between elements and iterate towards the end of that level. If the current element is greater than the search value, we move down to the previous node in the *below* level. This process repeats until we find the search value or reach the lowest level (level 0) (as shown in Figure 5.10).

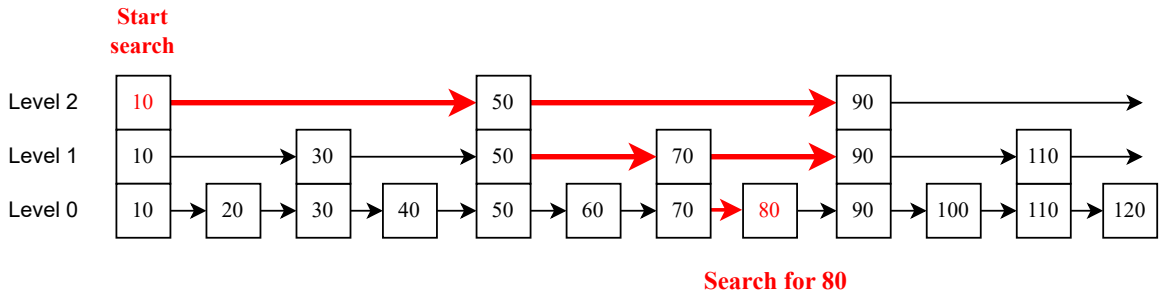


Figure 5.10: Skip list search process

HNSW borrows that idea and applies it in higher dimensions. Specifically, HNSW constructs multiple undirected graphs, where the graph at the lowest level (level 0) contains all dataset items as its nodes. The higher-level graphs contain fewer nodes, leading to longer distances between each node. To reduce the number of adjacent nodes during traversing, HNSW limits the maximum degree of each node in all graph levels, denoted as M. The HNSW data structure is constructed from a dataset by inserting each embedding vector one by one. Given M is the maximum degree of each node, $efConstruction$ and $efSearch$ are the maximum numbers of nodes to traverse at each level in the index and query process, respectively, the steps to insert a new embedding vector are as follows:

1. If this is the first item, insert it to all levels and continue.

2. Find the graph level L to insert the new item. The value of L has exponentially decreasing probability as it goes higher, specified by the equation $L = max(floor(-log(uniform(0,1))), log(M))$. This also means the max level

L_{max} is $log(M)$.

3. From level L_{max} down to $L + 1$, we repeatedly find the nearest embedding from the entry point (the first inserted item for level L_{max}) and use that embedding as the entry point for the next level.

4. From level L down to 0, we repeatedly find $efConstruction$ nearest embeddings from the entry point. Subsequently, we calculate the distance between the input embedding and those embeddings to find the nearest M and link the input embedding to them. We use the nearest embedding out of M as the entry point for the next level.

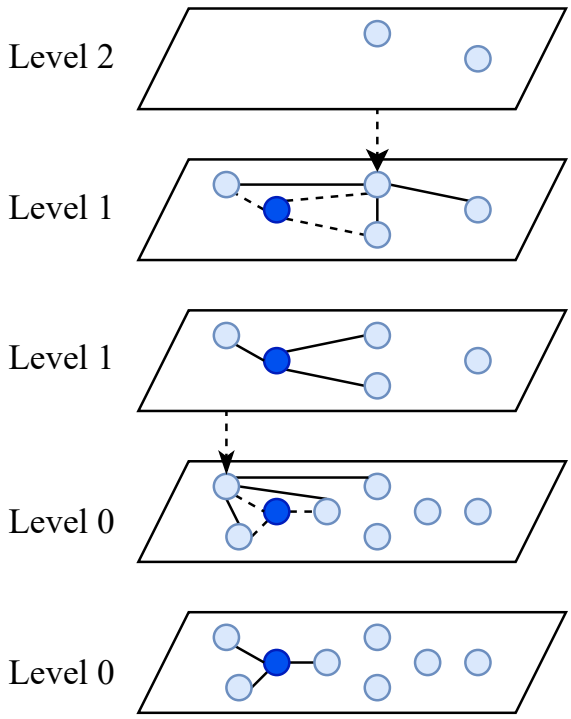


Figure 5.11: Insert a new vector into HNSW at level 1

Figure 5.11 illustrates the steps to insert a new item into HNSW, given $M = 3$ and $efConstruction = 4$.

Regarding the query process of HNSW, from level L_{max}, we repeatedly find

the nearest embeddings from the entry point and use that embedding as the entry point for the next level. Upon entering level 0, we find $efSearch$ nearest embeddings and compare the query embedding with each of them to get the nearest one. A higher $efSearch$ takes longer query time but gives more accurate results. The search process is greedy, and thus cannot ensure exact results.

5.7 Experiments

5.7.1 Detailed Implementation

For intra-category similar item retrieval, we employed the pretrained CLIP model released by Baldrati et al. [47] to extract the visual embeddings of the reference image and all items in the dataset.

For inter-category complementary item retrieval, the Outfit Retrieval Transformers (ORT) block has 6 layers, 8 attention heads, and 11 inputs in total (same as the number of categories of PolyvoreOutfits), and each input has 640 dimensions (same as CLIP output dimension). We trained the network from scratch on the PolyvoreOutfits dataset [22]. The AdamW optimizer was used with a learning rate 0.0001 and a weight decay of 0.3. The learning rate scheduler had a step size of 20 with a gamma factor 0.5.

For text feedback-guided item retrieval, we utilized the pretrained CLIP and Combiner models released by Baldrati et al. [47] to extract both the visual and textual embeddings from inputs and produce the merged output.

In terms of approximate search, we used the FAISS [70] implementation for IVF and HNSW algorithms, and the official implementation [50] of ANNOY algorithm.

5.7.2 Intra-category similar item retrieval evaluation

We performed a zero-shot evaluation on the DeepFashion dataset [71], without fine-tuning the CLIP model on it. This dataset contains 14,812 query images and 12,612 test images. Each image is assigned a label, and the dataset has a total of 8,082 unique labels.

For evaluation, we used the recall at rank K metric, with K in the list $\{1, 5, 10, 30, 50\}$. Specifically, we retrieved K images from the test set using the KNN algorithm for each query image. We considered the query correct if any of the K retrieved images had the same label as the query image. The recall score was calculated by dividing the number of correct queries by the total number of query images and multiplying the result by 100. The evaluation results are shown in Table 5.1. In a zero-shot evaluation, these results are acceptable. As we see in Figure 5.12, the false recommendations still look relevant to the reference images.

Table 5.1: Recall at rank K (R@K) of zero-shot CLIP on the DeepFashion dataset

	R@1	R@5	R@10	R@30	R@50
Zero-shot CLIP [47]	13.13	31.62	43.55	60.25	68.04

Figure 5.12: Some false recommendation in similar item retrieval evaluation. Top: reference image. Bottom: top 1 result

5.7.3 Inter-category complementary item retrieval evaluation

We performed the evaluation on the PolyvoreOutfits dataset [22] for this task. This dataset includes 251,008 fashion items, classified into 11 distinct categories. The distribution of items across these categories is shown in Figure 5.13.

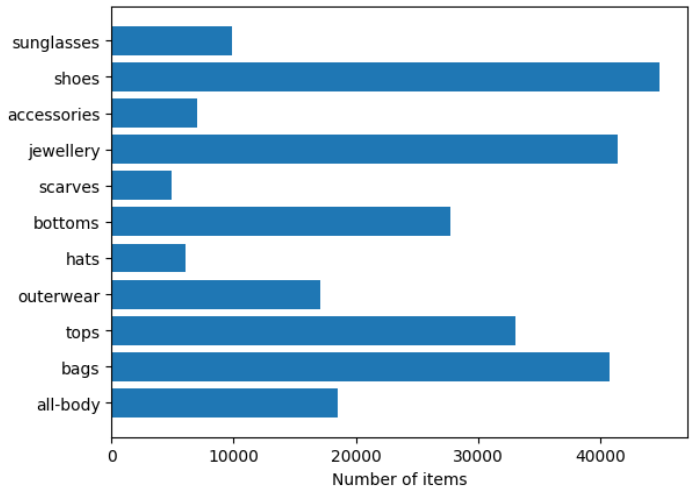


Figure 5.13: Number of items of each category in PolyvoreOutfits

Additionally, PolyvoreOutfits contains the metadata of 35,140 outfits. Each outfit in the dataset follows the constraint of including at most one item from each category. As a result, the number of items within each outfit can range from 2 to 11. Figure 5.14 shows the number of outfits each category appears in the train, validation, and test set.

To fine-tune the Outfit Retrieval Transformers, we utilized the outfits in the validation set. The metric for evaluation was the recall at rank K, which was calculated as follows:

1. For each *Output* item, retrieve K items within the same category as the original item using KNN. If the retrieved K items contain the original item, mark this *Output* correct.

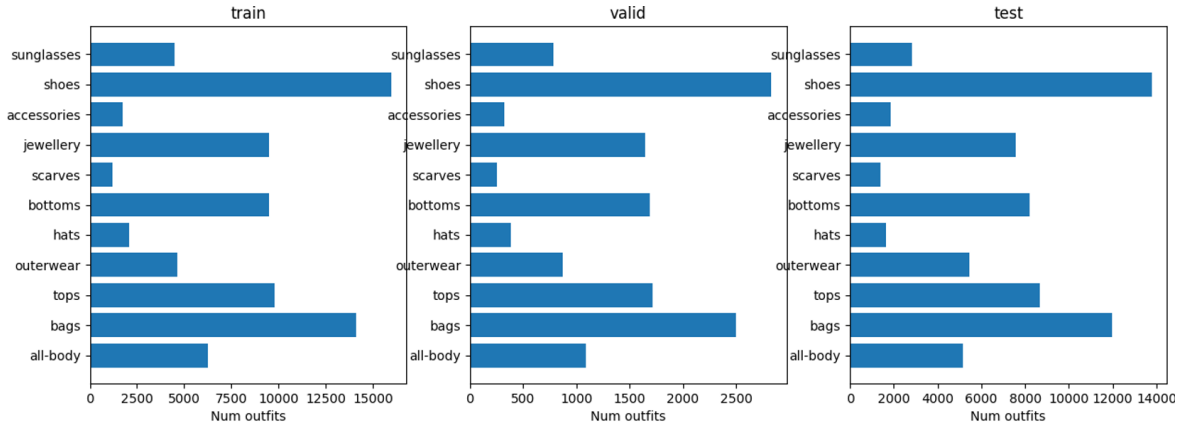


Figure 5.14: The number of outfits each category appears in PolyvoreOutfits

2. The recall score equals the number of correct *Output* items divided by the total number of *Output* items and multiplied by 100.

We investigated various settings and selected the configuration that achieved the highest recall at rank 10. The experiments were conducted with the following setups:

- Mean squared error loss and noise contrastive loss.
- Final loss involves output normalization and does not involve output normalization.
- Randomly select 90%, 50% and 10% of items in an outfit as *Input* items.

While fine-tuning the loss for Outfit Retrieval Transformer (ORT), the output normalization was included in the final loss and 90% of items in an outfit were chosen as *Input* items. Figure 5.15 shows the recall score of MSE loss in the train and validation set, whereas Figure 5.16 shows the result of noise contrastive loss. MSE loss was faster to converge, but the recall was relatively low, even at a higher recall rank. Meanwhile, the contrastive loss could still converge more and also achieve higher recall. In conclusion, we decided to use noise contrastive loss.

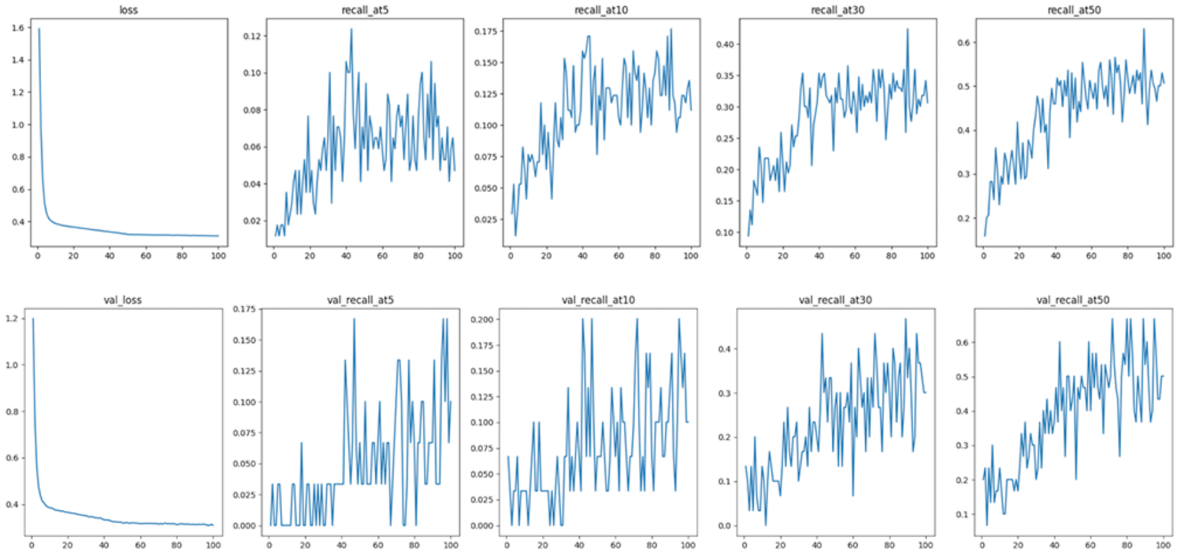


Figure 5.15: ORT w/ MSE loss. Top: train, bottom: validation

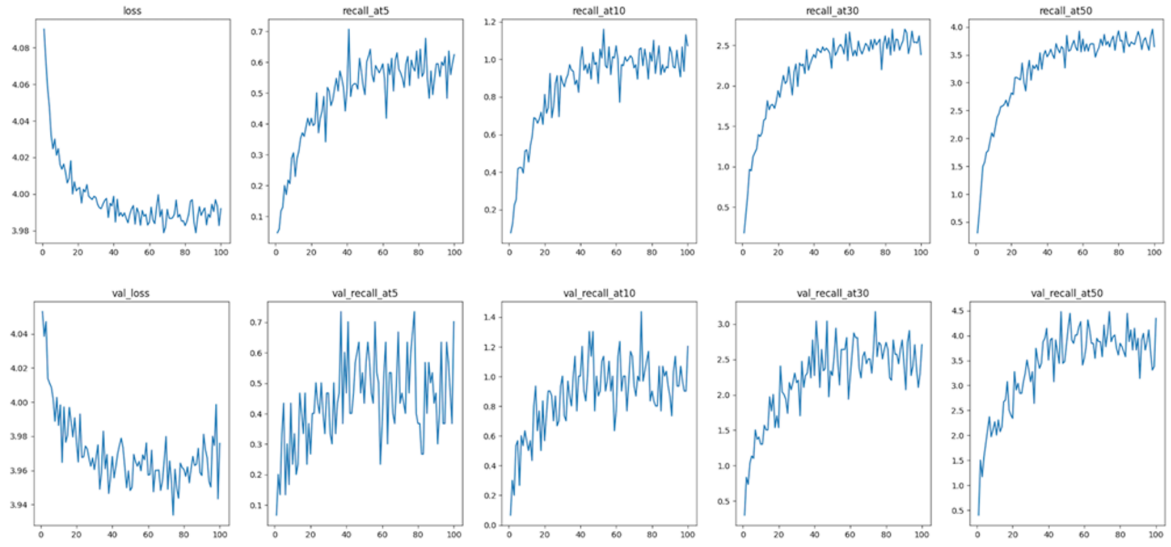


Figure 5.16: ORT w/ noise contrastive loss. Top: train, bottom: validation

We tuned the output normalization using the noise contrastive loss, as they achieved higher results in the above comparison. Moreover, we still kept the *Input* item's percentage at 90%. Figure 5.17 shows the results of not involving output normalizing in the final loss. Compared with Figure 5.16, we see that we can get higher recall and some signs of overfitting when removing the last output normalization during the training process. However, overall, it still achieves

higher recall in both the training and validation set.

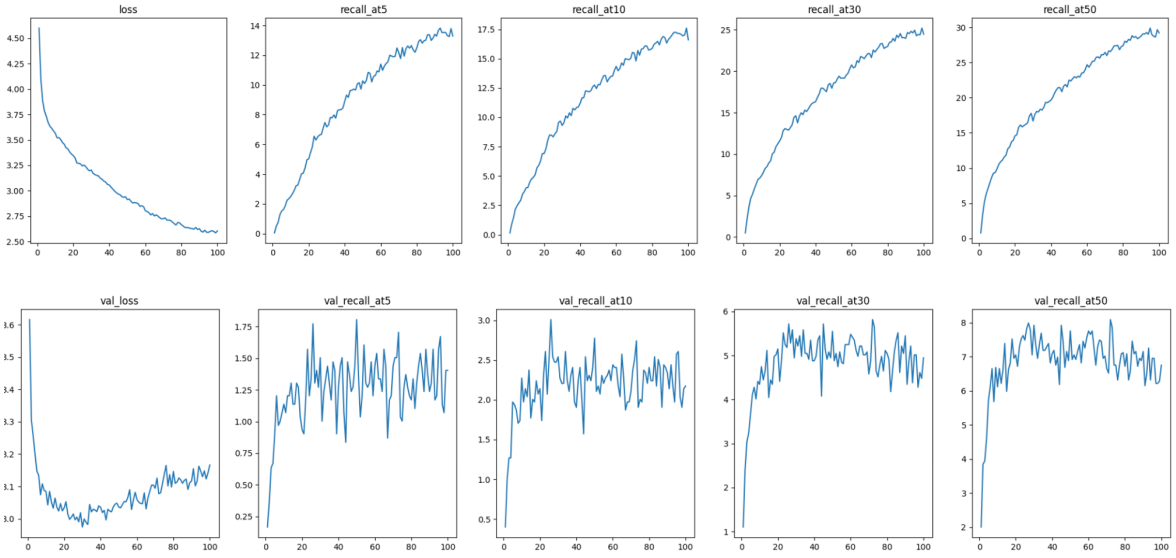


Figure 5.17: ORT w/o output normalization. Top: train, bottom: validation

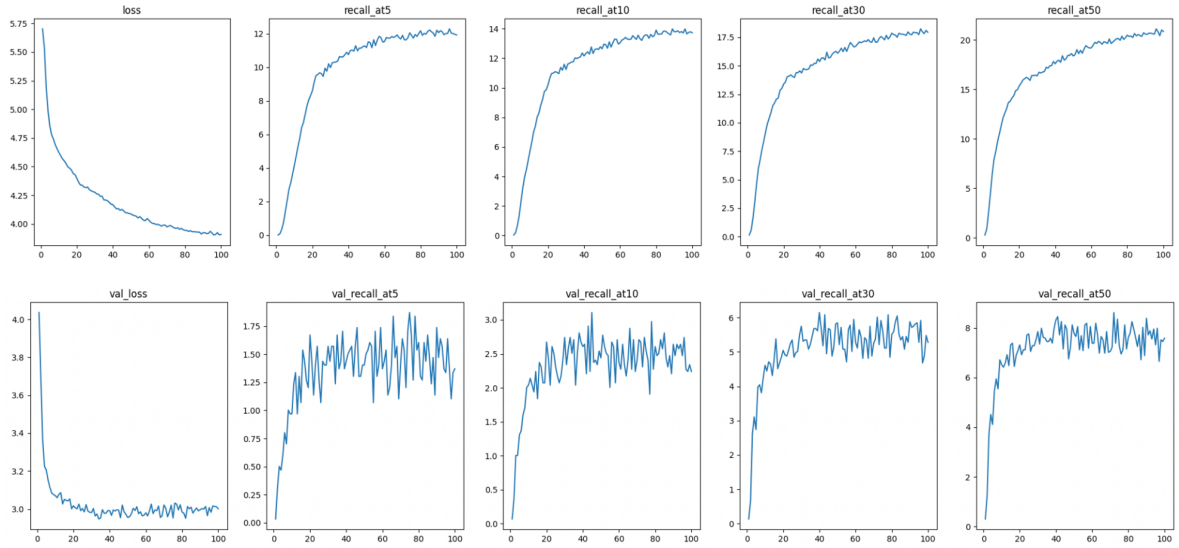


Figure 5.18: ORT w/ 50% of outfits are *Input* items. Top: train, bottom: validation

Using noise contrastive loss and not including output normalization in the final loss, we tuned the *Input* item's percentage in an outfit. Figure 5.18 shows the

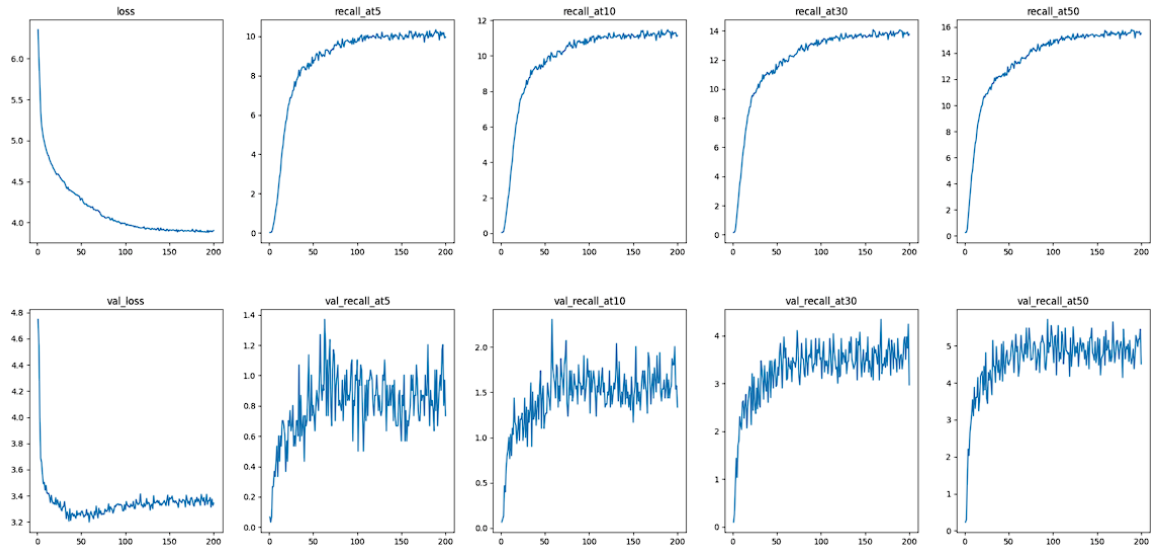


Figure 5.19: ORT w/ 10% of outfits are *Input* items. Top: train, bottom: validation

situation where 50% of all outfits are *Input* items in the training phase. Compared with Figure 5.17, the validation recall is still the same as when we only chose 10% of items, which means the model can learn even with fewer input items. Note that the percentage of *Input* items in the validation set was still kept at 90%. However, as we decreased the *Input* items percentage to 10%, the validation recall also dropped significantly (5 in Figure 5.19 in comparison with 8 in Figure 5.18). As a result, we decided to keep the percentage of *Input* items at 50%

In conclusion, we fine-tuned ORT with noise contrastive loss, no normalization before calculating final loss, and choosing 50% Additionally, We evaluated that model on the test dataset with different mask ratios and recall ranks, as shown in Figure 5.20. The recall dropped significantly when masking 70% of the items while slowly dropping when masking fewer items.

Comparison with SOTA methods

We also compared Outfit Retrieval Transformers (ORT) with other SOTA meth-

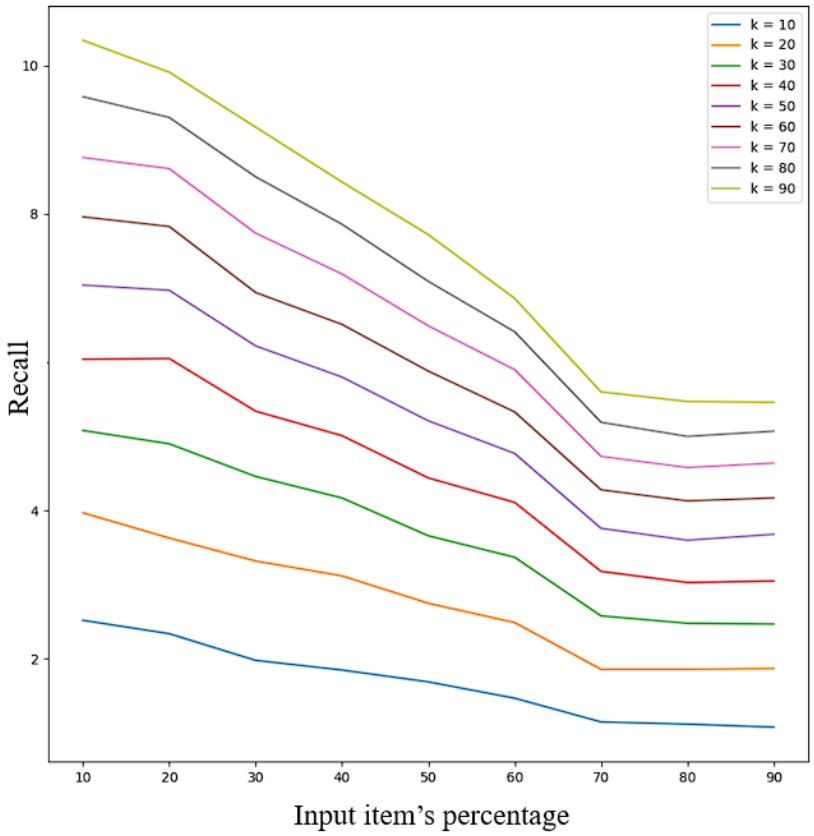


Figure 5.20: Recall score on test set with different *Input* item's percentage

ods: Type-Aware [22], CSA-Net [21], OutfitTransformer [23], and FashionViL [25], using their published results for the fill-in-the-blank task.

The comparison is performed on outfits of the PolyvoreOutfits test set, where we need to find 1 missing item for each outfit. The metric recall at rank K is slightly modified by Lin et al. [21] as follows:

1. For each missing item, retrieve K items within the same subcategory as that item using KNN (PolyvoreOutfits also contains 153 subcategories with no semantic meaning). If the retrieved K items contain the original item, mark the recommendation correct.

2. The recall score equals the number of correct recommendations divided by the total number of missing items and multiplied by 100.

The comparison result is shown in Table 5.2, showing that our model cannot achieve as high results as other methods in the fill-in-the-blank problem.

Table 5.2: Recall at rank K (R@K) score between our method and other SOTA

Method	Published	R@10	R@30	R@50
Type-Aware [22]	ECCV 2018	3.66	8.26	11.98
CSA-Net [21]	CVPR 2020	5.93	12.31	**17.85**
OutfitTransformer [23]	CVPRW 2022	**6.53**	12.12	16.64
FashionVIL [25]	ECCV 2022	5.83	**12.61**	17.49
ORT	**Ours**	4.19	8.55	11.96

Figure 5.21 shows some false recommendation samples. From our perspective, the recommendation results are still acceptable. Indeed, fashion taste varies among individuals, and a reasonable recommendation for one person may not be the same for another.

5.7.4 Text feedback-guided item retrieval evaluation

According to Baldrati's published work [47], their fine-tuned CLIP and Combiner models are evaluated on the FashionIQ validation set. Figure 5.22 shows the number of items of each category in the FashionIQ validation set.

The FashionIQ validation set also contains 12,034 triplet pairs. Each triplet pair comprises a reference image, relative feedback, and a target image. This format allows for the calculation of the recall at rank K metric, which can be computed as follows:

1. For each triplet, retrieve K items within the same category as the reference image using KNN. If the retrieved K items contain the target item, mark the recommendation correct.

2. The recall score equals the number of correct recommendations divided by

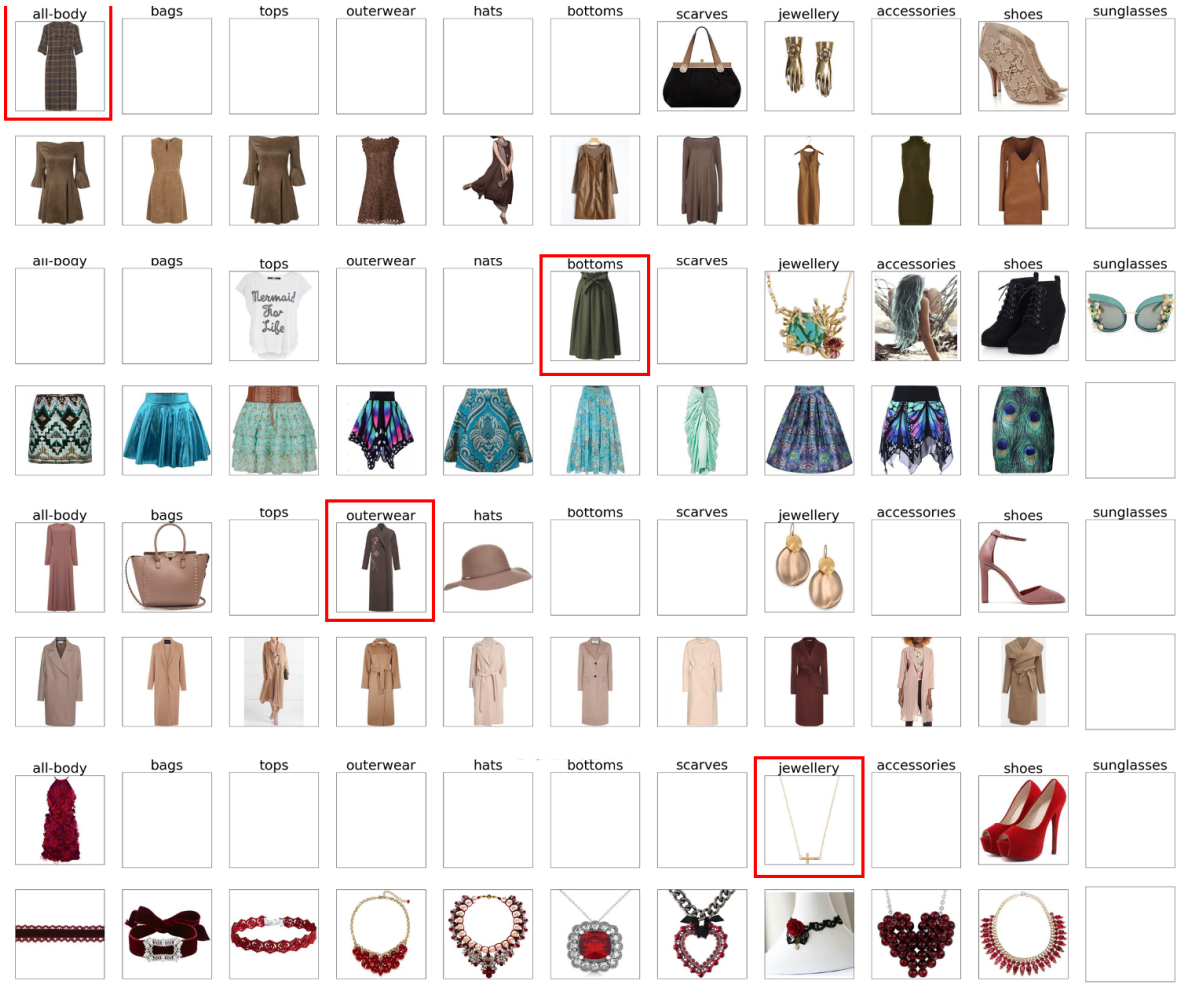


Figure 5.21: False recommendation according to the ground-truth missing items (marked as red). The following line shows the top 10 items recommended by Outfit Retrieval Transforms.

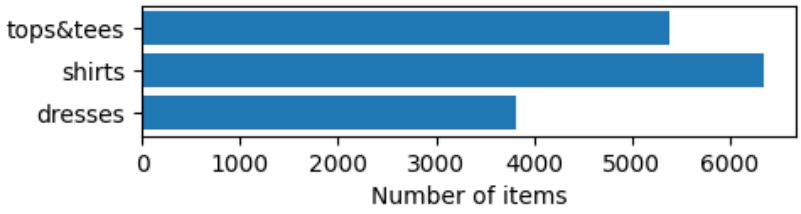


Figure 5.22: The number of items of each category in FashionIQ validation set

the total number of triplet pairs and multiplied by 100.

The evaluation results are shown in Table 5.3.

Table 5.3: Recall at rank K (R@K) on the FashionIQ validation set

Method	Shirt		Dress		Toptee		Average	
	R@10	R@50	R@10	R@50	R@10	R@50	R@10	R@50
CLIP4Cir[46]	36.36	58.00	31.63	56.67	38.18	62.42	35.39	59.03

5.7.5 Approximate searching evaluation

While evaluating an approximate searching method, besides the query time, we also utilized the recall at rank 100 metric with some customization:

1. Use K-Nearest Neighbor to query 100 nearest items in the dataset.
2. Use the approximate method to query 100 nearest items in the dataset.
3. The metric result is the number of items that belong to both query results.

As we are going to recommend items from the PolyvoreOutfits dataset [22] in our application, we used that dataset to compare the effectiveness of searching methods. We additionally appended it with 523,392 items from the DeepFashion dataset [71] to investigate the time complexity of approximate searching methods. As a result, there were over 774,400 items in total to index. Because all approximate searching methods contain multiple parameters to fine-tune, we conducted multiple experiments on the merged dataset to find the appropriate trade-off parameters for each method:

- IVF: we chose *nlist* from the list $\{32, 64, 128, 256, 512, 1024, 4096, 16384\}$. With each *nlist* value, we increased the *nprobe* value until it achieved the average recall of 98.

- ANNOY: we chose n_tree from the list $\{32, 64, 128, 256, 512\}$. With each $n_t ree$ value, we increased the $search\_k$ value until it achieved the average recall of 98.

- HNSW: we chose M from the list $\{4, 16, 32, 64, 128\}$, $efConstruction$ from the list $\{8, 16, 32, 64\}$ and $efSearch$ from the list $\{32, 64, 128, 256\}$. Subsequently, we removed all setups not achieving the average recall of 98.

We picked the setup having the fastest average query time for each method and compared those methods with KNN on both the merged dataset and Polyvore-Outfits dataset in terms of query time, recall score and memory usage.

Table 5.4 and Table 5.5 show various settings for Inverted File Index (IVF) and Approximate Nearest Neighbor Oh Yeah (ANNOY) that achieve the minimum average recall score of 98.

As for Hierarchical Navigable Small Worlds (HNSW), Figure 5.23 illustrates the recall score of all settings we prepared. The experiment results were just as we expected; higher values of M, $efConstruction$, and $efSearch$ led to higher recalls. Table 5.6 shows the results after inserting the query time and removing all settings, not achieving 0.98 recall.

Table 5.4: Some settings for IVF achieving average recall score of 98

nlist	*nprobe*	Query time (ms)	Recall
32	8	73	99.55
64	8	37.44	98.26
128	16	38.65	99.39
256	16	18.72	98.23
512	24	14.78	98.28
1024	40	12.77	98.41
4096	**88**	**8.05**	**98.11**
16384	176	10.14	98.08

Table 5.5: Some settings for ANNOY achieving average recall score of 98

n_tree	*search_k*	Query time (ms)	Recall
32	53248	15.16	98.10
64	45056	12.92	98.01
128	**49152**	**12.66**	**98.07**
256	49512	17.39	98.10
512	49512	13.79	98.01

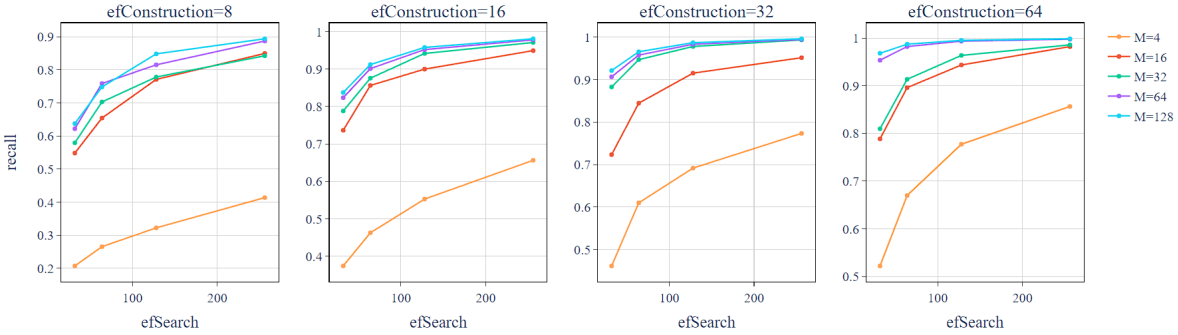


Figure 5.23: Recall score for multiple HNSW setups

Table 5.6: Some settings for HNSW achieving average recall score of 98

M	*efConstruction*	*efSearch*	Query time (ms)	Recall
64	64	64	2.42	98.21
128	64	64	5.37	98.71
32	**32**	**128**	**1.72**	**98.01**
64	32	128	3.05	98.33
128	32	128	3.85	98.71
64	64	128	4.85	99.35
128	64	128	3.92	99.52
128	16	256	3.91	98.09
32	32	256	3.66	99.36
64	32	256	5.26	99.48
128	32	256	4.19	99.63
16	64	256	2.75	98.20
32	64	256	3.92	98.57
64	64	256	6.99	99.76
128	64	256	7.27	99.85

From the comparison in Table 5.4, Table 5.5, and Table 5.6, we decided to use $nlist = 4096$ and $nprobe = 88$ for IVF; $n\_tree = 128$ and $search\_k = 49152$ for ANNOY; and $M = 32$, $efConstruction = 32$, and $efSearch = 128$ for HNSW.

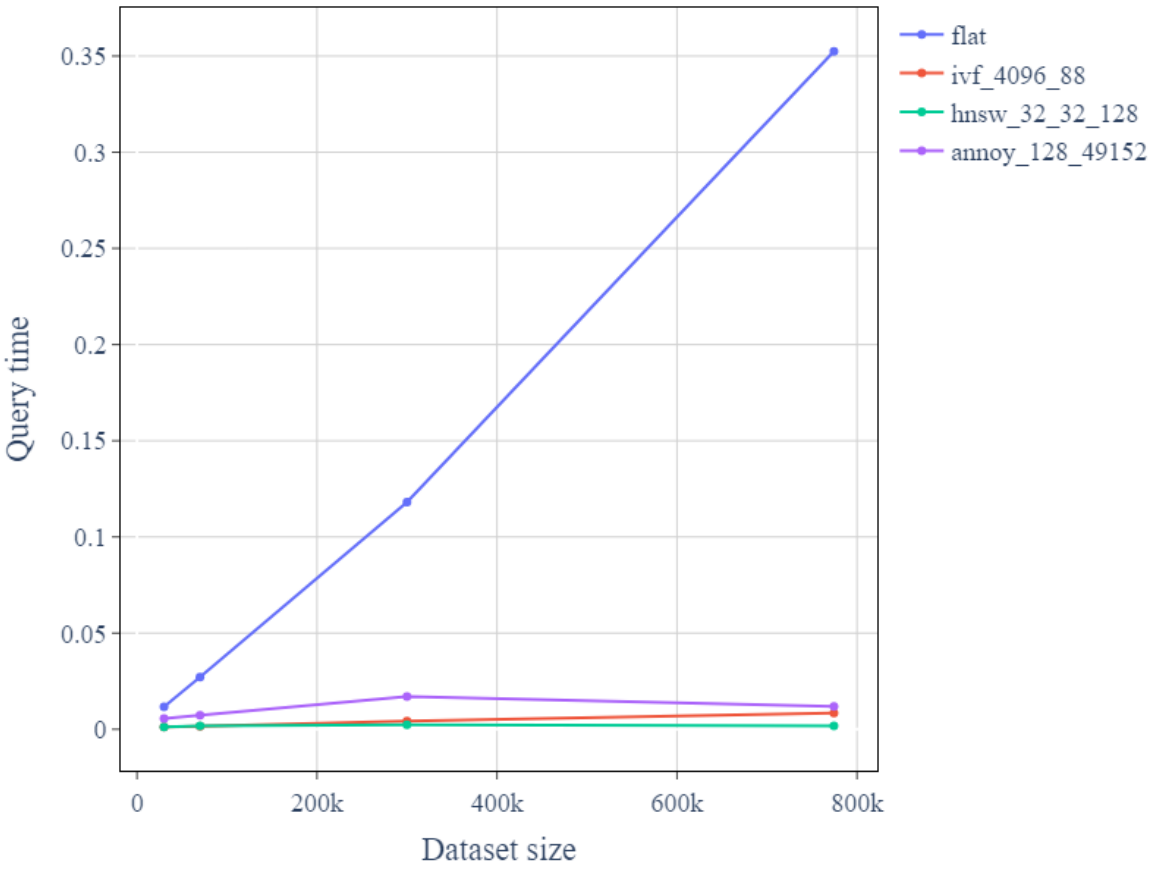


Figure 5.24: Query time of nearest neighbor methods

Using these settings, we evaluated the time complexity between KNN, IVF, ANNOY, and HNSW. We subsequently selected 30.000, 70.000, 300.000, and all items from the merged dataset (PolyvoreOutfits [22] and DeepFashion [71]) to run those algorithms. Figure 5.24 illustrates the result, showing that all investigated approximate methods run significantly faster than KNN and have nearly

constant runtime. This experiment has proven the scalability of approximate searching methods.

Additionally, we evaluated the query time, recall score and memory usage of these approximate methods on the PolyvoreOutfits dataset and compared them with those of KNN (Table 5.7). In summary, HNSW outperforms all other methods regarding query time while using nearly the same memory as KNN and maintaining a recall score of 0.98.

Table 5.7: Performance of searching methods on PolyvoreOutfits

Method	Query time (ms)	Recall	Memory (GB)
KNN	35.24	100	0.76
IVF	8.39	98.45	0.78
ANNOY	11.83	98.39	1.2
HNSW	**1.80**	**98.11**	**0.84**

CHAPTER 6
APPLICATIONS

This chapter presents our applications that assist customers in e-commerce based on the methods presented, including Smart Fashion Assistant, a system for online shopping support, and Magic Mirror, an application that allows users to try on clothing items virtually in an augmented reality scenario. We present an overview of each application, followed by the details of the conducted experiments, including a pilot study to evaluate their effectiveness and user satisfaction.

6.1 Magic Mirror

6.1.1 Overview

To demonstrate the efficiency of the DM-VTON framework, we developed an Augmented Reality (AR) application that can run on a local machine named Magic Mirror. This application simulates the experience of trying on garments in front of a mirror. However, instead of real garments, our application captures the user's image via the camera, applies virtual try-on processes, and displays the results on the screen. This protects the real garments from potential damage and enables faster try-on experiences. To enhance convenience, users can easily switch between different try-on garments and backgrounds using intuitive swiping gestures.

6.1.2 Implementation details

Figure 6.1 illustrates the fundamental architecture of Magic Mirror. We employ the Tkinter framework for the user interface display, Python's standard graphic user interface library. OpenCV is utilized to capture the user image through the camera. The captured frames are then passed to the processing components,

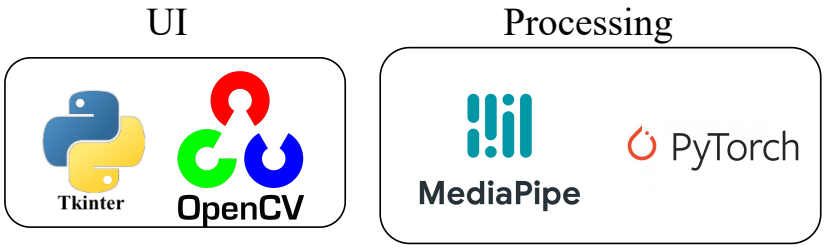


Figure 6.1: Fundamental components of Magic Mirror

which include MediaPipe for gesture detection and generating the background mask, as well as our DM-VTON model built on the PyTorch framework for the virtual try-on process. We deploy the application on an Acer nitro 5 AN517 laptop with an RTX 3050Ti GPU.

6.1.3 Usage and Examples

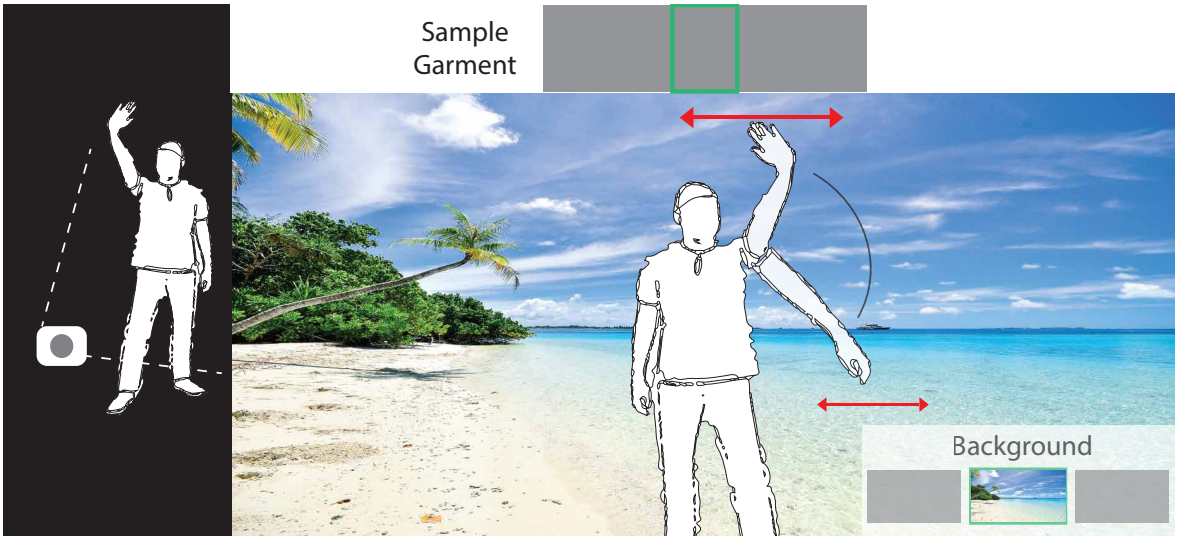
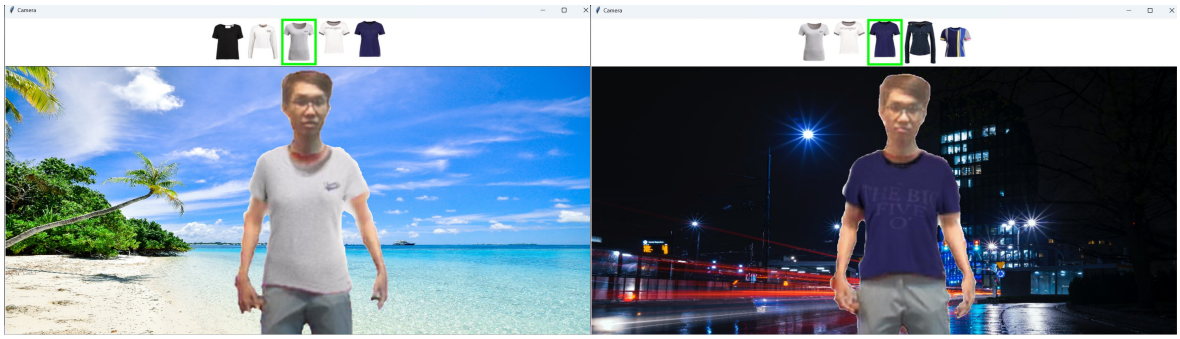


Figure 6.2: Instruction of Magic Mirror

The user interface (UI) of Magic Mirror consists of the following parts (as shown in Figure 6.2): the interaction area, the garment and the background display area. We prepare a list of in-shop garments and backgrounds that users can choose according to their needs.

The interaction area is the main display part of our application. First, the camera captures and displays the user's image in this area. And depending on the chosen outfit, the virtual try-on result is displayed correspondingly. If they want to change clothes, the users can use hand gestures to slide on the upper half of the interaction area to choose their desired item. And similarly, they can also use the slider gestures to change the background but on the lower half of the screen (as illustrated in Figure 6.2). Some examples of the capabilities of this application are demonstrated in Figure 6.3



(a) (b)

Figure 6.3: The UI of Magic Mirror

6.2 Smart Fashion Assistant System

6.2.1 Overview

We demonstrate the scenario where users find a particular garment that catches their eye while shopping online at home. They consider buying it but are unsure whether it looks good on them. We facilitate it by offering the Smart Fashion Assistance system. Using this system, users can try on such garments before purchasing. Upon try-on, the system recommends other fashion items that may suit their tastes and preferences. Users can also find new garments with some relative attributes (i.e. green, longer sleeves) to the selected garment by providing

feedback in natural language. Figure 6.4 illustrates how our system works.

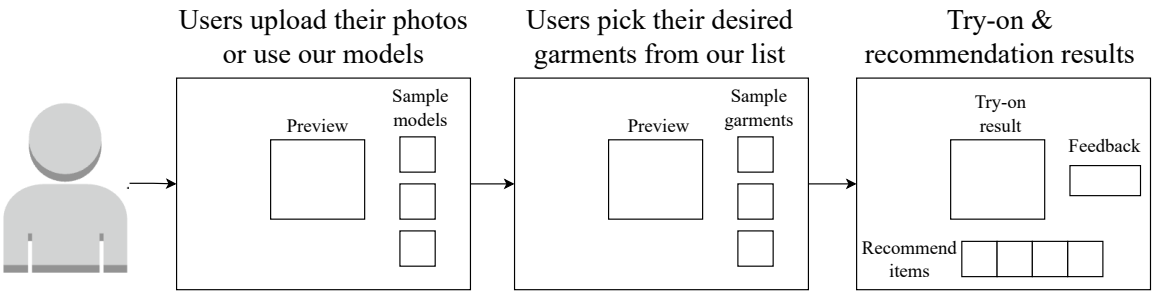


Figure 6.4: Demo overview of Smart Fashion Assistant System

6.2.2 Implementation details

Specifically, we develop the Smart Fashion Assistant system utilizing a client-server architecture Figure 6.5. We build the client user interface with HTML, Bootstrap, and Javascript to bring out the best user experiences on the browser. We develop our deep learning models in the back-end server utilizing the PyTorch framework [72]. We also incorporate the HNSW search algorithm using the FAISS implementation. The HTTPS entry points are handled by the FastAPI framework. We utilize Uvicorn, a Python implementation of an ASGI web server,

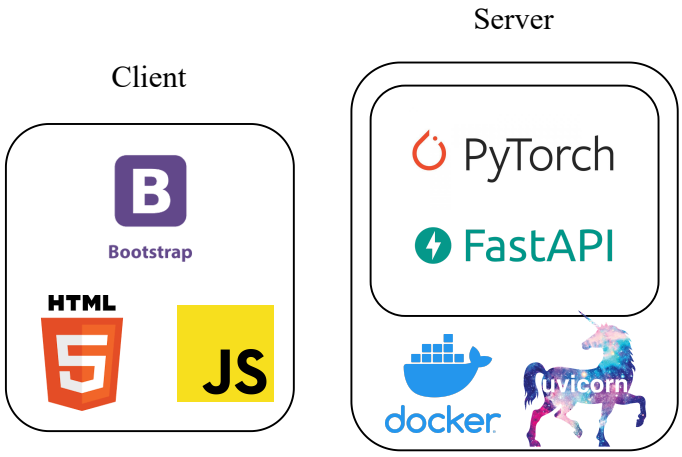


Figure 6.5: The architecture of Smart Fashion Assistance system

to serve the server. Finally, we package the entire server using Docker and deploy it on a cloud service with an NVIDIA A100 GPU.

6.2.3 Usage and Examples

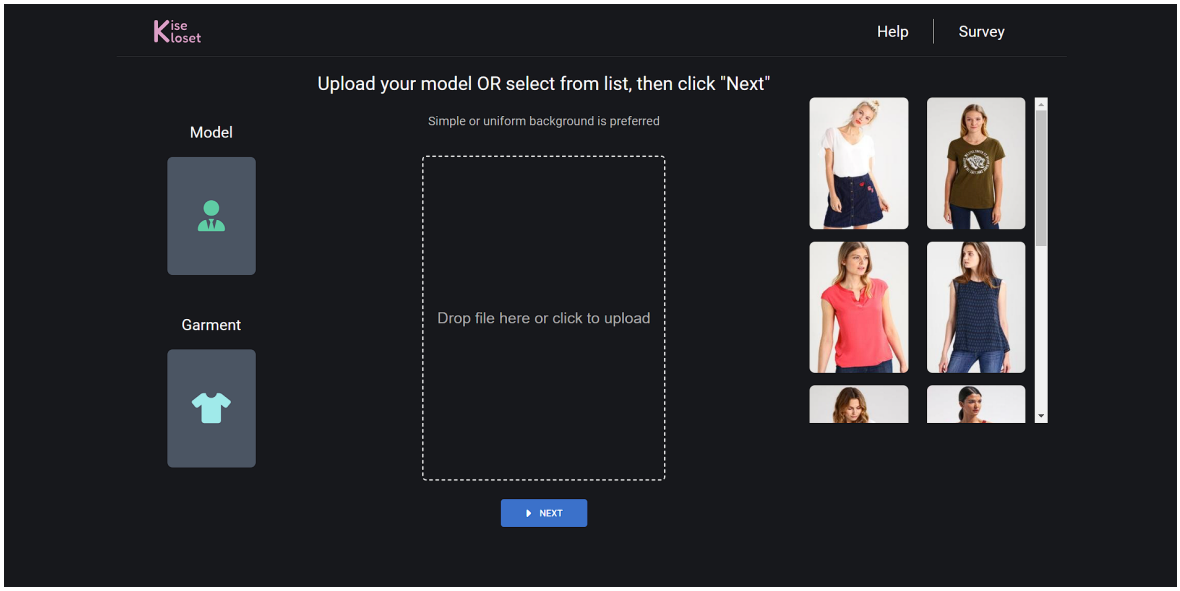


Figure 6.6: The initial UI of Smart Fashion Assistance system

Upon entering the website, users are presented with the user interface (UI) displayed in Figure 6.6. We prepare some human models beforehand as references for users. They can use those models or upload their photos as input images for the try-on framework. Figure 6.7 demonstrates the case where a user picks one of our provided models. After that, users can press the Next button below to advance to the next step.

Subsequently, users must upload or pick the garment they want to try on from our provided list on the right side (as shown in Figure 6.8). The left panel also displays the chosen model image in the previous step. Now users can press the Next button to view the try-on result.

Once users have chosen both the input person and garment image, they are

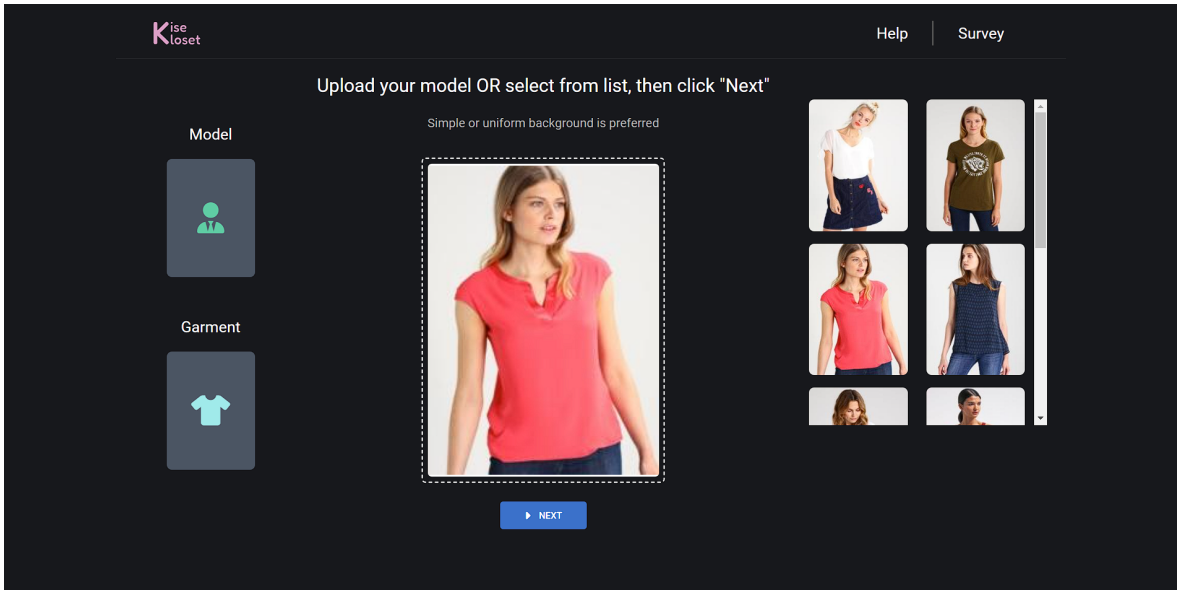


Figure 6.7: Choose the input person to try on. An example where a user picks one of our provided models

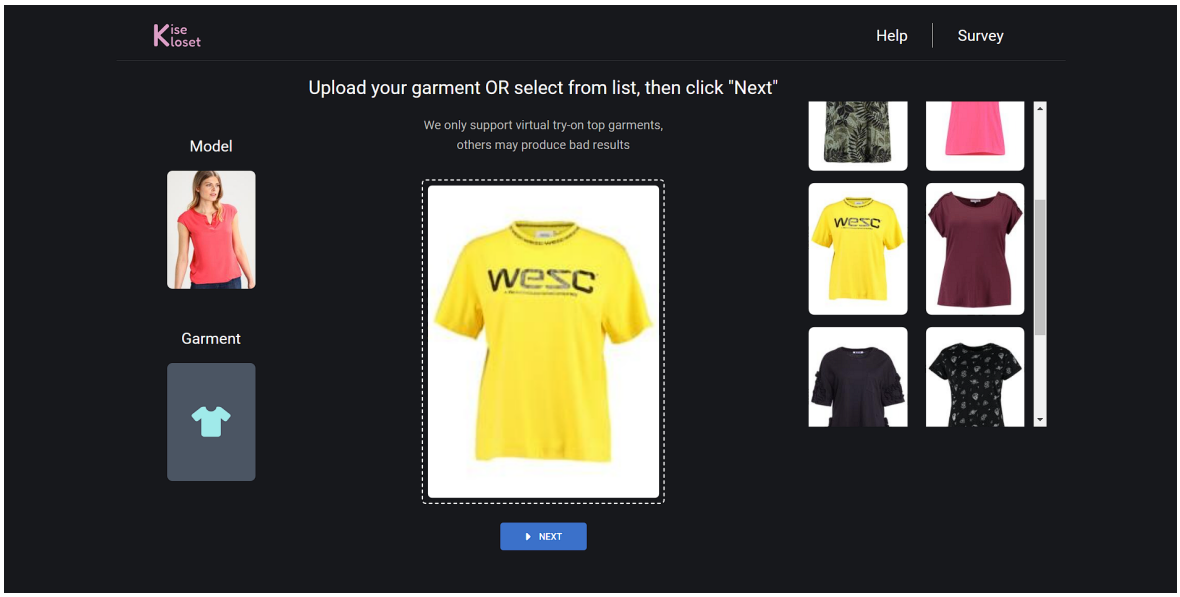


Figure 6.8: Choose the target garment to try on. Users must pick one of the garments provided on the right panel

presented with the try-on result, as shown in Figure 6.9. A slider in the middle allows users to compare the original and try-on images. Beneath the try-on result, we provide recommendations for the chosen garment. These recommen-

dations include items within the same category as the reference item and items from other categories that may catch users' attention. Users can try on these recommended items by clicking on them, but we currently only permit selecting items within the same category. As for items from other categories, we provided 2 options: Ton sur ton, items with the same color or textures, and, hipster style, items with mixed style and color. In case users want garments not included in the recommendation list but have some relation to the reference item, we offer a text box on the right side. Users can input their preferences; consequently, we provide new item recommendations based on their feedback.

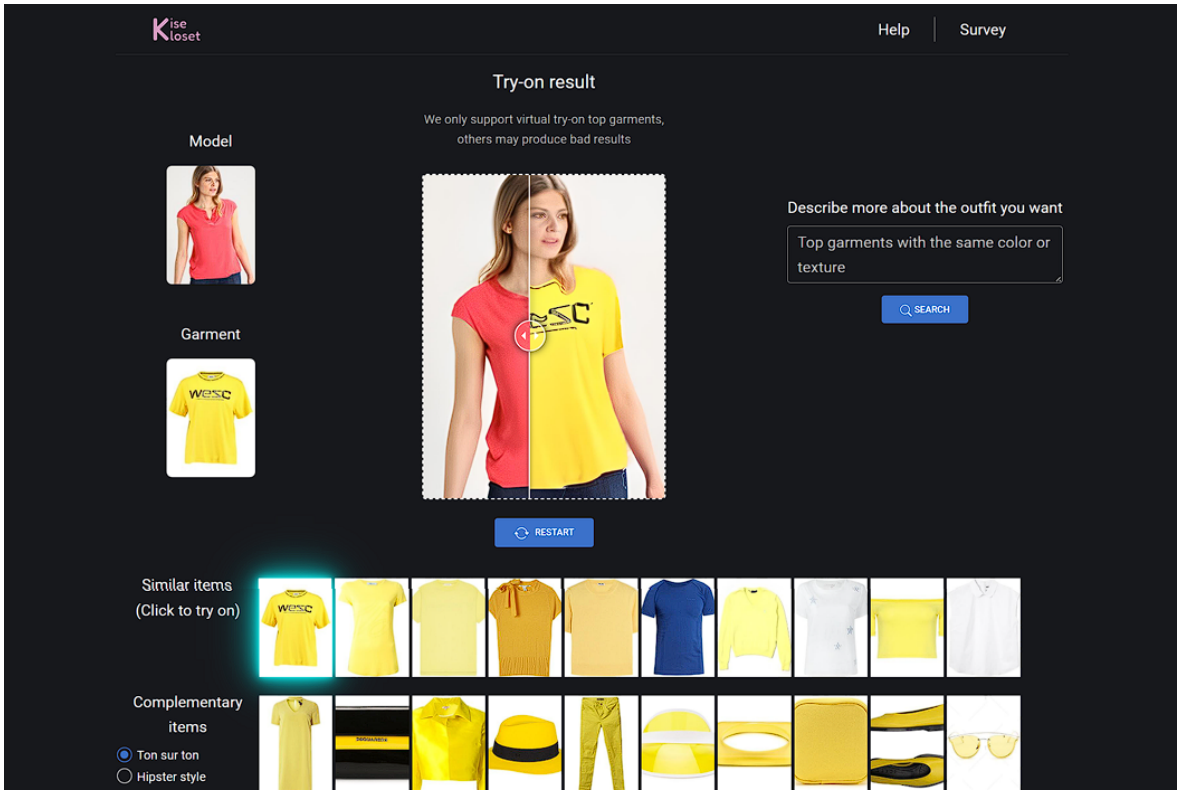


Figure 6.9: Try-on result and recommendation playground

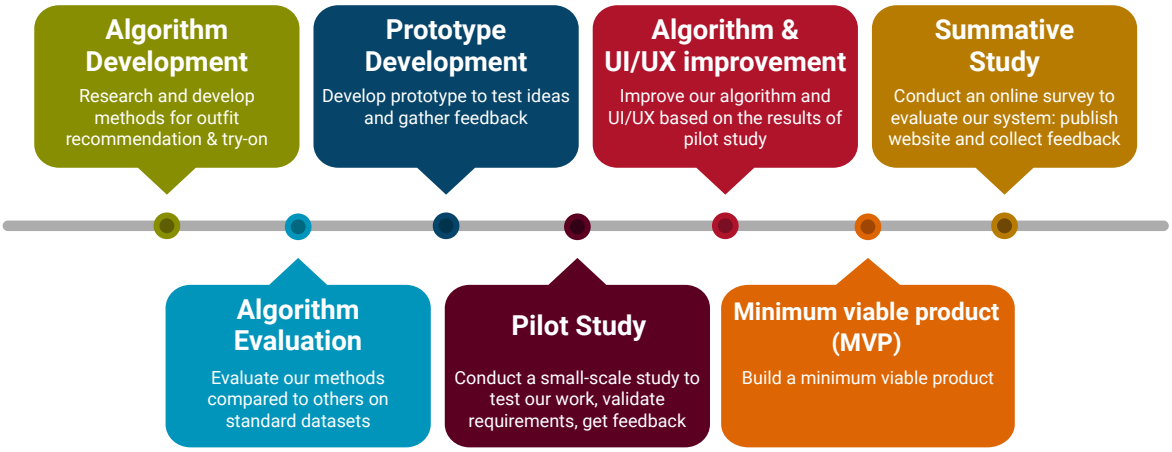


Figure 6.10: Workflow of development.

6.3 Experiments

6.3.1 Workflow Overview

We follow the workflow shown in Figure 6.10 during application research and development. The research workflow described here presents a comprehensive and iterative process to develop and refine algorithm and user experience (UX) design, leading to the creation of a viable product. The workflow comprises seven key stages: algorithm development, algorithm evaluation, prototype development, pilot study, algorithm & UI/UX improvement, minimum viable product (MVP), and summative study.

In the first stage, algorithm development, we focus on researching and developing algorithms for the virtual try-on and outfit recommendation problems. This stage involves reviewing existing literature, identifying knowledge gaps, and iterative experimentation to devise our approach. Following this, rigorous assessments of the developed algorithms' efficacy and performance are undertaken in the algorithm evaluation stage. We conduct various tests and benchmarking procedures on standard datasets to evaluate the effectiveness of our proposed methods. The works done during the first two stages create a solid foundation for

subsequent steps. The outcomes of these stages are documented and presented in chapter 3, chapter 4, and chapter 5.

Subsequently, the research progresses to the stage of prototype development. A rudimentary prototype is constructed, wherein the algorithm is integrated into a user interface. This version enables the execution of a pilot study with a small group of participants to gather valuable feedback about the algorithm's efficacy and the user interface/user experience (UI/UX) design. The detailed report of this pilot study can be found in subsection 6.3.2. The outcomes of this preliminary study help us identify existing issues and areas for enhancing the efficacy of our research.

After a small-scale study, we utilize the acquired insights to improve our system. Successive enhancements were implemented to optimize the algorithm's accuracy and performance, while concurrently refining the UI/UX to improve user satisfaction. Subsequently, integrating these improvements, a minimum viable product (MVP) was developed, representing a more advanced version of the initial prototype (see Figure 6.11).

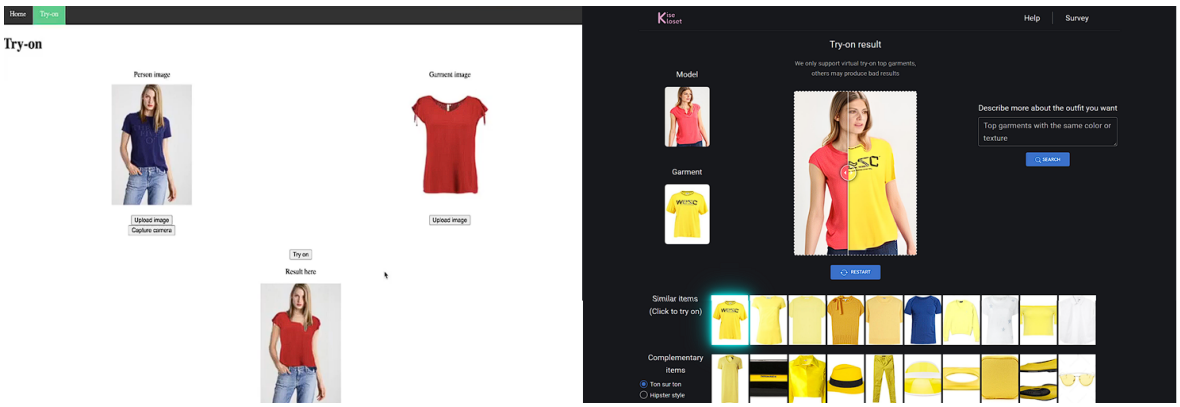


Figure 6.11: Smart Fashion Assistant UI comparison before and after improvements (left: prototype, right: MVP)

Finally, we conduct a large-scale summative study to assess our MVP from the

end-user's perspective. We open web access and invite online users to participate in trials and surveys. The valuable insights and opinions collected from these users play an important role in objectively evaluating the system. This study provides evidence of the system's effectiveness, validates its algorithmic capabilities, and measures user satisfaction, ensuring our application is viable. Further specifics about this study can be found in subsection 6.3.3.

In summary, this research workflow offers a systematic approach to developing and perfecting algorithms and UI/UX design, ensuring an efficient and user-friendly website is created. The iterative nature of the process allows for continuous refinement and optimization, leading to a well-validated product.

6.3.2 Pilot Study

We invited 12 participants who are university students and researchers in the 18-44 age range. We let the users experience our Smart Fashion Assistance system and collected their feedback.

Regarding the available garments, we prepared a collection of 20 garments taken from the VITON [1] and PolyvoreOutfits [22] datasets (see Figure 6.12). These garments were carefully selected to represent a variety of colours, shapes, and textures.

Each participant took part in a 10-minute session in which the participant was asked to perform a virtual try-on using our provided model images and virtual try-on on themselves directly captured from our camera. In terms of human models, we used 10 person images in the VITON [1] and DeepFashion [71] datasets (see Figure 6.12).

Upon completing the trial, we interviewed participants and asked for their feedback. Our primary objective was to evaluate how our system influenced their purchasing decisions. We also gathered their feedback about the output quality

(a) Human model samples

(b) Garment samples

Figure 6.12: Examples of data used in our pilot study.

and whether they prefer using the model images or their own images to enhance the overall user experience in the future.

Figure 6.13: Results obtained when users performed virtual try-on on our provided human models in the pilot study.

Most of the participants agreed that trying on clothes in various poses helped

them visualize the suitability of the garments before making a purchase decision. Specifically, 66.7% of the participants felt confident enough to make the purchasing decision after using our system, while the remaining 33.3% had doubts about the truthiness of the models. Moreover, 83.3% of the participants preferred using their images for try-on, as it provided a more realistic experience for them. On the other hand, the remaining 16.7% considered both options, as the provided models allowed them to see the best representation of the garments, such as with appropriate brightness and poses. Figure 6.13 illustrates some virtual try-on results on our provided models. Due to privacy issues, we did not capture the virtual try-on results on the participants' images.

We also received valuable feedback from participants on areas for improvement. In real-life conditions, the background, brightness, and quality of the captured images might not be suitable for trying on clothes, which is due to the fact that our train and test datasets only contain simple backgrounds and have proper brightness conditions. Thus, applying some pre-processing techniques such as segmentation and brightness equalization is necessary to address this issue. Additionally, participants also suggested that our system exhibited inconsistencies when dealing with complex poses such as half-turn poses or crossed-arm poses. Their feedback is useful in enhancing the user experience in the future.

6.3.3 Summative Study

We deployed our system at `http://selab.edu.vn:20440/` for online users to try out and provide feedback. Our survey primarily aimed to measure users' satisfaction in four key factors: ease of use, try-on quality, recommendation quality, response time, and response time. Figure 6.14 shows the average rating scores of each factor on a scale of 1 to 5 (1: very dissatisfied, 5: very satisfied) from the user study evaluation, which indicates that those users had a positive overall experience with our website.

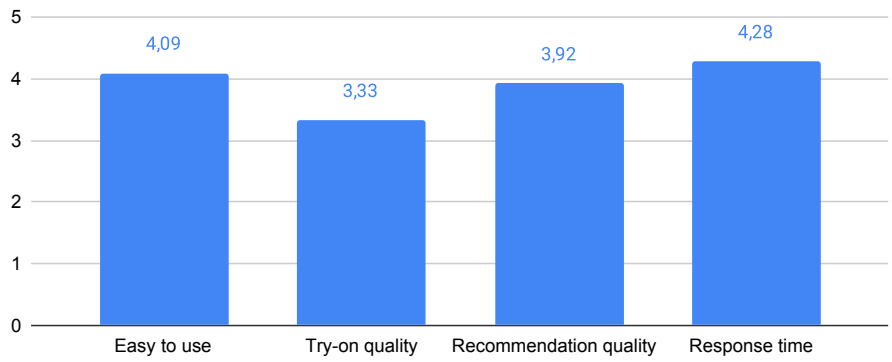


Figure 6.14: Average rating scores from the user study evaluation (1: very dissatisfied, 5: very satisfied)

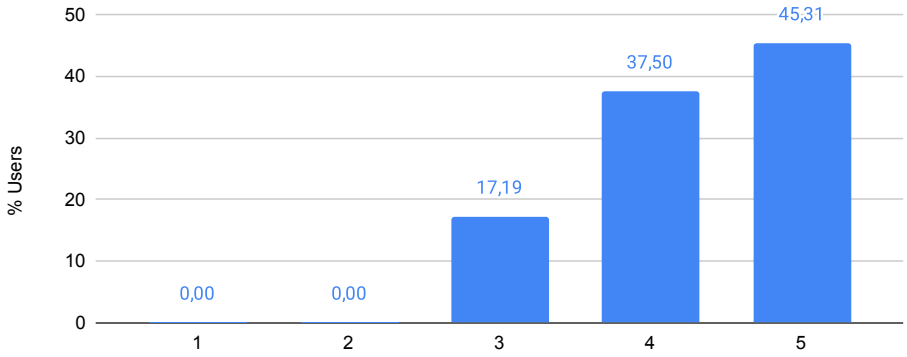


Figure 6.15: Rating scores of response time (1: very dissatisfied, 5: very satisfied)

Regarding the response times, it received the highest rating among the four factors, followed by ease of use. As shown in Figure 6.15, almost all users expressed their satisfaction with the response time. This aspect was our primary focus during the development of the deep learning models, as we prioritized optimizing the inference speed to deliver quicker results.

Regarding the quality of the results, most users either expressed satisfaction or had a neutral attitude towards the try-on output, as indicated in Figure 6.16 and Figure 6.17. Positive feedback highlighted the superb try-on quality, as it effectively assisted users in visualizing the garments in action, providing them with a valuable tool for making fashion choices.

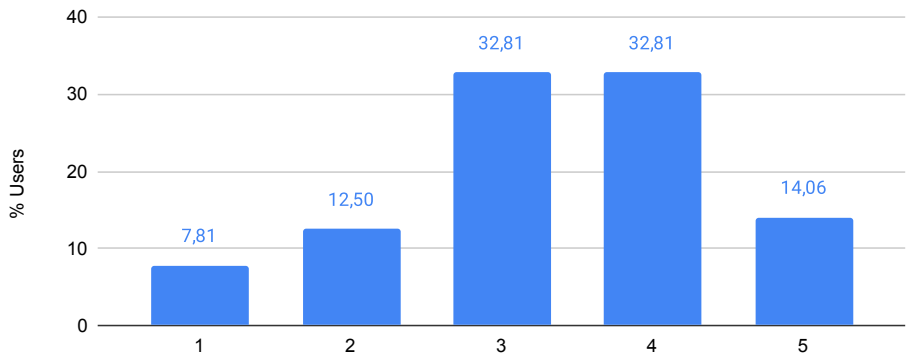


Figure 6.16: Rating scores of try-on quality (1: very dissatisfied, 5: very satisfied)

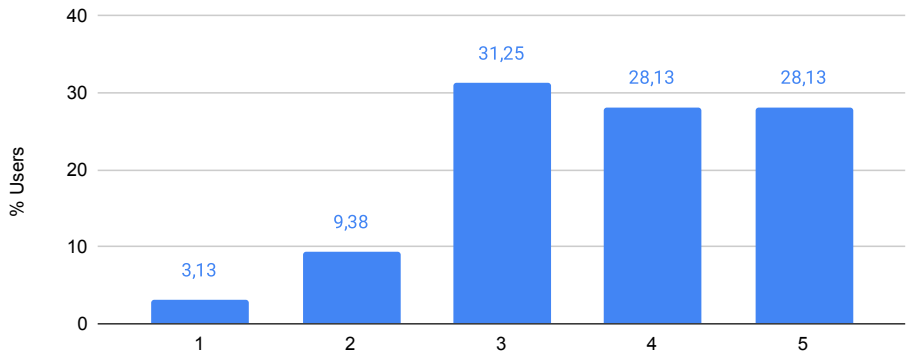


Figure 6.17: Statistics of user answers to questions: Do you agree that the virtual try-on feature helps you visualize the garment's appearance when you wear it? (1: completely disagree, 5: completely agree)

As for the recommendation results, the rating scores in Figure 6.18 indicated users' satisfaction with them, highlighting the positive impact of the feedback-based item recommendation on enhancing the overall user experience. This feature helped users receive recommendations that align with their unique tastes and preferences, making their shopping journey more enjoyable and efficient.

Moreover, the user study revealed that 84,375% of users found our system highly useful and effectively improved their online shopping experience. Furthermore, 65,625% of the users expressed their willingness to recommend our system to their friends and family, emphasizing the positive reception and satisfaction

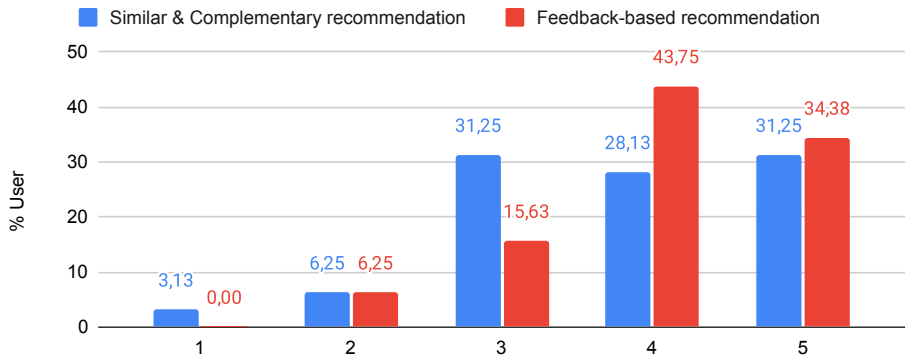


Figure 6.18: Rating scores on recommendation quality (1: very dissatisfied, 5: very satisfied)

among users.

Especially, we also get positive comments from experts in the industry, such as Mr. Le Tan Dang Khoa, a research software engineer at ViSenze, a leading company in Singapore known for providing AI solutions for visual commerce such as visual recommendation, augmented reality, and virtual try-on. He commends the system for its intuitive user interface, effectively catering to the diverse requirements of online shoppers. He further evaluates that with a few adjustments and fine-tunings, the system has the potential to transform into a minimum viable product suitable for integration within any e-commerce shop.

In conclusion, our system offers a comprehensive end-to-end solution for virtual try-on, accompanied by valuable additional features such as fashion recommendation, and text feedback-based search. The seamless and fast visual search ensures that similar and complementary items are promptly displayed without delays. As we continue to collect and analyze user feedback, we remain committed to improving our system further to ensure that our platform continues to deliver exceptional virtual try-on experiences and personalized fashion recommendations for our valued users.

6.4 Limitations & Discussions

Despite receiving positive feedback, it is essential to acknowledge the limitations of the Smart Fashion Assistance application. Through an analysis of negative feedback obtained from the summative study and previous evaluation experiments, several shortcomings have been identified.

Figure 6.19: The issues of existing parser-free virtual try-on methods (i.e. PF-AFN [6], FS-VTON [7])

First, the try-on results are unrealistic under certain circumstances, particularly in non-straight-arm poses or when there are differences in size between the model and the clothing items. Second, if the background color is not distinguishable from the input image, the model struggles to separate them and may blend the target garment with the background in the output image. And if the image is not captured in proper lighting conditions, the system may face difficulties in accu-

rately reconstructing the skin area. This is because our current training dataset only includes images with clean backgrounds, with similar lighting conditions. Third, the algorithm attempts to warp the input to align with the preserved region's boundary, resulting in texture distortion. It is important to note that similar issues have been observed in other SOTA parser-free methods (i.e. PF-AFN [6], FS-VTON [7]) also had similar problems (please refer to Figure 6.19).

Lastly, to enhance the application's capabilities, we also aim to incorporate additional clothes categories (e.g. accessories and pants), into the try-on feature. This will lead to a more comprehensive system that allows you to receive recommendations and try on complete outfits, not just a single item.

CHAPTER 7
CONCLUSION

We conclude our work by summarizing the results obtained and discussing the potential for future research directions. In this thesis, we solve two major problems in online fashion shopping: virtual try-on and fashion recommendation. Through intensive experiments, we prove the applicability and efficiency of our solutions and further demonstrate them in demo applications. The feedback we gained from the pilot study shows that our work still has some limitations, which we plan to improve in future works.

7.1 Summary

The principal contribution of the virtual try-on part lies in the introduction of a parser-free framework to boost processing speed while maintaining high-quality output and computational efficiency. Moreover, to address the inherent limitation of pose variation observed within the training images, we have devised the Virtual Try-on-guided Pose for Data Synthesis (VTP-DS) technique, which enriches the diversity of poses within the training data. VTP-DS automatically identifies input images with incorrect poses generated by the framework and generates additional images for those specific poses. Experimental results demonstrate that the proposed method achieves a frame rate of 40 frames per second on a single Nvidia Tesla T4 GPU, consuming a mere 37 MB of memory, while delivering output quality that is comparable to other state-of-the-art approaches. This, in conjunction with the pilot study, demonstrates the potential of our proposed method. It holds promise for real-time applications in augmented reality (AR), thereby paving the way for enhanced user experiences in the domain of virtual fashion.

During our study on fashion recommendation, we explore ways to retrieve intra-category similar items, inter-category complementary items and text feedback-guided items. The similar and text feedback-guided items retrieval problem can be solved effectively by utilizing the CLIP model. On the other hand, to tackle the complementary item retrieval task, we propose a transformer-based architecture and prove its effectiveness through numerous experiments. We also enhance the search pipeline's overall performance by integrating approximate algorithms, which leads to 30 times faster than the common K-Nearest Neighbor while producing nearly the same results.

By solving the above problems, we build a smart fashion assistance system along with an AR application to demonstrate the efficiency of our approaches.

7.2 Future works

Regarding our try-on framework, there exists significant room for improvement. Firstly, when dealing with challenging inputs (e.g. intricate human poses or complex garments), our synthesized try-on images may exhibit certain errors, including adhesive artifacts, failure to preserve the textures of the garments accurately, and distorted formations in the vicinity of the preserved region. Secondly, our training and testing dataset: VITON-Clean, primarily consists of images with a clean and uniform background, under similar lighting conditions. Consequently, the performance of our model in real-world scenarios remains uncertain. Lastly, our research focuses exclusively on upper-body garments, and we have not conducted any experiments involving outfits belonging to other categories, such as shorts and pants (upper-body) or full-body dresses.

Our try-on methods are currently applied to single-frame processing in video and augmented reality (AR) applications. Consequently, when users move rapidly, the results obtained may exhibit instability across frames. In the future, it is en-

couraged to focus on incorporating video processing techniques, such as memory-based approaches [73] to mitigate this issue. However, optimizing the processing speed to align with the requirements of AR scenarios is imperative, which remains a task for future investigations.

As for the inter-category complementary item recommendations, our current approach involves retrieving items from all other 10 categories excluding the category of the input item. However, this approach does not align with real-life scenarios where complete outfits typically consist of only 3 or 4 items. To address this issue, we could establish a predefined set of rules indicating which categories should be paired with each other, and then retrieve the target items based on those rules. However, this approach has a significant drawback, as certain categories may be compatible with each other in one outfit but not in others.

Moreover, when survey the users, we plan to separate the voting score of complementary item recommendations from that of similar items recommendations, to evaluate the effectiveness of those methods more accurately. And if the users give their consent, we will ask about their gender to further investigate if our method can satisfy female users, as they will be the main segment of users.

Furthermore, in our summative survey, we intend to distinguish the voting scores for complementary item recommendations from those for similar item recommendations. This will give a more precise evaluation of the effectiveness of these methods distinctively. Additionally, given the users' consent, we will collect information about their gender. Those data will help us to perform a deeper analysis to determine whether our approach can meet the preferences of female users, which is the primary segment of users that make outfit-based clothing purchases.

ACKNOWLEDGEMENTS

We thank Dr. Le Trung Nghia and Assoc. Prof. Tran Minh Triet being an incredible advisor for this thesis.

We thank Dr. Le Khanh Duy for helpful advice to improve our applications.

We thank Mr. Do Trong Le, Mr. Nguyen Nhat Minh Khoi and Mr. Tran Mai Khiem for their technical support.

We thank Nguyen Nhat Minh Khoi, Le Minh Tu, Le Pham Lan Anh, Nguyen Dai Nghia, Huynh Ngo Trung Truc, Hung Ngoc Phat, Vuong Gia Huy, Nguyen Quang Binh and other members of SELAB, University of Science - VNUHCM for participating in the pilot study.

We thank Mr. Le Tan Dang Khoa, Lê Minh Tú, Hoang-Khang Nguyen, Shruti Singh, Vinay, Duc Thanh, Zen, Bạch Sỹ Khang, Quan, and numerous anonymous users who generously participated in our summative survey.

We thank Ms. Trinh Thi Thuy for being the model in our presentation video.

We thank SELAB and the Faculty of Information Technology at the University of Science - VNUHCM for providing the GPUs during the work.

LIST OF PUBLICATIONS

1. **Khoi-Nguyen Nguyen-Ngoc***, **Thanh-Tung Phan-Nguyen***, Khanh-Duy Le, Tam V. Nguyen Minh-Triet Tran, Trung-Nghia Le. Dm-vton: Distilled mobile real-time virtual try-on. Accepted for IEEE International Symposium on Mixed and Augmented Reality (ISMAR), 2023.

2. Tuan-Luc Huynh, **Khoi-Nguyen Nguyen-Ngoc**, Chi-Bien Chu, Minh-Triet Tran, Trung-Nghia Le. Multilingual Communication System with Deaf Individuals Utilizing Natural and Visual Languages. In RIVF International Conference on Computing and Communication Technologies (RIVF), 2022.

3. Jie Qin and Shuaihang Yuan and Jiaxin Chen and Boulbaba Ben Amor and Yi Fang and Nhat Hoang-Xuan and Chi-Bien Chu and **Khoi-Nguyen Nguyen-Ngoc** and Thien-Tri Cao and Nhat-Khang Ngo and Tuan-Luc Huynh and Hai-Dang Nguyen and Minh-Triet Tran and Haoyang Luo and Jianning Wang and Zheng Zhang and Zihao Xin and Yang Wang and Feng Wang and Ying Tang and Haiqin Chen and Yan Wang and Qunying Zhou and Ji Zhang and Hongyuan Wang. SHREC'22 track: Sketch-based 3D shape retrieval in the wild. In Computers & Graphics, 2022.

4. Chiara Romanengo and Andrea Raffo and Silvia Biasotti and Bianca Falcidieno and Vlassis Fotis and Ioannis Romanelis and Eleftheria Psatha and Konstantinos Moustakas and Ivan Sipiran and Quang-Thuc Nguyen and Chi-Bien Chu and **Khoi-Nguyen Nguyen-Ngoc** and Dinh-Khoi Vo and Tuan-An To and Nham-Tan Nguyen and Nhat-Quynh Le-Pham and Hai-Dang Nguyen and Minh-Triet Tran and Yifan Qie and Nabil Anwer. SHREC 2022: Fitting and recognition of simple geometric primitives on point clouds. In Computers & Graphics, 2022.

REFERENCES

[1] Xintong Han, Zuxuan Wu, Zhe Wu, Ruichi Yu, and Larry S Davis. Viton: An image-based virtual try-on network. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7543–7552, 2018.

[2] Bochao Wang, Huabin Zheng, Xiaodan Liang, Yimin Chen, Liang Lin, and Meng Yang. Toward characteristic-preserving image-based virtual try-on network. In European Conference on Computer Vision (ECCV), pages 589–604, 2018.

[3] Xintong Han, Xiaojun Hu, Weilin Huang, and Matthew R Scott. Clothflow: A flow-based model for clothed person generation. In IEEE/CVF International Conference on Computer Vision (ICCV), pages 10471–10480, 2019.

[4] Han Yang, Ruimao Zhang, Xiaobao Guo, Wei Liu, Wangmeng Zuo, and Ping Luo. Towards photo-realistic virtual try-on by adaptively generating-preserving image content. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7850–7859, 2020.

[5] Thibaut Issenhuth, Jérémie Mary, and Clément Calauzenes. Do not mask what you do not need to mask: a parser-free virtual try-on. In European Conference on Computer Vision (ECCV), pages 619–635, 2020.

[6] Yuying Ge, Yibing Song, Ruimao Zhang, Chongjian Ge, Wei Liu, and Ping Luo. Parser-free virtual try-on via distilling appearance flows. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 8485–8493, 2021.

[7] Sen He, Yi-Zhe Song, and Tao Xiang. Style-based global appearance flow for virtual try-on. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3470–3479, 2022.

[8] Fashion e-commerce market value worldwide from 2023 to 2027. `https://www.statista.com/topics/9288/fashion-e-commerce-worldwide`. Accessed: 2023-06-29.

[9] Ian J. Goodfellow, Yoshua Bengio, and Aaron Courville. Deep Learning. MIT Press, Cambridge, MA, USA, 2016. `http://www.deeplearningbook.org`.

[10] Vincent Dumoulin and Francesco Visin. A guide to convolution arithmetic for deep learning. arXiv preprint arXiv:1603.07285, 2016.

[11] Cs231n: Deep learning for computer vision. http://cs231n.stanford.edu/. Accessed: 2023-07-17.

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. Advances in neural information processing systems (NeurIPS), 30:6000–6010, 2017.

[13] Transformers from scratch. https://peterbloem.nl/blog/transformers. Accessed: 2023-06-29.

[14] Text classification using bert. https://www.cse.chalmers.se/~richajo/nlp2019/l5/Text%20classification%20using%20BERT.html. Accessed: 2023-06-29.

[15] Benjamin Fele, Ajda Lampe, Peter Peer, and Vitomir Struc. C-vton: Context-driven image-based virtual try-on network. In IEEE/CVF Winter Conference on Applications of Computer Vision (WACV), pages 3144–3153, 2022.

[16] Shuai Bai, Huiling Zhou, Zhikang Li, Chang Zhou, and Hongxia Yang. Single stage virtual try-on via deformable attention flows. In European Conference on Computer Vision (ECCV), pages 409–425, 2022.

[17] Chao Lin, Zhao Li, Sheng Zhou, Shichang Hu, Jialun Zhang, Linhao Luo, Jiarun Zhang, Longtao Huang, and Yuan He. Rmgn: A regional mask guided network for parser-free virtual try-on. In International Joint Conference on Artificial Intelligence (IJCAI), pages 1151–1158, 2022.

[18] Baume & mercier updates their digital sales experience with the virtual try-on system. https://timeandtidewatches.com/baume-mercier-virtual-try-on-system. Accessed: 2023-06-29.

[19] Geenee ar. https://geenee.ar/. Accessed: 2023-06-29.

[20] Google introduces new ai virtual try-on feature. https://blog.google/products/shopping/ai-virtual-try-on-google-shopping. Accessed: 2023-06-29.

[21] Yen-Liang Lin, Son Tran, and Larry Davis. Fashion outfit complementary item retrieval. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3311–3319, 2020.

[22] Mariya I. Vasileva, Bryan A. Plummer, Krishna Dusad, Shreya Rajpal, Ranjitha Kumar, and David Forsyth. Learning type-aware embeddings for fashion compatibility. In European Conference on Computer Vision (ECCV), pages 405–421, 2018.

[23] Rohan Sarkar, Navaneeth Bodla, Mariya Vasileva, Yen-Liang Lin, Anurag Beniwal, Alan Lu, and Gerard Medioni. Outfittransformer: Outfit representations for fashion recommendation. In IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), pages 2262–2266, 2022.

[24] Hui Wu, Yupeng Gao, Xiaoxiao Guo, Ziad Al-Halah, Steven Rennie, Kristen Grauman, and Rogerio Feris. Fashion iq: A new dataset towards retrieving images by natural language feedback. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 11307–11317, 2021.

[25] Xiao Han, Licheng Yu, Xiatian Zhu, Li Zhang, Yi-Zhe Song, and Tao Xiang. Fashionvil: Fashion-focused vision-and-language representation learning. In European Conference on Computer Vision (ECCV), page 634–651, 2022.

[26] Product quantizers for k-nn tutorial part 1. https://mccormickml.com/2017/10/13/product-quantizer-tutorial-part-1/. Accessed: 2023-07-17.

[27] Tinghui Zhou, Shubham Tulsiani, Weilun Sun, Jitendra Malik, and Alexei A Efros. View synthesis by appearance flow. In European Conference on Computer Vision (ECCV), pages 286–301, 2016.

[28] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. Advances in neural information processing systems (NeurIPS), 30:6629–6640, 2017.

[29] The true cost of apparel returns: Alarming return rates require loss-minimization solutions. https://coresight.com/research/the-true-cost-of-apparel-returns-alarming-return-rates-require-loss-minimization-solutions/. Accessed: 2023-07-30.

[30] Ace your product recommendations to grow revenue. https://www.visenze.com/blog/2023/07/19/ace-your-product-recommendations-to-grow-revenue. Accessed: 2023-07-30.

[31] Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation ap-

plied to handwritten zip code recognition. Neural computation, 1(4):541–551, 1989.

[32] W. Rudin. Functional Analysis. International series in pure and applied mathematics. McGraw-Hill, 1991.

[33] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2018.

[34] Ke Gong, Xiaodan Liang, Dongyu Zhang, Xiaohui Shen, and Liang Lin. Look into person: Self-supervised structure-sensitive learning and a new benchmark for human parsing. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 932–940, 2017.

[35] Peike Li, Yunqiu Xu, Yunchao Wei, and Yi Yang. Self-correction for human parsing. IEEE Transactions on Pattern Analysis and Machine Intelligence, 44(6):3260–3271, 2020.

[36] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2d pose estimation using part affinity fields. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 7291–7299, 2017.

[37] RA Guler, Natalia Neverova, and IK DensePose. Densepose: Dense human pose estimation in the wild. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7297–7306, 2018.

[38] Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4401–4410, 2019.

[39] Taesung Park, Ming-Yu Liu, Ting-Chun Wang, and Jun-Yan Zhu. Semantic image synthesis with spatially-adaptive normalization. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 2337–2346, 2019.

[40] Warbyparker. `https://www.warbyparker.com`. Accessed: 2023-06-29.

[41] Interview : Our virtual try-on project for baume & mercier riviera watch. `https://hapticmedia.com/blog/riviera-try-on-interview`. Accessed: 2023-06-29.

[42] Amazon makes shopping easier with virtual try-on for shoes. `https://www.aboutamazon.com/news/retail/amazon-makes-shopping-easier-with-virtual-try-on-for-shoes`. Accessed: 2023-06-29.

[43] Artlabs. https://artlabs.ai/virtual-try-on. Accessed: 2023-06-29.

[44] Zalando brings a virtual fitting room pilot to millions of customers. https://corporate.zalando.com/en/technology/zalando-brings-virtual-fitting-room-pilot-millions-customers. Accessed: 2023-06-29.

[45] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. OpenAI blog, 1(8):9, 2019.

[46] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Effective conditioned and composed image retrieval combining clip-based features. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 21466–21474, 2022.

[47] Alberto Baldrati, Marco Bertini, Tiberio Uricchio, and Alberto Del Bimbo. Conditioned and composed image retrieval combining and partially fine-tuning clip-based features. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4959–4968, 2022.

[48] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. IEEE Transactions on Pattern Analysis and Machine Intelligence, 33(1):117–128, 2010.

[49] Sivic and Zisserman. Video google: A text retrieval approach to object matching in videos. In IEEE International Conference on Computer Vision (ICCV), pages 1470–1477, 2003.

[50] Erik Bernhardsson. Annoy: Approximate Nearest Neighbors in C++/Python, 2018. Python package version 1.13.0.

[51] Yu A Malkov and Dmitry A Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE Transactions on Pattern Analysis and Machine Intelligence, 42(4):824–836, 2018.

[52] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531, 2015.

[53] Yibing Song, Chao Ma, Lijun Gong, Jiawei Zhang, Rynson WH Lau, and Ming-Hsuan Yang. Crest: Convolutional residual learning for visual tracking. In IEEE International Conference on Computer Vision (ICCV), pages 2555–2564, 2017.

[54] Yining Li, Chen Huang, and Chen Change Loy. Dense intrinsic appearance

flow for human pose transfer. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 3693–3702, 2019.

[55] Hongyu Liu, Bin Jiang, Yibing Song, Wei Huang, and Chao Yang. Rethinking image inpainting via a mutual encoder-decoder with feature equalizations. In European Conference on Computer Vision (ECCV), pages 725–741, 2020.

[56] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 2117–2125, 2017.

[57] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 4510–4520, 2018.

[58] Deqing Sun, Stefan Roth, and Michael J Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. International Journal of Computer Vision (IJCV), 106:115–137, 2014.

[59] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In Medical Image Computing and Computer Assisted Intervention (MICCAI), pages 234–241, 2015.

[60] Justin Johnson, Alexandre Alahi, and Li Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In European Conference on Computer Vision (ECCV), pages 694–711, 2016.

[61] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, 2014.

[62] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In European Conference on Computer Vision (ECCV), pages 740–755, 2014.

[63] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 7464–7475, 2023.

[64] Ankan Kumar Bhunia, Salman Khan, Hisham Cholakkal, Rao Muhammad Anwer, Jorma Laaksonen, Mubarak Shah, and Fahad Shahbaz Khan. Person

image synthesis via denoising diffusion model. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 5968–5976, 2023.

[65] Richard Zhang, Phillip Isola, Alexei A Efros, Eli Shechtman, and Oliver Wang. The unreasonable effectiveness of deep features as a perceptual metric. In IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pages 586–595, 2018.

[66] Gagan Bhatia. keytotext.

[67] Cheng-I Lai. Contrastive predictive coding based feature for automatic speaker verification. arXiv preprint arXiv:1904.01575, 2019.

[68] Aristides Gionis, Piotr Indyk, Rajeev Motwani, et al. Similarity search in high dimensions via hashing. In International Conference on Very Large Data Bases (VLDB), pages 518–529, 1999.

[69] Hervé Jégou, Romain Tavenard, Matthijs Douze, and Laurent Amsaleg. Searching in one billion vectors: re-rank with source coding. In IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 861–864, 2011.

[70] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. IEEE Transactions on Big Data, 7(3):535–547, 2019.

[71] Ziwei Liu, Ping Luo, Shi Qiu, Xiaogang Wang, and Xiaoou Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pages 1096–1104, 2016.

[72] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. Advances in neural information processing systems (NeurIPS), 32:8026–8037, 2019.

[73] Xiaojing Zhong, Zhonghua Wu, Taizhe Tan, Guosheng Lin, and Qingyao Wu. Mv-ton: Memory-based video virtual try-on network. In ACM Multimedia (ACMMM), pages 908–916, 2021.

APPENDICES

CONTENT OF PAPERS

DM-VTON: Distilled Mobile Real-time Virtual Try-On

Category: Research

Abstract

The fashion e-commerce industry has witnessed significant growth in recent years, prompting exploring image-based virtual try-on techniques to incorporate Augmented Reality (AR) experiences into online shopping platforms. However, existing research has primarily overlooked a crucial aspect - the runtime of the underlying machine-learning model. While existing methods prioritize enhancing output quality, they often disregard the execution time, which restricts their applications on a limited range of devices. To address this gap, we propose Distilled Mobile Real-time Virtual Try-On (DM-VTON), a novel virtual try-on framework designed to achieve simplicity and efficiency. Our approach is based on a knowledge distillation scheme that leverages a strong Teacher network as supervision to guide a Student network without relying on human parsing. Notably, we introduce an efficient Mobile Generative Module within the Student network, significantly reducing the runtime while ensuring high-quality output. Additionally, we propose Virtual Try-on-guided Pose for Data Synthesis to address the limited pose variation observed in training images. Experimental results show that the proposed method can achieve 40 frames per second on a single Nvidia Tesla T4 GPU and only take up 37 MB of memory while producing almost the same output quality as other state-of-the-art methods. It also demonstrates the potential of integrating image-based virtual try-on into real-time AR applications.

Index Terms: Mixed/augmented reality—Real-time systems—Virtual try-on—Knowledge distillation

1 Introduction

In recent years, the fashion industry, particularly fashion e-commerce, has witnessed significant advancements. Despite these improvements, customers still face the limitation of having to physically visit stores to try on their wanted clothes. As a result, there is a growing interest in virtual try-on [4, 9, 25, 29], which demonstrates the potential to enhance shopping experiences by integrating Augmented Reality (AR).

To the best of our knowledge, existing works on image-based virtual try-on do not put their concern about the complexity of their models. They depend on many information like the human semantic segmentation map (body-parser map) and the human pose, denoted as *human representation*, to enhance the output quality. As a result, they take too much time for inference, preventing them from being applied in real-time applications (ACGPN [27], SDAFN [1] and C-VTON [3] as in Fig. 1). Recent methods [4, 12] have removed the dependency on human representation, thus, reducing the run time considerably. However, they still suffer from using large memory footprints, impacting the requirements for AR devices to run them (PF-AFN [4] and FS-VTON [9] as in Fig. 1).

Addressing those issues, we propose a novel framework, **D**istilled **M**obile real-time **V**irtual **T**ry-**ON (DM-VTON)**, to achieve faster run time and less memory consumption while also producing results of the same quality. The framework consists of two networks: a Teacher network and a Student network. The Teacher network serves as the source of information and guides the Student network through Knowledge Distillation [11]. Specifically, the Teacher is trained with the objective of generating the try-on result from the person image, target garment, and human representation (parser-based approach). Because we only use the Teacher during the training phase, we build it using the state-of-the-art virtual try-on architecture [9]. Taking advantage of the synthetic result and the original garment, the Student can learn to reconstruct the original realistic image without the need for human representation (parser-free approach). As we use the Student network for inference, we consider the trade-off between its runtime and output quality. We introduce two components in the Student network: the Mobile Feature Pyramid Network (MFPN) and Mobile Generative Module (MGM). Both components are based on the lightweight MobileNet [23] architecture, which has been proven to achieve higher throughput and smaller memory usage while producing comparable results to other architectures in the same work.

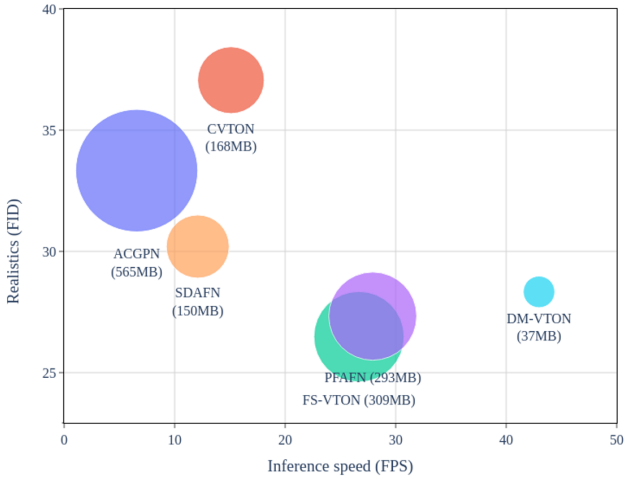


Figure 1: The comparison of our method (DM-VTON) and SOTA methods on VITON test set [8] in terms of realistic results (FID [10], lower is better), inference speed (FPS, higher is better), and memory usage. The size of each bubble represents the memory footprint. FPS is measured using a single Nvidia Tesla T4 GPU.

Besides, common virtual try-on datasets [8, 20] only contain a limited range of pose variations. That makes the models trained on those datasets suffer from overfitting. As a result, they perform poorly in real-life scenes. To overcome this problem, we introduce **V**irtual **T**ry-on-guided **P**ose for **D**ata **S**ynthesis (VTP-DS), an automatic pipeline to enrich the diversity of the poses in the mentioned datasets. The pipeline has two key ideas: self-checking the results by calculating the Object Keypoint Similarity (OKS) [18], and synthesizing new images by using a diffusion network [2]. Given a virtual try-on framework, the pipeline can automatically identify input images where the framework generates results with incorrect poses and extract the corresponding pose information. Then those poses are used as guidance to synthesize new person images from a single image of that person. By utilizing the VTP-DS pipeline, we can enrich the datasets with a wider range of pose variations, thus enhancing the training process and improving the performance of virtual try-on models in real-life scenarios.

We conducted experiments comparing our DM-VTON framework with other state-of-the-art (SOTA) methods in terms of inference speed, memory usage, and the realisticness of the output. During the experimentation, we carefully evaluated the trade-off between those factors. As depicted in Fig. 1, our DM-VTON framework

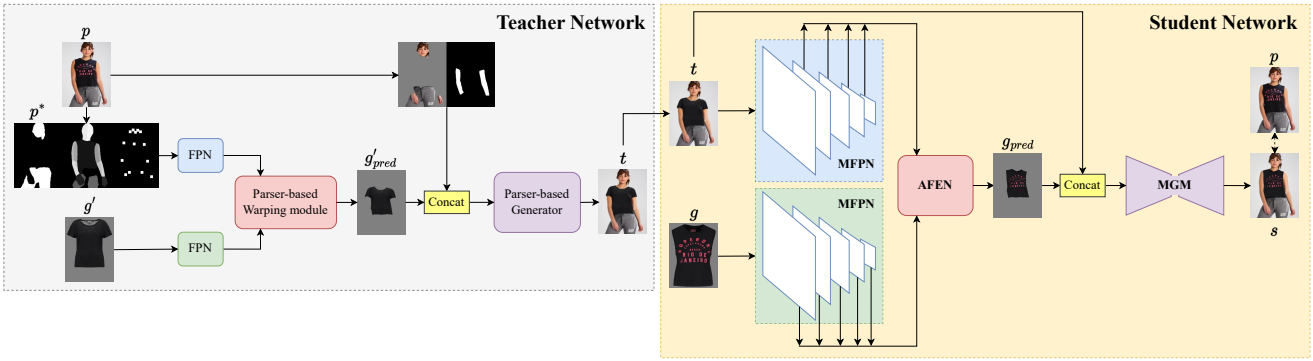


Figure 2: Overview architecture of the proposed Distilled Mobile Real-time Virtual Try-On (DM-VTON) framework. The parser-based Teacher network generates a synthetic image as the input for training the Student network.

outperforms all existing state-of-the-art methods regarding inference speed and memory usage while maintaining an equal quality of results. In summary, our contributions are as follows:

- To address the limitation of image-based virtual try-on that concerns inference time and memory consumption, we propose the DM-VTON framework based on knowledge distillation learning. By developing a lightweight Student network, we can reduce the number of floating point operations (FLOPs) and parameters of the model, thus making it easier to deploy and operate on AR devices.
- We introduce VTP-DS, an automatic fashion-pose data generation pipeline designed to enrich existing fashion datasets by synthesizing new person poses from a single image of that person. The pipeline utilizes a virtual try-on framework to identify challenging poses and subsequently generates additional images for those specific poses. These generated images are then utilized in the training process, improving the framework's performance in real-world scenarios.
- Experiment results show that DM-VTON achieves faster inference and more efficient resource utilization while producing comparable result quality with existing state-of-the-art methods, which proves the efficiency of our proposed framework.

2 Related Work

Image-based virtual try-on techniques can be classified into two categories: parser-based and parser-free approaches. Both of them typically involve three steps: extracting the intrinsic input features, warping the input garment to fit the clothing area of the person image, and performing the replacement using a generative model.

As for parser-based virtual try-on methods, they require human representation, including the body-parser map and human pose, to calculate the warping transformation matrix. The very first methods that pave for this approach are VITON [8] and CP-VTON [27]. To improve the output quality, ACGPN [30] also warps the body-parser map along the target garment. SDAFN [1] reduces the need for the parser map but still needs the human pose, though. Recently, ClothFlow [6] is the first method that uses the appearance flow to guide the warping procedure, and this approach is still used in current SOTA methods [4, 9].

On the other hand, the parser-free approach only requires an input garment and a person image for inference. Thus, this makes the inference process much faster and more independent from other intermediate models. WUTON [12], the pioneering parser-free method, produces noticeable artifacts due to using the same input-output pairs for both Teacher and Student networks [4]. Addressing that issue, PF-AFN [4] introduces a new Knowledge Distillation-based training pipeline, in which the Student network takes the Teacher output as its input and has its output supervised by the original images. This training methodology has become the standard for subsequent parser-free methods. RMGN [16] improves the generation part by using SPADE blocks [21], while FS-VTON [9] improves the warping part by using StyleGan blocks [14].

There are also methods that aim to perform virtual try-on on a sequence of frames. These methods use techniques like memory-based [32] or optical flow [15] to keep the temporary consistency between the frames. However, these methods are still based on the parser-based approach, which takes considerable time to calculate the human representation.

In this paper, we adopt the parser-free approach to prioritize speed, as calculating human representation is a time bottleneck in the try-on process. However, we take a distinct step from existing methods by modifying the Student network for improved speed and reduced memory consumption while keeping the parser-based approach in the Teacher network to preserve the output quality.

3 Method

3.1 Overview

Our objective is to generate an image of a person wearing a specific garment while preserving the rest of the image. To achieve this goal, we adopt the knowledge distillation training pipeline [4, 9, 11, 12, 16] to develop a Distilled Mobile Real-time Virtual Try-On (DM-VTON) framework (see Fig. 2). Our proposed DM-VTON consists of two networks: Teacher and Student networks. Both include three main components: feature extractor, clothes-warping module, and generator.

The Teacher network aims to produce the virtual try-on result using the parser-based training process. The Student network then utilizes the Teacher network to generate synthetic input images, enabling the Student network to be supervised by the original images without relying on human representation. To ensure high-quality output, the Teacher network is built upon SOTA virtual try-on models. As we focus on inference speed, we propose lightweight components for the Student network.

3.2 Teacher Network

The main purpose of this network is to generate a synthetic person image that serves as the input for the Student training process. Furthermore, the Teacher also helps this process through a knowledge distillation scheme. In particular, we take advantage of the SOTA method of virtual try-on task: FS-VTON [9]. As shown in Fig. 2, it incorporates two feature pyramid networks (FPN) [17] constructed from residual blocks, enabling the extraction of features from the human representation p^* and garment image g'. To achieve the

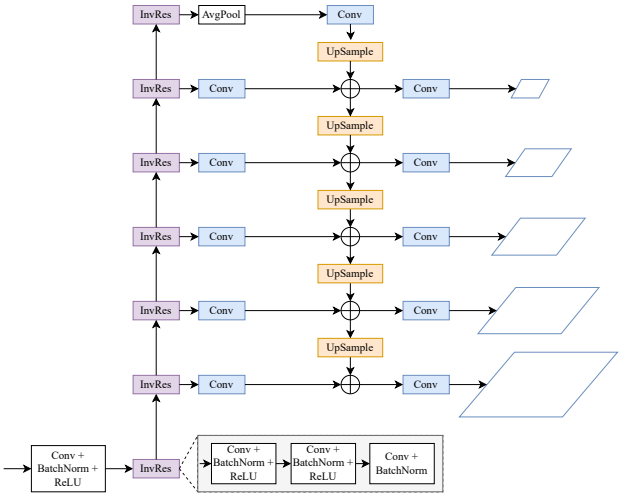


Figure 3: Mobile Feature Pyramid Network architecture

garment deformation functionality, the Teacher network utilizes a style-based global appearance flow estimation network that uses modulated convolution [14]. This network first predicts a coarse appearance flow via extracted global style vector and then refines it locally. The last flow thus can capture the global and local correspondence between the garment and the target person. This makes the Teacher network more robust against the problems of detail-preserving and large misalignment. Finally, the warped clothes and the preserved region on the human body are concatenated as the generator input for try-on result generation. The generator of our Teacher network follows the encoder-decoder architecture with skip connections, which have been proven effective in detail preservation.

Because the inputs of the parser-based model (i.e., human representation) contain more semantic information when compared to those in the parser-free model, we employ an adjustable knowledge distillation learning scheme [4] with a distillation loss to guide the feature extractor of the Student network:

$$L_{dis} = \psi \sum_{i=1}^{N} \|t_{p_i} - s_{p_i}\|_2, \tag{1}$$

$$\psi = \begin{cases} 1, \ if \ \|t-p\|_1 < \|s-p\|_1 \\ 0, \ otherwise \end{cases}, \tag{2}$$

where t_{p_i} and s_{p_i} denote the feature maps at the i-th scale extracted from the human representation p^* within the Teacher network and the synthetic image t within the Student network; t, s are the try-on result of the Teacher and Student, respectively; p is the person image ground truth.

3.3 Student Network

We propose a parser-based approach for synthesizing try-on images with increased speed compared to previous methods while ensuring accuracy. As shown in Fig. 2, our Student network consists of three key components: Mobile Feature Pyramid Network (MFPN), Appearance Flow Estimation Network (AFEN), and Mobile Generative Module (MGM). These components synergistically collaborate to extract features, manipulate garments through deformation, and generate try-on images. The AFEN introduced by Ge et al. [4] proved effective in deforming garments by using appearance flow estimates from pyramid features. The MFPN and MGM are built upon the architecture of MobileNetV2 [23] with Inverted Residual blocks specifically designed to optimize computational efficiency and model size.

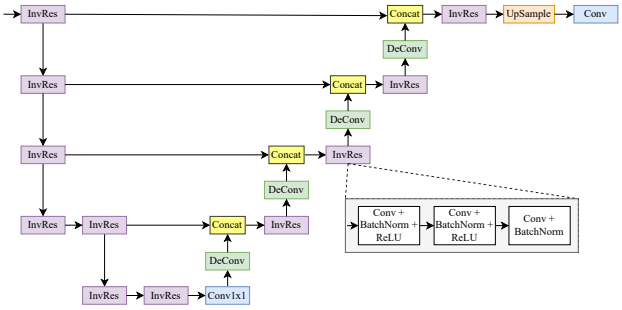


Figure 4: Mobile Generative Module architecture

3.3.1 Mobile Feature Pyramid Network

As shown in Fig. 3, MFPN incorporates the architecture of MobileNetV2 with Inverted Residual blocks [23] to a Feature Pyramid Network. By leveraging the capabilities of two MFPN blocks, we extract two-branch N-level feature maps from person and garment images within a parser-free network. These features are fed into the Appearance Flow Estimation Network (AFEN) to predict the appearance flow map for garment deformation.

3.3.2 Appearance Flow Estimation Network

This component aims to deform the garment to fit the human pose while preserving the texture. Following the work of Ge et al. [4], we adopt an appearance flow estimation network (AFEN) comprising subnetworks equipped with varying sizes of convolution layers. These subnetworks are responsible for estimating flows based on extracted multi-level feature maps. The outcome of this network can capture the long-range correspondence between the garment image and the person image, effectively minimizing issues related to misalignment. To enhance the preservation of clothing characteristics, this module is optimized with the second-order constraint:

$$L_{sec} = \sum_{i=1}^{N} \sum_{t} \sum_{\pi \in N_t} CharLoss\left(f_i^{t-\pi} + f_i^{t+\pi} - 2f_i^t\right), \tag{3}$$

where f_i^t denotes the t-th point on the i-th scale flow map; N_t is the set of horizontal, vertical, and diagonal neighborhoods around the t-th point; and *CharLoss* denotes generalized Charbonnier loss [26].

3.3.3 Mobile Generative Module

To synthesize the entire try-on image from the warped image and target person image, we develop a Mobile Generative Module, the integration of the architectural principles of UNet [22] and MobileNetV2 [23] as illustrated in Fig. 4. The primary objective behind the design of this generator is to reduce both the computational burden and the model's overall size.

3.3.4 Loss Function

During training, we optimize the warping module separately in the first stage and then train together with the generator in the last stage. The loss function used in the first stage is defined as:

$$L^{warp} = \lambda_l^{warp} L_l^{warp} + \lambda_{per}^{warp} L_{per}^{warp} + \lambda_{sec} L_{sec} + \lambda_{dis} L_{dis}, \tag{4}$$

where $L_l^{warp} = \|g_{pred} - p \odot m_{gt}\|$ denotes pixel-wise L1 loss, $L_{per}^{warp} = \sum_i \|\Phi_i(g_{pred}) - \Phi_i(p \odot m_{gt})\|$ is the perceptual loss [13], L_{sec}^{warp} is the smooth loss (second-order constrain), L_{dis}^{warp} is the distillation loss, g_{pred} is warped garment. p is the person image ground truth with the garment mask m_{gt}; Φ_i denotes the i-th block of pre-trained VGG19 [24].

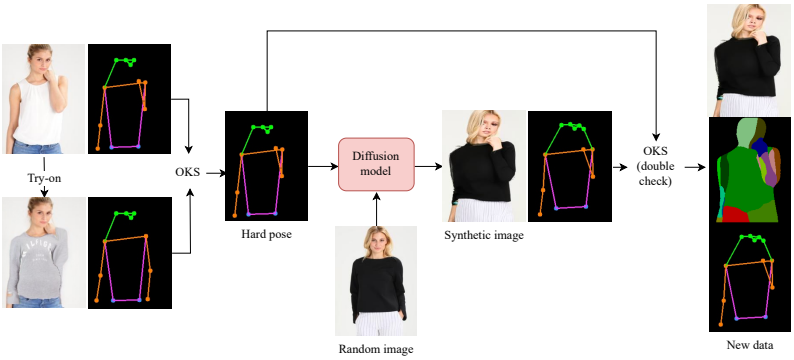


Figure 5: Overview of Virtual Try-on-guided Pose for Data Synthesis pipeline.

With the generative module, we also apply L1 loss perceptual loss [13] between the synthesized image and the ground truth image to supervise the training process of MGM:

$$L^{gen} = \lambda_l^{gen} L_l^{gen} + \lambda_{per}^{gen} L_{per}^{gen}, \tag{5}$$

where $L_l^{gen} = \|s - p\|$ is L1 loss and $L_{per}^{gen} = \sum_i \|\Phi_i(s) - \Phi_i(p)\|$ is the perceptual loss [13]. s and p are the generated output of the Student network and the person image ground truth, respectively.

In practice, we empirically set $\lambda_l^{warp} = 1$, $\lambda_{per}^{warp} = 0.2$, $L_{sec}^{warp} = 6$, $L_{dis}^{warp} = 0.04$, $\lambda_l^{gen} = 5$, $\lambda_{per}^{gen} = 1$. The overall loss function when training the whole model in the last stage is:

$$L = 0.25 * L^{warp} + L^{gen}. \tag{6}$$

3.4 Virtual Try-on-guided Pose for Data Synthesis

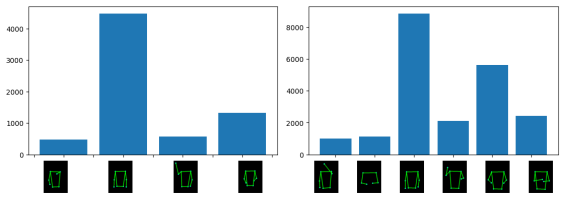
By using the K-Means clustering algorithm, we observe that the original VITON dataset [8] is mainly composed of images with straight-arm poses (as in Fig. 6(a)). This bias creates a challenge as models trained on such data are prone to overfit and perform poorly on images with different upper-body poses. To tackle this problem, we propose the Virtual Try-on-guided Pose for Data Synthesis (VTP-DS) pipeline. With the objective of improving the existing virtual try-on framework, the pipeline incorporates two key ideas: automatically detecting poorly performed poses using the Object Keypoint Similarity (OKS) metric [18] and synthesizing new training data specifically targeting those poses. The overview of the VTP-DS pipeline is illustrated in Fig. 5.

Given an input image containing a person, we extract that person's pose by using the YOLOv7 pose estimation method [28]. Then, we utilize our trained DM-VTON model to perform virtual try-on on the input image. The extracted pose from the resulting image is compared with the pose of the input image using a customized OKS metric Equation 7.

$$\frac{1}{|P|} \sum_{i \in P} \exp\left(\frac{-d_i^2}{2s^2 k_i^2}\right), \tag{7}$$

where P denotes the set of arm and hand keypoints, while the original formula uses all keypoints; d_i denotes the Euclidean distance between the keypoint i of two poses; s denotes the total area containing the pose; k_i is the constant provided by Lin et al. [18] to represent the standard deviation for keypoint i. As we focus on distinguishing different upper-body poses, only the arm and hand keypoints contribute to the formula. If the OKS score falls below a specified threshold $t = 0.9$, it is identified as a hard pose.

Once identifying a hard pose, we randomly pick a person image from the VITON dataset. Then it leverages Bhunia's Diffusion model [2] to synthesize a new image of the person in the corresponding pose. To ensure the accuracy of the synthesized image,



(a) VITON (b) VITON + synthesized data

Figure 6: Pose distribution in VITON dataset [8].

we perform a double-check using the OKS metric to verify the correctness of the output pose. Finally, DensePose [5] is utilized to generate the body-parser map of the synthesized image.

To synthesize additional data for training networks, we initially collected videos from Youtube, specifically focusing on posing or catwalk videos. These videos had varying durations, ranging from 1 to 10 minutes. Subsequently, we extracted individual frames from these videos, which served as the input for our VTP-DS pipeline. After that, we manually removed low-quality results, resulting in 14,314 high-quality synthesized images for training the networks.

To access the quality of synthesized images, we use the K-means algorithm combined with our modified OKS metric. As details of the pose clustering results shown in Fig. 6, data in the VITON training set mainly focuses on poses with simple poses (i.e. arms are less covered, low rotation amplitude). Meanwhile, when combined with our synthesized images, new pose clusters appear and the concentration of data in groups is less imbalanced, which can help to train robust virtual try-on models.

4 Experiments

4.1 Detailed Implementation

The Teacher and Student network training process follows the same strategy with two stages: the first stage only trains the warping module, while the latter trains the entire network. Both were under the same setting and carried out on a single Nvidia A100 GPU. We trained the model for 100 epochs with the initial learning rate is 5×10^{-5}, which decays linearly after the first 50 epochs.

4.2 Experimental Settings

VITON [8], the most popular dataset for evaluating virtual try-on, was used to evaluate methods. It contains 16,253 frontal-view upper-body woman and top clothing image pairs with 256×192 resolution. However, we followed the work of Han et al. [6] to filter out duplicates and ensure no data leakage happens, remaining 6,824 training image pairs and 416 testing image pairs in the cleaned VITON

Table 1: Quantitative results between DM-VTON and SOTA virtual try-on methods. The † marker indicates the results measured by the generated images provided by the authors. The speed was evaluated on a single Nvidia T4 GPU.

Method	**Published**	Parser	Pose	**FID** ↓	**LPIPS** ↓	**Runtime (ms)** ↓	**FLOPs (B)** ↓	**Memory usage (MB)** ↓
ACGPN [30]	CVPR 2020	✓	✓	33.33	0.231	153.64	399.08	565.86
PF-AFN [4]	CVPR 2021			27.33	0.216	35.80	137.85	293.25
C-VTON† [3]	CVPRW 2022	✓		37.06	0.241	66.90	108.47	168.60
SDAFN [1]	ECCV 2022		✓	30.20	0.245	83.42	149.40	150.87
FS-VTON [9]	CVPR 2022			26.48	0.200	37.49	132.98	309.25
DM-VTON	Ours			28.33	0.215	23.27	69.82	37.79

Figure 7: Qualitative comparison on VITON-Clean dataset [8].

dataset, denoted by VTION-Clean. We combine the VTION-Clean training set and our synthesized images to train our proposed method.

Fréchet Inception Distance (FID) [10] and Learned Perceptual Image Patch Similarities (LPIPS) [31] metrics were used to evaluate the similarity of try-on results to real images.

4.3 Experimental Results

We compared the performance of our proposed MD-VTON with SOTA methods in virtual try-on, such as ACGPN [30], PF-AFN [4], C-VTON [3], SDAFN [1], FS-VTON [9]. Comparison of MD-VTON against those methods in terms of image quality (i.e., FID and LPIPS), inference speed (i.e., ms), FLOPs (Floating point operations), and memory usage (MB) is shown in Table 1. Our proposed method outperforms all other SOTAs in terms of runtime, FLOPs, and memory consumption. On the other hand, our DM-VTON achieves slightly higher FID and LPIPS scores than those of PF-AFN [4] and FS-VTON [9]. The experimental results prove that the proposed DM-VTON can run in real-time (i.e., 43 frames per second) with small memory consumption but still retains high-quality virtual try-on results. The visualization of compared methods is illustrated in Fig. 7.

5 Pilot Study

We invited 12 participants who are university students and researchers in the 18-44 age range. We let the users experience our DM-VTON and collected their feedback. The experimental scenario was that while shopping online at home, the users come across a particular garment that catches their eyes, but they are unsure whether it looks good on them. We offered them an application to try on such garments before making the purchasing decision. Specifically, we prepared a collection of 20 garments taken from the VITON [8] and PolyvoreOutfits [7] datasets (see Fig. 8). These garments were carefully selected to represent a variety of colors, shapes, and textures.

Each participant took part in a 10-minute session in which the participant was asked to perform a virtual try-on using our provided model images and virtual try-on on themselves directly captured

(a) Human model samples

(b) Garment samples

Figure 8: Examples of data used in our pilot study.

from our camera. In terms of human models, we used 10 person images in the VITON [8] and DeepFashion [19] datasets (see Fig. 8).

Upon completing the trial, we interviewed participants and asked for their feedback. Our primary objective was to evaluate how our system influenced their purchasing decisions. We also gathered their feedback about the output quality and whether they prefer using the model images or their own images to enhance the overall user experience in the future.

Most of the participants agreed that trying on clothes in various poses helped them visualize the suitability of the garments before making a purchase decision. Specifically, 66.7% of the participants felt confident enough to make the purchasing decision after using our system, while the remaining 33.3% had doubts about the truthiness of the models. Moreover, 83.3% of the participants preferred using their images for try-on, as it provided a more realistic experience for them. On the other hand, the remaining 16.7% considered both options, as the provided models allowed them to see the best representation of the garments, such as with appropriate brightness and poses. Fig. 9 illustrates some virtual try-on results on our provided models. Due to privacy issues, we did not capture the virtual try-on results on the participants' images.

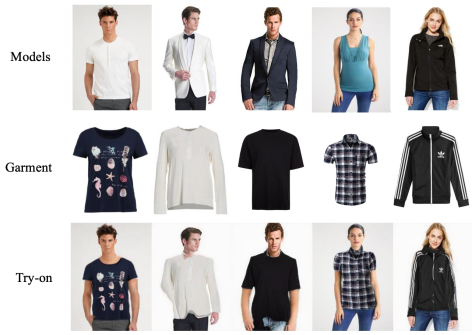


Figure 9: Results obtained when users performed virtual try-on on our provided human models in the pilot study.

We also received valuable feedback from participants on areas for improvement. In real-life conditions, the background, brightness, and quality of the captured images might not be suitable for trying on clothes, which is due to the fact that our train and test datasets only contain simple backgrounds and have proper brightness conditions. Thus, applying some pre-processing techniques such as segmentation and brightness equalization is necessary to address this issue. Additionally, participants also suggested that our system exhibited inconsistencies when dealing with complex poses such as half-turn poses or crossed-arm poses. Their feedback is useful in enhancing the user experience in the future.

6 Conclusion

This paper presents the simple yet efficient Distilled Mobile Real-time Virtual Try-On (DM-VTON) framework. By leveraging the knowledge distillation scheme, we developed a lightweight parser-free network to boost the processing speed. Our proposed network utilizes mobile-based architectures, resulting in achieving real-time virtual try-on capabilities while maintaining high-quality output and computational efficiency. In addition, the Teacher network, trained using a parser-based approach, provides supervision to the Student network, enabling it to learn without relying on the human representation of ground truth.

Additionally, to address the limited pose variation observed in the training images, we introduced the Virtual Try-on-guided Pose for Data Synthesis (VTP-DS) to enrich the diversity of poses in the training data. VTP-DS automatically identifies input images with incorrect poses generated by the framework and generates additional images for those specific poses.

Experimental results and the pilot study showcase the potential of our proposed framework. It can be applied to real-time augmented reality (AR) applications, paving the way for improved user experiences in a virtual fashion context.

References

[1] S. Bai, H. Zhou, Z. Li, C. Zhou, and H. Yang. Single stage virtual try-on via deformable attention flows. In *ECCV*, pp. 409–425, 2022.

[2] A. K. Bhunia, S. Khan, H. Cholakkal, R. M. Anwer, J. Laaksonen, M. Shah, and F. S. Khan. Person image synthesis via denoising diffusion model. In *CVPR*, pp. 5968–5976, 2023.

[3] B. Fele, A. Lampe, P. Peer, and V. Struc. C-vton: Context-driven image-based virtual try-on network. In *WACV*, pp. 3144–3153, 2022.

[4] Y. Ge, Y. Song, R. Zhang, C. Ge, W. Liu, and P. Luo. Parser-free virtual try-on via distilling appearance flows. In *CVPR*, pp. 8485–8493, 2021.

[5] R. Guler, N. Neverova, and I. DensePose. Dense human pose estimation in the wild. In *CVPR*, pp. 18–23, 2018.

[6] X. Han, X. Hu, W. Huang, and M. R. Scott. Clothflow: A flow-based model for clothed person generation. In *ICCV*, pp. 10471–10480, 2019.

[7] X. Han, Z. Wu, Y.-G. Jiang, and L. S. Davis. Learning fashion compatibility with bidirectional lstms. In *ACM Multimedia*, pp. 1078–1086, 2017.

[8] X. Han, Z. Wu, Z. Wu, R. Yu, and L. S. Davis. Viton: An image-based virtual try-on network. In *CVPR*, pp. 7543–7552, 2018.

[9] S. He, Y.-Z. Song, and T. Xiang. Style-based global appearance flow for virtual try-on. In *CVPR*, pp. 3470–3479, 2022.

[10] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium. *NeurIPS*, 30, 2017.

[11] G. Hinton, O. Vinyals, and J. Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[12] T. Issenhuth, J. Mary, and C. Calauzenes. Do not mask what you do not need to mask: a parser-free virtual try-on. In *ECCV*, pp. 619–635, 2020.

[13] J. Johnson, A. Alahi, and L. Fei-Fei. Perceptual losses for real-time style transfer and super-resolution. In *ECCV*, pp. 694–711, 2016.

[14] T. Karras, S. Laine, and T. Aila. A style-based generator architecture for generative adversarial networks. In *CVPR*, pp. 4401–4410, 2019.

[15] G. Kuppa, A. Jong, X. Liu, Z. Liu, and T.-S. Moh. Shineon: Illuminating design choices for practical video-based virtual clothing try-on. In *WACV*, pp. 191–200, 2021.

[16] C. Lin, Z. Li, S. Zhou, S. Hu, J. Zhang, L. Luo, J. Zhang, L. li Huang, and Y. He. Rmgn: A regional mask guided network for parser-free virtual try-on. In *IJCAI*, pp. 1151–1158, 2022.

[17] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, and S. Belongie. Feature pyramid networks for object detection. In *CVPR*, pp. 2117–2125, 2017.

[18] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *ECCV*, pp. 740–755, 2014.

[19] Z. Liu, P. Luo, S. Qiu, X. Wang, and X. Tang. Deepfashion: Powering robust clothes recognition and retrieval with rich annotations. In *CVPR*, pp. 1096–1104, 2016.

[20] D. Morelli, M. Fincato, M. Cornia, F. Landi, F. Cesari, and R. Cucchiara. Dress code: High-resolution multi-category virtual try-on. In *CVPR*, pp. 2231–2235, 2022.

[21] T. Park, M.-Y. Liu, T.-C. Wang, and J.-Y. Zhu. Semantic image synthesis with spatially-adaptive normalization. In *CVPR*, pp. 2337–2346, 2019.

[22] O. Ronneberger, P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pp. 234–241, 2015.

[23] M. Sandler, A. Howard, M. Zhu, A. Zhmoginov, and L.-C. Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, pp. 4510–4520, 2018.

[24] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014.

[25] W. Song, Y. Gong, and Y. Wang. Vtonshoes: Virtual try-on of shoes in augmented reality on a mobile device. In *ISMAR*, pp. 234–242, 2022.

[26] D. Sun, S. Roth, and M. J. Black. A quantitative analysis of current practices in optical flow estimation and the principles behind them. *IJCV*, 106:115–137, 2014.

[27] B. Wang, H. Zheng, X. Liang, Y. Chen, L. Lin, and M. Yang. Toward characteristic-preserving image-based virtual try-on network. In *ECCV*, pp. 589–604, 2018.

[28] C.-Y. Wang, A. Bochkovskiy, and H.-Y. M. Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. In *CVPR*, pp. 7464–7475, 2023.

[29] Y. Xu, S. Yang, W. Sun, L. Tan, K. Li, and H. Zhou. 3d virtual garment modeling from rgb images. In *ISMAR*, pp. 37–45, 2019.

[30] H. Yang, R. Zhang, X. Guo, W. Liu, W. Zuo, and P. Luo. Towards photo-realistic virtual try-on by adaptively generating-preserving image content. In *CVPR*, pp. 7850–7859, 2020.

[31] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, pp. 586–595, 2018.

[32] X. Zhong, Z. Wu, T. Tan, G. Lin, and Q. Wu. Mv-ton: Memory-based video virtual try-on network. In *ACM Multimedia*, pp. 908–916, 2021.

Multilingual Communication System with Deaf Individuals Utilizing Natural and Visual Languages

Tuan-Luc Huynh[†1,2], Khoi-Nguyen Nguyen-Ngoc[†1,2], Chi-Bien Chu[†1,2], Minh-Triet Tran[1,2], and Trung-Nghia Le[‡1,2]

[1]Faculty of Information Technology, University of Science, VNU-HCMC, Vietnam
[2]Vietnam National University, Ho Chi Minh City, Vietnam

***Abstract*—According to the World Federation of the Deaf, more than two hundred sign languages exist. Therefore, it is challenging to understand deaf individuals, even proficient sign language users, resulting in a barrier between the deaf community and the rest of society. To bridge this language barrier, we propose a novel multilingual communication system, namely MUGCAT, to improve the communication efficiency of sign language users. By converting recognized specific hand gestures into expressive pictures, which is universal usage and language independence, our MUGCAT system significantly helps deaf people convey their thoughts. To overcome the limitation of sign language usage, which is mostly impossible to translate into complete sentences for ordinary people, we propose to reconstruct meaningful sentences from the incomplete translation of sign language. We also measure the semantic similarity of generated sentences with fragmented recognized hand gestures to keep the original meaning. Experimental results show that the proposed system can work in a real-time manner and synthesize exquisite stunning illustrations and meaningful sentences from a few hand gestures of sign language. This proves that our MUGCAT has promising potential in assisting deaf communication.**

***Index Terms*—Sign Language Recognition, Text-to-Image Synthesis, Image Captioning**

I. Introduction

Communication with deaf people is mainly based on sign language, a combination of hand gestures, facial expressions, and postures to convey semantic information. However, these visual communication systems are difficult to learn and remember, leading to barriers between the deaf community and the rest of society; this problem has not been fully solved until now.

Although technologies have been developed to understand the behaviors of deaf people, such as sign language translation via cameras and sensory gloves, they still have several issues. Sign language translation via camera systems [1], [2] needs a fixed camera and a simple background to recognize sign gestures accurately. Besides that, translating sign languages to human-understandable languages often leads to unnatural results. It causes difficulties in jobs that require smoothness in words, such as explaining new concepts or telling stories. The birth of sensory devices like smart gloves [3], [4] is a big step forward in this field. However, it still does not solve the problem of unnatural translations. In addition, the more modern sensors will come with high prices, which makes it difficult to reach the deaf community.

Combined with natural language, which can be expressed as text or voice, visual language can reform the communication between ordinary people and deaf/dumb people. Indeed, visual cues (*e.g.*, images, videos, 3D models) are the best aid to express new concepts intuitively. Visual cues play an important role in deaf communication, especially in literacy education for deaf children. Visual communication efficiently bridges ordinary people with the deaf community, regardless of different nationalities or different languages.

To assist communication with deaf individuals, we propose a **MU**ltilin**G**ual **C**ommunic**AT**ion system (MUGCAT). Inspired by the adage "A picture is worth a thousand words," our system supports diverse cues, such as sign languages, natural languages, and visual languages, to help deaf people express their thoughts more clearly. The proposed MUGCAT system consists of two main phases: converting sign language to intermediate language, which ordinary people can understand, and enriching the translated information by reconstructing a meaningful sentence aided by illustrations. We first recognize and translate these hand gestures of deaf individuals (*i.e.*, sign language) into human-understandable language (*i.e.*, textual words or phases). Illustrations are then synthesized via a text-to-image model for visual communication. By transforming sign languages into pictures - universal mediums of expressiveness - our system significantly help deaf individuals convey their thoughts. Due to the limitation of sign languages, it is challenging to translate hand gestures to complete meaningful sentences for ordinary people. Therefore, we propose using an image captioning method to assist the incompleted translated text. Furthermore, our MUGCAT

system can measure the semantic similarity of generated image captions with the intermediate translated text to keep the original meaning of the sign language. In this way, our MUGCAT system can help to express the intentions of deaf communicators more intuitively and clearly.

Experimental results on the WLASL dataset [5] show the potential of our MUGCAT system in assisting natural communication with deaf individuals. The proposed system can recognize sign gestures with an accuracy of 46.8% in real time. In addition, meaning sentences for humans are generated with corresponding exquisite, beautiful, and stunning illustrations. We expect our MUGCAT system to benefit both the deaf community and the sign language research community.

Our main contributions are summarized as follows:

- We propose a novel system, namely MUGCAT, to support multilingual communication for deaf individuals. Our simple yet efficient system utilizes both natural and visual languages to enhance the interpretation of deaf communicators.
- Our MUGCAT system accurately recognizes and translates sign languages to human-understandable text. The proposed system also can transform the translated text into illustrative and expressive images in real-time performance.
- The synthesized images might be misleading; hence, we propose to use an image captioning model to select the image that best fit the translated text, further improving the efficiency of MUGCAT.

II. Related Work

A. Deaf Communication

Communicating with deaf individuals mainly occurs through auditory (*e.g.*, lip reading) and visual (*e.g.*, sign language) modes. However, sign language is more popular than lip reading because understanding speech by visually interpreting the movements of the lips, face, and tongue is extremely challenging, even for deaf people.

With the development of modern technology, many technological devices have been invented to translate sign language into text, speech, etc. Some special sensors were made to detect hand movements. The translation glove products (EnableTalk [3] and SignAloud [4]) work on the integration of sensors that attach to the finger to record hand posture and movements and then convert sensor signals into speech through an independent processing unit. However, these products are difficult to access widely due to their high cost and difficulty to use in daily life.

B. Sign Language Recognition

Recently, many computer vision algorithms have been proposed to recognize sign language from video only, thus avoiding the dependence on costly sensor devices. Given a video, besides RGB frames, we can also obtain other modalities of input such as image depth [6], [7] and optical flow [8], [9] (pixel-wise motions between consecutive video frames). For RGB input only, 3D ConvNets were widely applied [10], [11] to extract spatial-temporal information from videos. Lin *et al*. [12] inserted a Temporal Shift Module into 2D ConvNets to get an accuracy commensurate with 3D ConvNets while keeping the complexity of 2D ConvNets. Komkov *et al*. [13] combined the learned knowledge from multiple single-modality models with mutual learning technique [14] to obtain the best model on each input modality.

C. Text-to-Image

In the last couple of years, text-to-image models [15] have attracted big tech companies' attention and thus have received rapid and massive improvements. Classifier-free guided diffusion models have recently been shown to be highly effective at high-resolution image generation, and they have been widely used in large-scale diffusion frameworks, including GLIDE [16], DALL-E 2 [17], and Imagen [18]. Nevertheless, the latest development of diffusion model-based text-to-image model, namely Stable Diffusion [19], has been the most significant impact since its release. Stable Diffusion offers excellent image quality while significantly lowering the computation cost. What makes Stable Diffusion exceptionally attractive compared to other competitors due to its open-source. On the other hand, Google and OpenAI do not intend to open-source Imagen [20] and DALL-E 2 [17], respectively.

While the artificial intelligence community has dominantly used text-to-image models to create beautiful artworks, there is little attention on using these models on real-world problems. In this work, we customized Stable Diffusion to generate meaningful and expressive images from the translated sign language text in a real-time manner to help visualize conversation with or between sign language users.

D. Image Captioning

Research on image captioning in recent years generally uses the encoder-decoder architecture. The encoder extracts the visual information from images for the decoder, which generates an acceptable description. In the early, the encoder was a CNN backbone [21], [22]. Later, it was replaced by an object detector such as Faster R-CNN to extract object-level features [23]. This proved more efficient and improved performance because the object information and their relationships are very useful in describing an image. However, due to the high computational cost of the object detection model, it is hard to apply in a problem that requires high speed, such as communication. Besides that, Transformer applications in the encoder to extract features or the decoder for caption generating task [24], [25] also demonstrated surprising efficiency improvements.

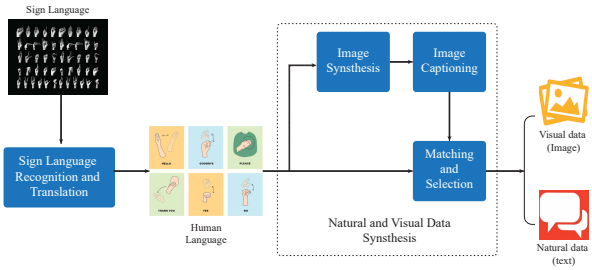


Fig. 1. Pipeline of our multilingual communication (MUGCAT) system

In this study, with the aim of balancing accuracy and efficiency, we used a recent state-of-the-art method [26] which proposed a Transformer-only neural architecture utilizing dual visual features to improve performance and increase speed.

III. Proposed System

A. Overview

Figure 1 illustrates the pipeline of our proposed multilingual communication (MUGCAT) system, which consists of two main components: Sign language recognition and translation (SLRT), Natural and visual data synthesis. First, words obtained from SLRT are illustrated by the text-to-image module resulting in several images. Then, image captioning is carried out to achieve complete descriptions of all synthesized images. Finally, with each image and its description, we compare its semantic similarity with the translated keywords from SLRT to choose the most suitable image and description. Unlike conventional SLRT systems that need to correctly recognize the whole sentence to express the meaning, our system generates a suggested image and complete description to represent the keywords made in sign language to overcome the disadvantage of missed recognized sign language.

B. Sign Language Recognition and Translation

Sign language recognition aims to predict the sequence of signs performed in a video, while sign language translation further translates the signs into spoken/written languages. To synthesize images that fully convey the meaning of a signer, we only need to identify some keywords from the hand gesture sequence. Therefore, the recognition task is more suitable for our MUGCAT system because it is simple but still responsive to the system. Due to the lack of suitable datasets in this domain, we treat the problem as an action recognition task, where the objective is to identify single words from short clips. This simplifies the problem and meets our requirement for indicating visual keywords.

In this work, we employed and compared several action recognition methods [10], [12], [13] on WLASL dataset [5], the largest video dataset of word-level American sign language (ASL). The main ideas of employed methods are summarized as follows:

Two-Stream Inflated 3D ConvNets (I3D) [10] combines two 3D ConvNets (one for RGB image stream, one for optical-flow stream) to both take advantage of pre-trained ImageNet weights and force the model to learn motion features directly. Two 3D ConvNets are trained separately, and their predictions are averaged at test time.

Temporal Shift Module (TSM) [12] inserted TSM module into 2D ConvNets to capture temporal relationships between video frames. Feature maps are shifted along the temporal dimension to maintain 2D ConvNet's complexity while achieving the performance of 3D ConvNets.

Mutual Modality Learning (MML) [13] ensembled knowledge from single-modality models into a single model to obtain the best single-modality model for each modality. The algorithm can be summarized in three steps: train two separate networks A_1, A_2 on the RGB modality; respectively initialize two networks B_1, B_2 with the weights of A_1, A_2, then train B_1, B_2 together using mutual learning technique on RGB modality; from B_1's weights, initialize N models C_1, C_2, ..., C_N corresponding to N different modalities (RGB, optical flow, and depth), then train these N models together using mutual learning.

C. Natural and Visual Data Synthesis

1) Text-To-Image Synthesis: Stable Diffusion [19], a state-of-the-art diffusion-based text-to-image model, is the core component of our system, which strives to actualize the adage "A picture is worth a thousand words." This method can offer excellent image quality while significantly lowering computation costs.

However, the sequential sampling process of diffusion-based models is time-consuming. As a result, the text-to-image module is also the bottleneck of our system. To overcome this limitation, we customized hyperparameters of Stable Diffusion to retain high-quality images while significantly reducing the sampling process time.

Another issue that affects our system performance is the relevancy of synthesized images. Prompt engineering (*i.e.*, prompt modifiers) is necessary for guiding the text-to-image models to generate superior-quality art. However, in our proposed system, the prompt text for Stable Diffusion is limited keywords from the SLRT module. Therefore, it is unavoidable that the prompt's quality is limited, which leads to potential drops in generated image relevancy. We addressed this issue by introducing the image captioning model in the system's next stage, which serves as a filter to select the most relevant image.

2) Image Captioning: Image captioning methods are classified into two main approaches: grid features and region features. Methods based on grid features directly extract object features from high-layer feature maps of

the whole image. Thus, generated captions can contain information about the whole image. Meanwhile, methods based on region features [23] rely on detecting objects in the image and then extracting local features of image regions to infer results. However, detected objects cannot represent the overall context of the image nor the relationships of objects that affect generated captions.

In this work, we used Grid- and Region-based Image captioning Transformer (GRIT) [26], a state-of-the-art image captioning method, which uses both types of mentioned features to enhance both contextual information and object-level information. Grid features are extracted using a standard self-attention Transformer, and region features are extracted by Deformable DETR detector [27]. Then, the extracted features are fed to a caption generator based on Transformer to generate the final caption. In this step, we employed Parallel Cross-Attention [26] to relate between dual visual features and caption words.

3) Matching and Selection: SLRT generates incomplete keywords; thus, the images synthesized from the previous step inevitably are not completely consistent with each other and the communicator's expression. Therefore, we propose an extra step of matching and selecting the caption whose meaning is closest to the input keywords. From there, our system is able to recover the complete sentence that the communicator wanted to express from just the discrete words.

Concretely, given K results $\{I_1, I_2, \ldots, I_K\}$ of the previous step, the goal of the matching and selection is to find the most suitable pair of image $\widehat{I}$ and its caption $q_{\widehat{I}}$. For each caption sentence, we measure its semantic similarity with keywords obtained from SLRT. We used Sentence Transformers [28], denoted by $\psi(\cdot)$, to compute sentence embeddings and evaluate them with cosine similarity. Mathematically, it performs a maximization expressed as:

$$\{\widehat{I}, q_{\widehat{I}}\} = \underset{i \in \{1,2,\ldots,K\}}{\operatorname{argmax}} D(\psi(q_{I_i}), \psi(Q)), \qquad (1)$$

where Q is a sentence that includes keywords received from SLRT, $D(\cdot)$ is a cosine similarity function.

IV. Experiments

In this section, we elaborate on the extensive experiments conducted on our proposed system. All experiments were tested on a machine with a single Nvidia V100 GPU.

A. Sign Language Recognition

As shown in Table IV-A, we compared several action classification methods [10], [12], [13] on the WLASL [5] test set. In detail, for TSM [12], we trained a model from scratch and fine-tuned another model, which was pre-trained on the Kinetic [10] dataset. For MML [13], we trained a model from scratch with only RGB input as WLASL [5] dataset does not provide optical flow or depth annotations. All the networks above use an ImageNet-pre-trained ResNet50 as the backbone. Lastly, we reused two public I3D [10] with different pretraining datasets [5], [29] for our experiment.

TABLE I
Accurary and efficiency on the WLASL [5] test set. All the compared methods utilize a pre-trained backbone on ImageNet, and then they were finetuned on the WLASL dataset. The FPS was measured on a Nvidia V100 GPU.

Method	**Pretraining Dataset**	**Accuracy (%)**	**FPS** (infer only)	**FPS** (infer & load data)
I3D [10]	BSL1K [29]	**46.8**	**1429**	95
	Kinetic [10]	32.5		
TSM [12]	✗	20.8	357	60
	Kinetic [10]	13.9		
MML [13]	✗	20.8	323	**104**

We first evaluated top-1 accuracy on the WLASL [5] test set. The main challenge of this dataset is the number of words to classify up to 2,000, while the number of videos in the training set is just over 14,000. Therefore compared methods only achieve acceptable accuracy. I3D achieved the top-1 accuracy of 46.8%, which is also state-of-the-art top-1 accuracy on the WLASL test set.

We then evaluated the efficiency of methods by measuring the execution time and the number of processed frames on the whole test set to obtain the average FPS. All methods can run in a real-time manner. Especially, MML [13] can achieve 104 FPS counting all initialization steps, such as loading the model, preparing the dataset, etc. We also tried to compute FPS in the practice scenario, where SLRT methods directly process the video stream and ignore the initialization steps. All methods can achieve more than 300 FPS, and I3D [10] achieved a surprising speed of 1429 FPS. The results show that these models have the potential to be deployed on mobile devices and embedded systems while still achieving real-time speed.

B. Stable Diffusion Hyperparameters Adjustment

The default settings of Stable Diffusion [19] hardly achieve near real-time performance. This section discusses our extensive experiments in various settings to discover the optimal trade-off point between execution time and image quality. Specifically, we focus on the number of sampling steps, the desired resolution, and the number of samples. We used the public checkpoint *sd-v1-4.ckpt* in our experiments.

The number of sampling steps is the most crucial hyperparameter that directly controls the quality of generated images and positively correlates with the execution duration. The default hyperparameter of 50 sampling steps using PLMS sampler [30] generates high-quality images. Since our system ideally should work in real-time performance, we figured that 20 sampling steps could speed up 2.4 times while having subtle drops

TABLE II
Performance of Stable Diffusion on a single Nvidia V100 GPU. The prompt is "A beautiful flower garden on a sunny day with a valley background." Resolution is 512×512. The FID score [31] was calculated using 50 sampling steps as the real distribution, 128 images per distribution. The optimal hyperparameter is 20 sampling steps, which can keep the image quality but speed up 2.4 times.

Sampling steps	FID Score ↓	Seconds per Batch ↓
50	0	35.50
45	33.43	32.05
40	30.44	28.66
35	31.70	25.24
30	31.55	21.79
25	33.19	18.39
20	33.51	14.97
15	40.33	12.25

Fig. 2. The renowned "a photograph of an astronaut riding a horse." The top and down rows are 20 and 50 sampling steps, respectively.

in contextual information, as demonstrated in Fig. 2. Indeed, Table IV-B shows that setting the sampling steps to 20 is optimal with the FID of 33.5, approximately equivalent to higher sampling steps but can process a batch of 128 images in only 15 seconds. Going lower than 20 sampling steps results in a surge of FID scores.

Image resolution is another element that significantly affects Stable Diffusion's running time. Following tips of Suraj *et al*. [32], we tried reducing the resolution and came up with seven different resolutions in decreasing execution time, namely 512×512, 512×448, 448×448, 512×384, 448×384, 512×320, and 384×384. As illustrated in Fig. 4, the first two images on the top row have a valley background. As the resolution decreases, contextual information in the prompt will gradually become less constrained.

Fully utilizing GPU's capability is another technique to enhance the performance of the model. We tried setting the largest batch size on each respective resolution and recorded the run time accordingly. Figure 3 illustrates the benchmark result of the highest resolution, lowest resolution, and median one. The experimental result shows that the optimal number of synthesis images (*i.e.*, K in Eq. 1) is either 8 or 16. The reason is twofold: Firstly, a small batch size can easily fit into a conventional GPU; Secondly, since the image captioning

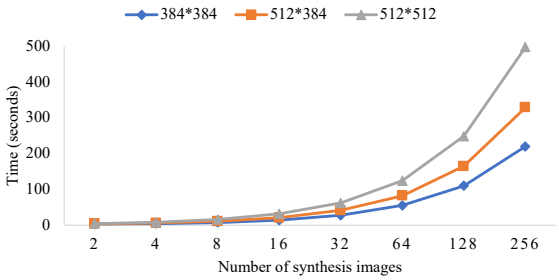


Fig. 3. Stable Diffusion's execution time in different resolutions using a single Nvidia V100 GPU. The batch size is maximized for each respective resolution, and the hyperparameter of sampling steps is 20. The optimal numbers of synthesis images are from 8 to 16.

Fig. 4. Images generated by Stable Diffusion using the same prompt "A beautiful flower garden on a sunny day with a valley background" in decreasing resolution order (from left to right, top to bottom).

module's execution time scales linearly with the number of generated images, selecting a small batch size can thus improve both modules' performance.

C. Image Captioning Visualization

We employed GRIT [26] model that uses the pre-trained of object detector on four datasets: COCO [33], Visual Genome, Open Images [34], Object365 [35], and applied Parallel Cross-Attention [26] for image captioning. We can achieve the per-batch inference time of about 0.75s when setting the batch size of 16 and 0.87s with the batch size of 8 on a single Nvidia V100 GPU.

Example results are visualized in Fig. 5. With the developed text refinement mechanism, our MUGCAT system obviously generates an illustration and complete caption with high semantic similarity with the original sentence from the keywords, as shown in Fig.5.

V. Conclusion

We have proposed a **MU**ltilin**G**ual **C**ommunic**AT**ion system (MUGCAT), which integrates sign language recognition and translation, text-to-image, and image captioning methods. The proposed system harmonizes three different methodologies to help overcome the difficulty of communicating with deaf individuals. Leveraging the latest development in text-to-image synthesis and image captioning to transform written text into visual images, we strive to lift the language barrier that has always existed in the sign language community.

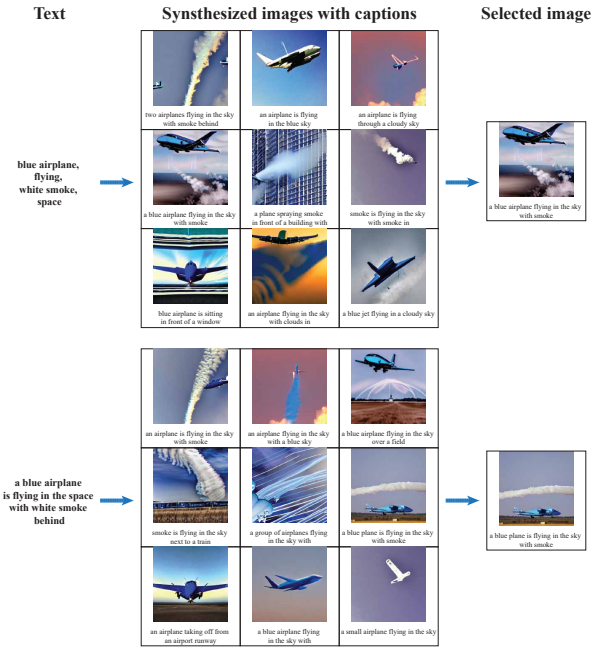


Fig. 5. Example of text refinement on a complete sentence (below) and only on keywords (above) giving similar results.

Experiments show the potential of our proposed system in practice. In the future, it would be interesting to modify our system's camera to first-person. We want to explore the possibility of sign language recognition and translation methods from a first-person perspective since this would overcome the problem of requiring standing in front of a fixed camera.

Acknowledgment. This research is funded by University of Science, VNU-HCM, under grant number CNTT 2022-15.

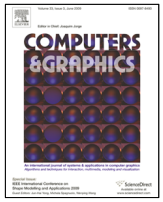
REFERENCES

[1] A. Hao, Y. Min, and X. Chen, "Self-mutual distillation learning for continuous sign language recognition," in *ICCV*, 2021.
[2] Y. Min, A. Hao, X. Chai, and X. Chen, "Visual alignment constraint for continuous sign language recognition," in *ICCV*, 2021, pp. 11 542–11 551.
[3] F. Lardinois, "Ukrainian students develop gloves that translate sign language into speech," https://techcrunch.com/2012/07/09/enable-talk-imagine-cup, 2012.
[4] "UW undergraduate team wins $10,000 lemelson-mit student prize for gloves that translate sign language," https://www.washington.edu/news/2016/04/12/uw-undergraduate-team-wins-10000-lemelson-mit-student-prize-for-gloves-that-translate-sign-language, 2016.
[5] D. Li, C. Rodriguez, X. Yu, and H. Li, "Word-level deep sign language recognition from video: A new large-scale dataset and methods comparison," in *WACV*, 2020, pp. 1459–1469.
[6] A. Gurram, A. F. Tuna, F. Shen, O. Urfalioglu, and A. M. López, "Monocular depth estimation through virtual-world supervision and real-world sfm self-supervision," *IEEE T-PAMI*, 2021.
[7] J. Wang, Y. Zhong, Y. Dai, S. Birchfield, K. Zhang, N. Smolyanskiy, and H. Li, "Deep two-view structure-from-motion revisited," *CVPR*, 2021.
[8] Z. Huang, X. Shi, C. Zhang, Q. Wang, K. C. Cheung, H. Qin, J. Dai, and H. Li, "FlowFormer: A transformer architecture for optical flow," *ECCV*, 2022.
[9] S. Bai, Z. Geng, Y. Savani, and J. Z. Kolter, "Deep equilibrium optical flow estimation," in *CVPR*, 2022.
[10] J. Carreira and A. Zisserman, "Quo vadis, action recognition? a new model and the kinetics dataset," in *CVPR*, 2017.
[11] S. Ji, W. Xu, M. Yang, and K. Yu, "3d convolutional neural networks for human action recognition," *IEEE T-PAMI*, vol. 35, no. 1, pp. 221–231, 2013.
[12] J. Lin, C. Gan, and S. Han, "Tsm: Temporal shift module for efficient video understanding," in *ICCV*, 2019.
[13] S. Komkov, M. Dzabraev, and A. Petiushko, "Mutual modality learning for video action classification," *arXiv preprint arXiv:2011.02543*, 2020.
[14] Y. Zhang, T. Xiang, T. M. Hospedales, and H. Lu, "Deep mutual learning," in *CVPR*, 2018, pp. 4320–4328.
[15] S. S. Baraheem, T.-N. Le, and T. V. Nguyen, "Text-to-image synthesis via aesthetic layout," in *ACM MM*, 2020.
[16] A. Nichol, P. Dhariwal, A. Ramesh, P. Shyam, P. Mishkin, B. McGrew, I. Sutskever, and M. Chen, "Glide: Towards photorealistic image generation and editing with text-guided diffusion models," *arXiv preprint arXiv:2112.10741*, 2021.
[17] A. Ramesh, P. Dhariwal, A. Nichol, C. Chu, and M. Chen, "Hierarchical text-conditional image generation with clip latents," *arXiv preprint arXiv:2204.06125*, 2022.
[18] C. Meng, R. Gao, D. P. Kingma, S. Ermon, J. Ho, and T. Salimans, "On distillation of guided diffusion models," *arXiv preprint arXiv:2210.03142*, 2022.
[19] R. Rombach, A. Blattmann, D. Lorenz, P. Esser, and B. Ommer, "High-resolution image synthesis with latent diffusion models," in *CVPR*, 2022.
[20] Google, "Imagen," https://imagen.research.google/, 2022.
[21] K. Xu, J. Ba, R. Kiros, K. Cho, A. Courville, R. Salakhudinov, R. Zemel, and Y. Bengio, "Show, attend and tell: Neural image caption generation with visual attention," in *ICML*, 2015.
[22] S. J. Rennie, E. Marcheret, Y. Mroueh, J. Ross, and V. Goel, "Self-critical sequence training for image captioning," in *CVPR*, 2017.
[23] P. Anderson, X. He, C. Buehler, D. Teney, M. Johnson, S. Gould, and L. Zhang, "Bottom-up and top-down attention for image captioning and visual question answering," in *CVPR*, 2018.
[24] G. Li, L. Zhu, P. Liu, and Y. Yang, "Entangled transformer for image captioning," in *ICCV*, 2019.
[25] Y. Pan, T. Yao, Y. Li, and T. Mei, "X-linear attention networks for image captioning," in *CVPR*, 2020.
[26] V.-Q. Nguyen, M. Suganuma, and T. Okatani, "Grit: Faster and better image captioning transformer using dual visual features," *ArXiv preprint arXiv:2207.09666*, 2022.
[27] X. Zhu, W. Su, L. Lu, B. Li, X. Wang, and J. Dai, "Deformable {detr}: Deformable transformers for end-to-end object detection," in *ICLR*, 2021.
[28] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," *ArXiv preprint arXiv:1908.10084*, 2019.
[29] S. Albanie, G. Varol, L. Momeni, T. Afouras, J. S. Chung, N. Fox, and A. Zisserman, "BSL-1K: Scaling up co-articulated sign language recognition using mouthing cues," in *ECCV*, 2020.
[30] L. Liu, Y. Ren, Z. Lin, and Z. Zhao, "Pseudo numerical methods for diffusion models on manifolds," in *ICLR*, 2022.
[31] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter, "Gans trained by a two time-scale update rule converge to a local nash equilibrium," *NeurIPS*, vol. 30, 2017.
[32] S. Patil, P. Cuenca, N. Lambert, and P. von Platen, "Stable diffusion with diffusers," https://huggingface.co/blog/stable_diffusion.
[33] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick, "Microsoft coco: Common objects in context," in *ECCV*, 2014, pp. 740–755.
[34] A. Kuznetsova, H. Rom, N. Alldrin, J. Uijlings, I. Krasin, J. Pont-Tuset, S. Kamali, S. Popov, M. Malloci, A. Kolesnikov, T. Duerig, and V. Ferrari, "The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale," *IJCV*, 2020.
[35] S. Shao, Z. Li, T. Zhang, C. Peng, G. Yu, X. Zhang, J. Li, and J. Sun, "Objects365: A large-scale, high-quality dataset for object detection," in *ICCV*, 2019.

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag



SHREC'22 Track: Sketch-Based 3D Shape Retrieval in the Wild

Jie Qin[a,1,*], Shuaihang Yuan[b,1], Jiaxin Chen[c,1], Boulbaba Ben Amor[d,e,1], Yi Fang[b,f,1], Nhat Hoang-Xuan[g,i,2], Chi-Bien Chu[g,i,2], Khoi-Nguyen Nguyen-Ngoc[g,i,2], Thien-Tri Cao[g,i,2], Nhat-Khang Ngo[g,i,2], Tuan-Luc Huynh[g,i,2], Hai-Dang Nguyen[g,i,2], Minh-Triet Tran[g,h,i,2], Haoyang Luo[j,2], Jianning Wang[j,2], Zheng Zhang[j,2], Zihao Xin[k,2], Yang Wang[k,2], Feng Wang[k,2], Ying Tang[k,2], Haiqin Chen[k,2], Yan Wang[k,2], Qunying Zhou[k,2], Ji Zhang[k,2], Hongyuan Wang[k,2,*]

[a] *Nanjing University of Aeronautics and Astronautics, Nanjing, China*
[b] *New York University, New York, USA*
[c] *Beihang University, Beijing, China*
[d] *IMT Nord Europe, Lille, France*
[e] *Inception Institute of Artificial Intelligence, Abu Dhabi, UAE*
[f] *New York University Abu Dhabi, Abu Dhabi, UAE*
[g] *University of Science, VNUHCM, Ho Chi Minh City, Vietnam*
[h] *John von Neumann Institue, VNUHCM, Ho Chi Minh City, Vietnam*
[i] *Viet Nam National University, Ho Chi Minh City, Vietnam*
[j] *Harbin Institute of Technology, Shenzhen, China*
[k] *Changzhou University, Changzhou, China*

ARTICLE INFO

Article history:
Received July 12, 2022

Keywords: Sketch-based 3D Shape Retrieval, Cross-modality Retrieval, Shape Retrieval in the Wild, Point Cloud Classification

ABSTRACT

Sketch-based 3D shape retrieval (SBSR) is an important yet challenging task, which has drawn more and more attention in recent years. Existing approaches address the problem in a restricted setting, without appropriately simulating real application scenarios. To mimic the realistic setting, in this track, we adopt large-scale sketches drawn by amateurs of different levels of drawing skills, as well as a variety of 3D shapes including not only CAD models but also models scanned from real objects. We define two SBSR tasks and construct two benchmarks consisting of more than 46,000 CAD models, 1,700 realistic models, and 145,000 sketches in total. Four teams participated in this track and submitted 15 runs for the two tasks, evaluated by 7 commonly-adopted metrics. We hope that, the benchmarks, the comparative results, and the open-sourced evaluation code will foster future research in this direction among the 3D object retrieval community.

1. Introduction

Sketch-based 3D shape retrieval (SBSR) [1, 2, 3] has drawn a significant amount of attention, owing to the succinctness of free-hand sketches and the increasing demands from real applications. It is an intuitive yet challenging task due to the large discrepancy between the 2D and 3D modalities.

To foster the research on this important problem, several tracks focusing on related tasks have been held in the past SHREC challenges, such as [4, 5, 6, 7]. However, the datasets they adopted are not quite realistic, and thus cannot well simulate real application scenarios. To mimic the real-world scenario, the dataset is expected to meet the following requirements. First, there should exist a large domain gap between the two modalities, *i.e.*, sketches and 3D shapes. However, current datasets unintentionally narrow this gap by using projection-based/multi-view representations for 3D shapes (*i.e.*, a 3D

*Corresponding author:
e-mail: jie.qin@nuaa.edu.cn (Jie Qin)
[1] Track organizers.
[2] Track participants.

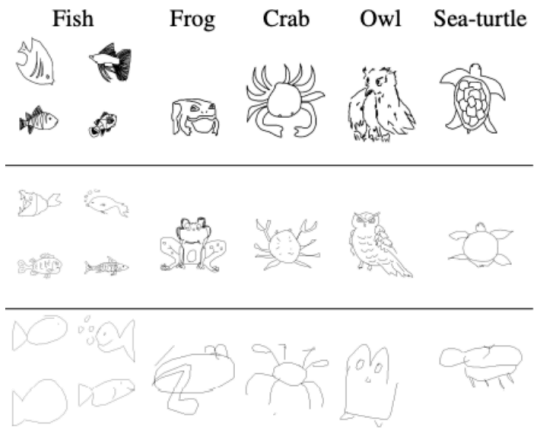


Fig. 1. Comparisons of some sketch samples between different benchmarks (images are from [8]). The first, second, and third rows depict the sketches from Sketchy [9], TU-Berlin [10], and QuickDraw [11], respectively.

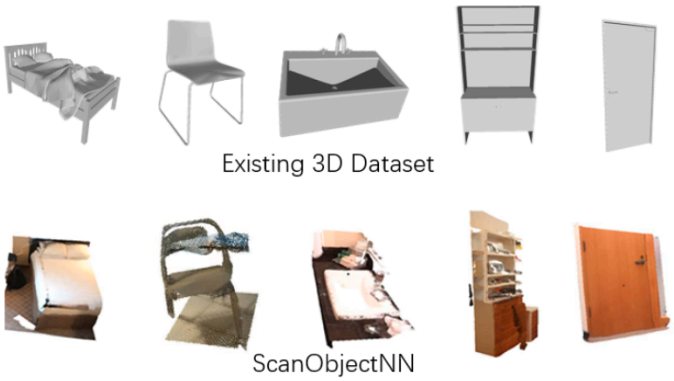


Fig. 2. Comparisons of synthetic 3D CAD models from existing datasets with 3D models from ScanObjectNN [12] scanned in realistic scenarios.

shape is manually rendered into a set of 2D images). In this way, the large 2D-3D domain discrepancy is unnecessarily reduced to the 2D-2D one. Second, the data themselves from both modalities should be realistic, mimicking the real-world scenario. More specifically, we need a full variety of sketches per category as real users possess various drawing skills. As for 3D shapes, we need to frame 3D models with real-world settings more than create them artificially. However, human sketches on existing datasets tend to be semi-photorealistic drawn by experts and the number of sketches per category is quite limited; in the meantime, most current 3D datasets used in SBSR are composed of CAD models, losing certain details compared to 3D models scanned from real objects.

To circumvent the above limitations, this track proposes a more realistic and challenging setting for SBSR. On the one hand, we adopt highly abstract 2D sketches drawn by amateurs, and at the same time, bypass the projection-based representations for 3D shapes by directly adopting and representing 3D point cloud data. On the other hand, we adopt a full variety of free-hand sketches with various samples per category, as well as a collection of realistic point cloud data framed from indoor objects. Therefore, we name this track 'sketch-based 3D shape retrieval in the wild' (SBSRW). As stated above, the term 'in the wild' is reflected in two perspectives: 1) The domain gap between the two modalities is realistic as we adopt sketches of high abstraction levels and 3D point cloud data. 2) The data themselves mimic the real-world setting as we adopt a full variety of sketches and 3D point clouds captured from real objects.

2. Benchmark Overview

To fulfill our goal of SBSR in the wild, our benchmark takes advantage of four existing 2D/3D datasets, including a large-scale 2D sketch collection, *i.e.*, QuickDraw [11], and three 3D point cloud datasets, *i.e.*, ModelNet40 [13], ShapeNet [14], and ScanObjectNN [12].

QuickDraw [11] is a million-scale sketch dataset obtained via the online game "QuickDraw" created by Google. It contains over 50 million sketches belonging to 345 categories, collected from non-expert drawers around the world, with large variability derived from human abstraction. The large domain gap between non-expert drawers and photos is reflected on this dataset, which is not considered in previous sketch benchmarks. Some comparisons of the sketch samples between different benchmarks (including Sketchy [9] and TU-Berlin [10]) are shown in Figure 1.

ModelNet40 [13] is one of the most widely-used benchmark datasets in 3D shape analysis. The organizers of this dataset perform statistics analysis on the SUN dataset [15] to find the most popular categories. Once those categories are determined, the corresponding CAD 3D shapes are collected by online searching. The collected shapes are then verified by human labors to ensure their correctness. As a result, 12,311 high-resolution 3D shapes are collected in the mesh format, and all 3D shapes are categorized into 40 classes.

ShapeNet Core55 [14] is a subset of ShapeNet. It is a rich-annotated but challenging dataset for 3D shape analysis, especially for 3D shape retrieval and 3D part segmentation. There are 55 classes, among which around 57,000 3D meshes are distributed. 35,764, 5,133, and 10,265 3D shapes are selected for training, validation, and test, respectively.

ScanObjectNN [12] includes 3D data in the real world represented by 3D point clouds. 3D shapes on this dataset are significantly different from those on the ShapeNet or the ModelNet40 datasets, due to the real-world background noise, occlusions, and *etc*. It is composed of 2,902 3D point clouds from 15 categories. Three variants are proposed for different levels of difficulties. The vanilla version of the ScanObjectNN dataset contains only 3D point data cropped from the real-world scene scan using the ground truth bounding box. The second version contains the real-world indoor background information acquired during the scene scanning process. The last version contains the perturbed 3D shapes with background noise. In this track, we adopt the vanilla ScanObjectNN dataset containing clean 3D object point clouds. Some examples compared to the existing datasets are shown in Figure 2.

Based on the existing 2D/3D datasets above, we construct our benchmark from two perspectives. First, we combine all the shapes from ModelNet40 and ShapeNet to form the large-scale collection of 3D CAD models. Due to some discrepancies

Table 1. Detailed statistics of different benchmarks for sketch-based 3D shape retrieval.

Datasets	#Category	Sketches			Shapes	
		#Training	#Test	#Avg/Class	#Total	#Avg/Class
SHREC'13	90	4500	2700	80	1258	14
SHREC'14	171	8550	5130	80	8987	53
STC	44	96009	23992	2727	46612	1059
STW	10	20135	5032	2517	1731	173

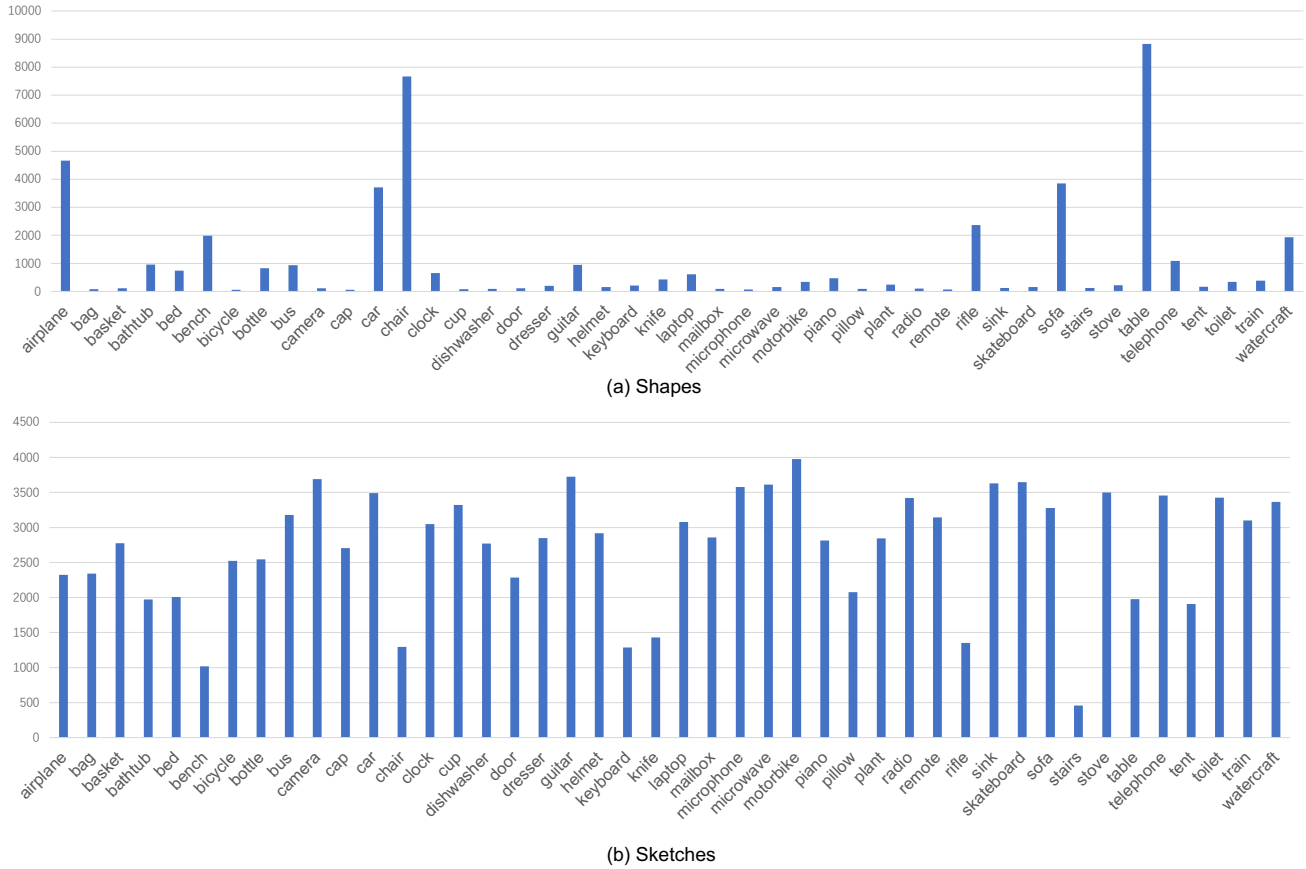


Fig. 3. Detailed statistics of the 44 classes on the STC benchmark w.r.t. Task 1.

between the categories of these two datasets, we finally obtain around 46,000 models from 44 classes in total. According to the models, we select an average of 2,700 sketches from the corresponding categories on QuickDraw. It is noteworthy that some sketches from QuickDraw are too abstract to be recognized even by human beings. So we manually remove those sketches as a preprocessing step. For brevity, we name this benchmark "Sketch to CAD" (STC). Second, all the scanned 3D objects from the vanilla ScanObjectNN dataset are adopted and the corresponding sketches are selected from QuickDraw in a similar manner as with the STC benchmark. Since several categories on ScanObjectNN are missing their counterparts on QuickDraw, we finally select around 1,700 realistic 3D models from 10 classes on ScanObjectNN and an average of 2,500 sketches per class from QuickDraw, constituting our "Sketch to Wild" (STW) benchmark. The detailed statistics of the STC and STW benchmarks can be found in Figures 3 and 4, respectively. In addition, Table 1 shows the comparison between the proposed two benchmarks and the existing SHREC'13 and SHREC'14 benchmarks. It is clearly observed that our proposed STC and STW datasets include significantly more examples for each category, making the SBSR task much more challenging and realistic.

3. Evaluation

In accordance with the constructed STC and STW benchmarks, we proposed two tasks to evaluate the performance of different SBSR algorithms, *i.e.*, sketch-based 3D CAD model retrieval and sketch-based realistic scanned model retrieval. Note that all 3D models in both tasks are provided in the form of point cloud data.

Specifically, in terms of the first task on the STC benchmark, we randomly select 80% sketches from each class for training, and the remaining 20% sketches per class are used for testing/query. Considering the large scale of 3D shapes, we ran-

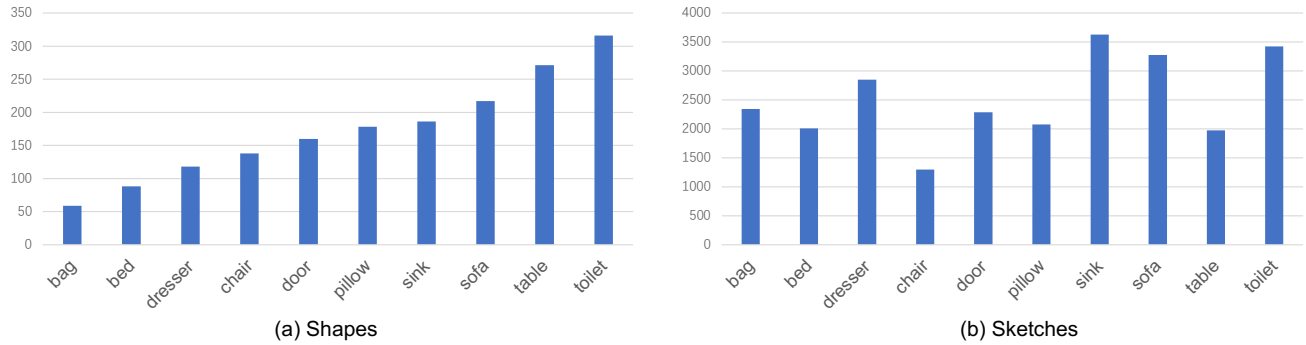


Fig. 4. Detailed statistics of the 10 classes on the STW benchmark w.r.t. Task 2.

domly choose 20% of the total 3D models as the target/gallery dataset to evaluate the retrieval performance. As for the second task on the STW benchmark, similar to the first task, we randomly select 80% sketches from each class for training, and the remaining 20% sketches per class are used for testing/query. Since the number of 3D models is not that large, all the 3D point clouds as a whole are utilized as the target/gallery dataset to evaluate the retrieval performance. Participants are asked to submit the results on the test sets.

3.1. Evaluation Metric

For a comprehensive evaluation of different algorithms, we employ the following widely-adopted performance metrics in SBSR, including nearest neighbor (NN), first tier (FT), second tier (ST), E-measure (E), discounted cumulated gain (DCG), mean average precision (mAP), and precision-recall (PR) curve. The source code to compute all the aforementioned metrics is provided. More specifically, during test, all the participants are required to submit the distance matrices computed based on the test data, where the (i, j)-th entry indicates the distance between the i-th sketch and the j-th shape. The smaller the distance is, the more similar the corresponding sketch and shape are.

4. Participants

A total of 7 teams registered for the SHREC'22 Track on Sketch-Based 3D Shape Retrieval in the Wild and eventually 4 of them submitted the final test results. For the first task, 6 rank list results (runs) for 3 different methods developed by 3 teams have been submitted. For the second task, 4 teams have developed 4 different methods, resulting in the submission of 9 rank list results (runs). The participants and their runs are listed as follows:

- **Team A**: Thien-Tri Cao, Nhat-Khang Ngo, Tuan-Luc Huynh, Hai-Dang Nguyen, and Minh-Triet Tran from VNUHCM-University of Science submitted 1 run for Task 2 (Section 5.2).
- **Team B**: Nhat Hoang-Xuan, Chi-Bien Chu, Khoi-Nguyen Nguyen-Ngoc, Hai-Dang Nguyen, and Minh-Triet Tran from VNUHCM-University of Science submitted 2 runs for Task 1 and 4 runs for Task 2, respectively (Section 5.3).
- **Team C**: Haoyang Luo, Jianning Wang, and Zheng Zhang from Harbin Institute of Technology submitted 1 run for Task 1 and 1 run for Task 2, respectively (Section 5.4).
- **Team D**: Zihao Xin, Yang Wang, Feng Wang, Ying Tang, Haiqin Chen, Yan Wang, Qunying Zhou, Ji Zhang, and Hongyuan Wang from Changzhou University submitted 3 runs for task 1 and 3 runs for task 2, respectively (Section 5.5).

5. Methods

In this section, in line with our motivation, we first introduce two kinds of baseline methods (*i.e.*, a projection-based/multi-view method and a point cloud based one). Then, we present the methods and describe the runs developed by the 4 teams in detail. Since the two tasks only differ in the data source (one is from CAD and the other from realistic scanner), the participated teams basically develop one general cross-modality retrieval framework, which is then trained based on different training sets.

5.1. Baseline Methods

Prior to introducing the methods provided by the participated teams, we first present two baseline methods. Specifically, we keep the sketch feature extraction head the same for two different baselines and vary the 3D model feature extraction head. For the 3D model feature extraction, the first baseline method renders a 3D model into multi-view images, based on which a multi-view method is used for 2D feature extraction. Differently, the second baseline method takes raw point cloud data as input and directly extracts 3D features without the additional rendering step.

5.1.1. Multi-View Method

As shown in Figure 5(a), the multi-view baseline (*i.e.*, Baseline-MV) contains three steps: 1) 3D model rendering, 2) multi-view feature extraction, and 3) sketch feature extraction.

3D Model Rendering. We first render 3D models in the point cloud format to multi-view 2D images. For each 3D point cloud, we render k images that correspond to the projections from different camera views. The rendered images are then fed

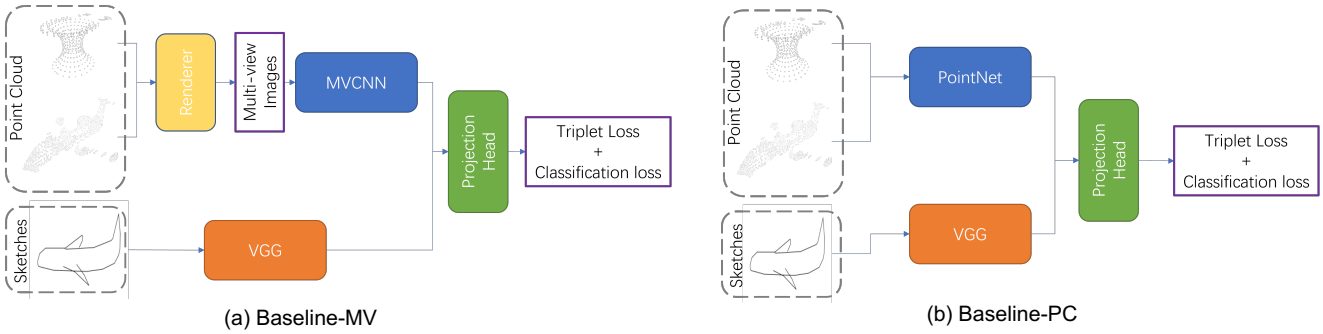


Fig. 5. Overall frameworks of (a) the multi-view baseline (Baseline-MV) and (b) the point cloud based baseline (Baseline-PC).

to a feature learning module to learn multi-view features for the 3D point cloud.

Multi-View Feature Extraction. We follow multi-view convolutional neural networks (MVCNN) [16] to extract multi-view based 3D shape features. As stated above, instead of using raw point clouds as input, the rendered 2D images are fed to the network. The multi-view feature extractor first generates 2D image descriptors for each view and then aggregates each individual descriptor by a max-pooling layer. The extracted multi-view shape features are then fed to multi-layer perceptrons (MLPs) to project the features to a common latent space.

Sketch Feature Extraction. We employ the VGG [17] network for sketch representation learning. Due to the high abstraction of the 2D sketches in this track, we pre-train the VGG network on the ImageNet and fine-tune the network parameters on our newly proposed datasets using the classification loss to better capture the semantic information. Similar to the shape feature extractor, we further adopt a projection head composed of MLPs to project the sketch features to a common latent space shared by the projected shape features.

Objective Function. We train the multi-view baseline with three objective functions. Specifically, our model takes triplets as input. Given N triplets $\{x_i^{sketch}, x_i^+, x_i^-\}$, where x_i^{sketch} is a sketch image, x_i^+ represents a positive point cloud that belongs to the same category as the sketch image, and x_i^- indicates a negative point cloud that is randomly selected from other categories, the first objective function measures the classification ability of the feature extractor, for which we use the cross entropy loss:

$$\mathcal{L}_{sketch} = -\frac{1}{N}\sum_{i=1}^{N} y_{x_i^{sketch}} log(p_{x_i^{sketch}}), \quad (1)$$

where N is the total number of sketch samples, $y_{x_i^{sketch}}$ is the ground-truth label, and $p_{x_i^{sketch}}$ is the predicted softmax probability.

In addition to the loss term above, we also adopt the cross entropy loss to measure the quality of the point cloud feature extractor:

$$\mathcal{L}_{shape} = -\frac{1}{N}\sum_{i=1}^{N}(y_{x_i^+} log(p_{x_i^+}) + y_{x_i^-} log(p_{x_i^-})), \quad (2)$$

where N is the total number of 3D shapes, $y_{x_i^+}$ and $y_{x_i^-}$ are the ground-truth labels of positive and negative samples, respectively, and $p_{x_i^+}$ and $p_{x_i^-}$ are the predicted softmax probabilities for the positive and negative samples, respectively.

Finally, to capture the correspondence between 2D sketches and 3D shapes, we further adopt a soft-margin triplet loss as follows:

$$\mathcal{L}_{tri} = \sum_{i=1}^{N} ln(1 + exp(\|z_{x_i^{sketch}} - z_{x_i^+}\|^2 - \|z^{x_i^{sketch}} - z_{x_i^-}\|^2)), \quad (3)$$

where $z_{x_i^{sketch}}$, $z_{x_i^+}$ and $z_{x_i^-}$ denote the projected sketch feature, positive multi-view 3D shape feature, and negative multi-view 3D shape feature, respectively.

Implementation Details. We use the PyTorch3D library to render the 3D point clouds. We adopt 12 cameras uniformly distributed on a unit sphere where the 3D object is placed at the center of this sphere. In other words, we render each 3D shape into 12 2D images for 3D multi-view feature extraction. As stated above, we use the MVCNN and VGG networks to extract 3D point cloud features and sketch features, respectively. We use MLPs with three hidden layers to project 2D sketch and 3D shape features onto a common latent space. The Adam optimizer with a batch size of 32 is used to train the entire network. The momentum is set to 0.8, and the weight decay rate is set to 10^{-4}. The learning rate starts at 0.001 and is then decreased by a factor of 3 every 40 epochs.

5.1.2. Point Cloud Based Method

As shown in Figure 5(b), the point cloud based baseline (*i.e.*, Baseline-PC) contains two steps: 1) point cloud feature extraction, and 2) sketch feature extraction.

Point Cloud Feature Extraction. In contrast to the multi-view baseline, this baseline directly receives raw point clouds as input, avoiding the time-consuming rendering process. Specifically, we use PointNet [18] as the backbone network for 3D representation learning of the input point clouds. Concretely, PointNet utilizes multi-layer perceptrons (MLPs) to extract per-point feature signatures in the latent space and fuses signatures into high-level global representation by adopting a max-pooling layer. The extracted latent point cloud representations are further fed to a projection head to transform the point cloud features to a common latent space shared by the projected sketch features. We also employ MLPs as the projection head.

Sketch Feature Extraction. Similar to the sketch feature extractor mentioned in the multi-view baseline, we use the pre-trained VGG network and fine-tune it on our newly proposed

datasets to extract sketch features. The feature projection head is applied to the extracted sketch features for learning the common latent space.

Objective Function. We train the point cloud based method using the same objective functions as the multi-view baseline, including two cross-entropy losses (*i.e.*, Eqs. (1) and (2)) and one triplet loss (*i.e.*, Eq. 3)), as described in Sec. 5.1.1.

Implementation Details. The point could based method is trained end-to-end based on the aforementioned objective functions. We use an Adam optimizer with the momentum set to 0.7 and the weight decay rate set to 0.005. The initial learning rate is 0.001 and we decrease it by a factor of 10 every 60 epochs. During training, we use a batch size of 32 and sample 1024 points for each 3D point cloud.

5.2. Team A: Distance-based Methods for Sketch-based 3D Shape Retrieval

Figure 6 shows the pipeline of our approach. Overall, our approach includes three steps: (1) feature extraction, (2) concatenation, and (3) matching.

Sketch Feature Extraction. We employ ResNet18 [19] to extract deep features for sketches. In practice, we fine-tune several available models pre-trained on ImageNet. The SGD optimizer is adopted with a decay learning rate by a factor of 0.1 every 7 epochs, and the momentum is 0.7. Then, we randomly choose 80% of the original training set for training and the remaining for validation. We choose the model with the best validation accuracy during the training process.

Point Cloud Feature Extraction. We use PointNet[18] as the feature extractor for point cloud data. Similar to the training process w.r.t. sketches, we split 80% of the original training set for training and the rest is used for validation. Specifically, we use `sparse_categorical_crossentropy` as the loss and the Adam as the optimizer. The learning rate is manually justified after a number of epochs. We choose the model with the best validation accuracy as the one for inference.

As mentioned above, ResNet18 [19] and PointNet [18] are employed to extract features of 512 dimensions from sketches and point clouds, respectively. Then, we concatenate the feature vectors from both modalities to create positive and negative samples for training a matching model in the next phase. Two vectors belonging to the same classes in the sketch and the point cloud sets are combined to produce a positive sample, and vectors from different classes are used to generate negative samples.

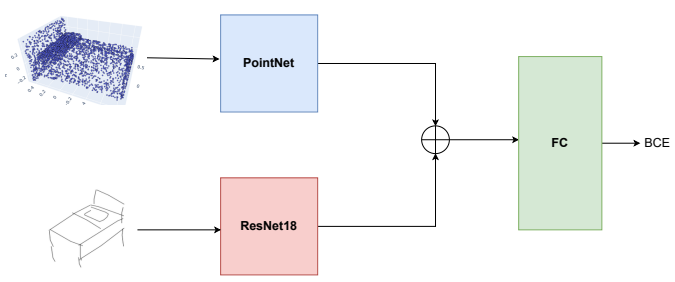


Fig. 6. Overall framework of Team A's approach.

We assume that there exists a match between a point cloud and a sketch if their final score is greater than a threshold, which is set as 0.5 in our experiments. Hence, we regard this problem as a binary classification with two classes matching (1) or not matching (0). We train a binary classifier to distinguish positive and negative inputs. The input of the classifier is a concatenated 1024-d vector. The hidden layers are fully connected (FC). The output of this model is a score that represents how the two inputs match with each other.

5.3. Team B: Common Space Embedding for Sketch-based 3D Shape Retrieval

To retrieve 3D models based on 2D sketches, we first note that a sketch generally does not provide an accurate illustration of the models, but describes the concept presented to the drawer [11]. This gap can be observed in Figure 7. Therefore, it would be helpful if we can identify the concept described by every sketch and 3D model, and retrieve models with a similar concept to the sketch.

The overview of our method can be seen in Figure 8. First, we utilize two feature extractors, one for sketches and the other for 3D models, to obtain features that have P and Q dimensions, respectively. Note that P and Q can be different depending on the models used, creating a gap between the features. To address this, we use two embedding layers to project them to the same D-dimensional space. In this space, we can directly compare the representations using cosine similarity. This allows simple and scalable retrieval by methods such as k-nearest neighbors.

Note that there are many possible implementations of the aforementioned architecture; for example, we might train the joint embeddings using some form of metric learning. Based on the fact that each sketch/model only describes a single concept, we use the classification probability distribution (*i.e.*, one generated from a softmax function) as the common space. This approach is simple as we can train two classifiers separately, and use the probabilities directly. In the following, we will describe our method and the training process in detail.

5.3.1. Motivation

There are a few reasons we do not opt for multi-view approaches. First, the 3D models can be hard to recognize from viewing in 2D. This is because they lack the information of planes and edges, making the process of inferring the shape,

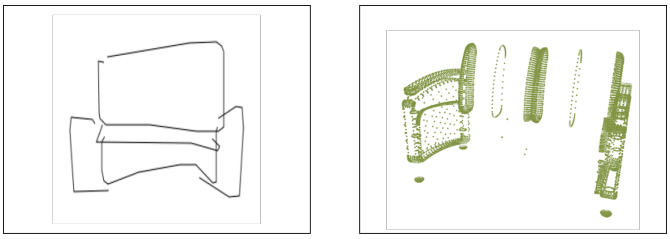


Fig. 7. A sketch and a point cloud from the training set within the same class (sofa). The uneven point density makes it hard to deduce its shape from the points.

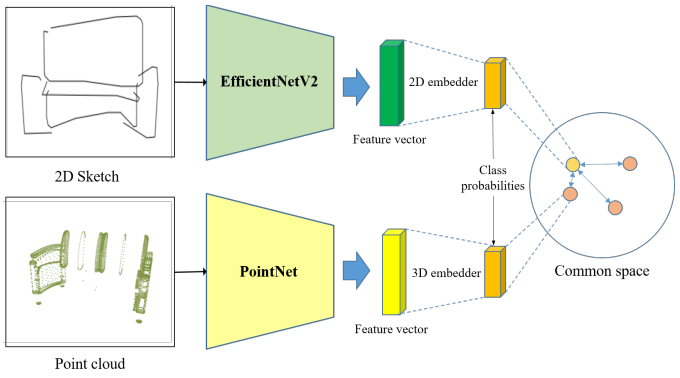


Fig. 8. Overall framework of Team B's approach.

and therefore their class, especially challenging even for humans. Trying to reconstruct the meshes is also not trivial since the object can be hollow at arbitrary locations, and the point meshes can be varyingly dense. For example, a table can be represented by as few as 8 points. For those reasons, we choose to directly model the relations between points to find out the semantics of the model. The PointNet model fulfills this requirement; hence we choose it.

5.3.2. Sketch Model

We employ the recent EfficientNetV2 [20] architecture for image feature extraction, using the classification task as the learning objective. This family of models has many different scales and fast convergence speed when training, which are desirable. For each class, we leave out 100 random samples to create the validation sets. We fine-tune available models pretrained on ImageNet using the remaining training sets. We choose small models because they are pretrained with image sizes close to the size of images in the training set (256×256).

We use the RMSprop optimizer with a piecewise constant decay learning rate scheduler that starts at 10^{-3} and decays to 10^{-6} after 14 epochs. For each task, we pick the model with the best validation accuracy. We note that for these tasks, class imbalance does not cause great troubles, so we do not perform any balancing techniques.

5.3.3. Point Cloud Model

Method. We use the PointNet [18] model to perform the classification of 3D shapes for 2 tasks. In Task 1, we also apply the voting concept to the model (PointNet k-fold) to provide more accurate classification results. The final results will be synthesized from k ($k = 5$ for this task) PointNet models trained separately on different training and test sets.

Training. We split the STC dataset into 5 groups, where the samples of each class are divided equally into each group. For each group, we take it as a validation set and take the remaining groups as a training set, based on which we then train a PointNet model. Therefore, there are 5 different PointNet models trained on the STC dataset. On the STW dataset, we take 80% of the samples as the training set and the rest as the validation set.

Implementation Details. The PointNet models above are trained with the same settings. More details are as follows:

- The input of the networks is a point cloud of 1024 points. If a point cloud has less than 1024 points, we clone the points up to sufficient numbers. If a point cloud has more than 1024 points, we randomly select 1024 of those points as input.
- We apply the Adam optimizer with a learning rate $\alpha = 0.0025, \beta_1 = 0.9, \beta_2 = 0.999$.
- We train each model for 20 epochs with a batch size of 32; finally, we select the weights that give the best accuracy on the corresponding validation set for the inference stage.

5.3.4. Common Space Embedding and Distance-based Retrieval

As mentioned above, we use the classification probabilities as the common embedding space, *i.e.*, $D = 44$ for Task 1 and $D = 10$ for Task 2. The feature dimensions are $P = 1536$ and $Q = 1024$ for all cases. The embedding layers are fully-connected layers with D-dimensional outputs, followed by a softmax activation function.

For each query and each 3D model, we extract their embeddings using our trained models. Then we compute cosine similarity to obtain the distance matrix. Models are ranked by increasing cosine similarity to the sketch query.

We experimented with keeping only the top k ($k = 2, 3, 5$) confidence scores for each sample and zeroing-out the rest, but this resulted in lower metrics on the validation set. This suggests that the full representation is meaningful, even in the low confidence range.

5.3.5. Runs Description

Task 1: We have two runs for this task.

- Run 1: Using the best PointNet model to infer the confidence vector (corresponding to 44 classes).
- Run 2: Using the PointNet k-fold, and the confidence vector will be synthesized from the average of the results of 44 separately trained models.

For both runs, we use a fine-tuned EfficientNetV2-B2 for sketch class probabilities.

Task 2: We propose the following runs for the second task.

- Run 1, 2: Reusing the two models (*i.e.*, PointNet and PointNet k-fold) already running on the STC test set for these runs. Because we want to check if our models trained on the training set of CAD objects can be good enough to recognize 3D objects in the STW test set.
- Run 3: Using the PointMLP [21] model pre-trained on ScanObjectNN.
- Run 4: Using the PointNet model trained on the STW training set.

For this task, we fine-tune an EfficientNetV2-B3 with the same procedure as the first task.

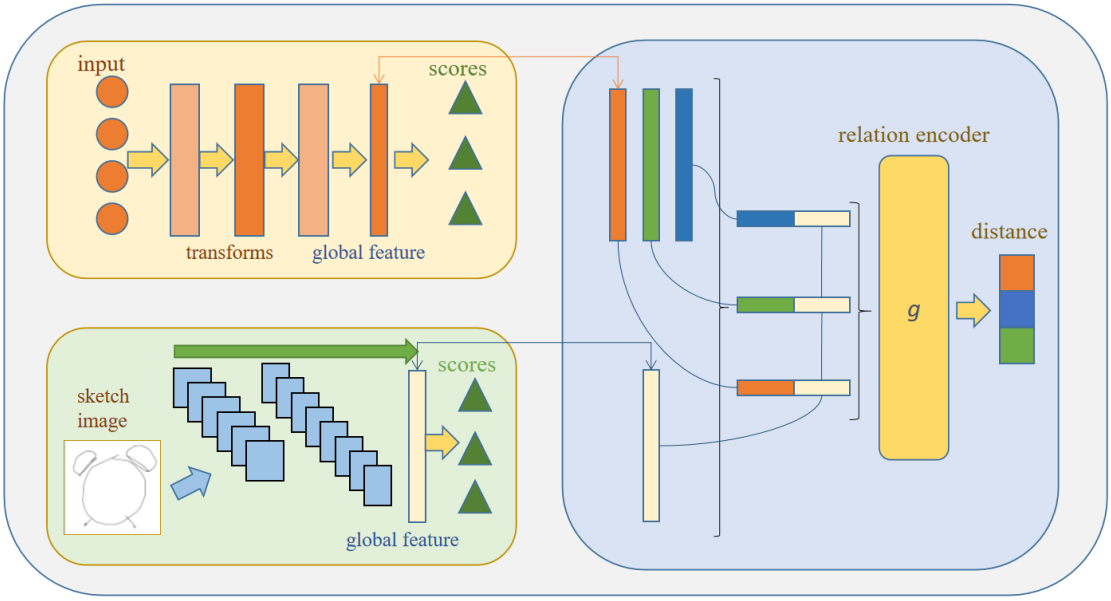


Fig. 9. Overall framework of Team C's proposed DDRN method. The upper left part is the 2D sketch representation module and the lower left part is the 3D shape branch. The relation evaluation module (right part) takes features from two different modalities as inputs for model construction, and it models the cross-modality co-relation for precise retrieval. The whole network adopts pre-training and fine-tuning paradigm with an optimization in an end-to-end manner.

5.4. Team C: Dual-stream Deep Relation Network for Sketch-based 3D Shape Retrieval

We propose a dual-stream deep relation network (DDRN) to solve the problem of sketch-based 3D shape retrieval in the wild. To get informative and discriminative representations from two domains, two sub-modules with deep neural networks are employed to extract most representative features of sketches and 3D shapes, respectively. A cross-modality relation evaluation metric is introduced for measuring the correlation scores among different categories and further guiding the feature extractors. Instead of trivial distance metric, *e.g.*, cosine distance and Euclidean distance, the proposed deep neural relation measurement gives more flexibility and is also helpful to enhance the discrimination of the learning model for the cross-modal retrieval task. The entire DDRN learning model is trained in a two-step manner, *i.e.*, pre-training and fine-tuning. The overall architecture of this method is shown in Figure 9.

5.4.1. 2D Sketch Branch

Sketches are very different from regular photos. To understand the semantics behind these hand drawing, we adopt the widely-used ResNet-50 [19] for deep sketch feature representation learning. The network parameters are initialized by using the pretrained model on ImageNet. To acquire a discriminative representation of the sketches, the pre-training loss of this sub-network is the multi-class cross entropy loss:

$$\mathcal{L}_{CE} = -\frac{1}{N_q}\sum_{i=1}^{N_q}\sum_{c=1}^{M_q} y_{ic} log(p_{ic}), \tag{4}$$

where N_q is the total number of sketch samples, M_q is the number of classes, y_{ic} is the c-th class label of i-th sample's label y_i, and p_{ic} is the corresponding probability acquired from the network output.

5.4.2. 3D Shape Branch

The network architecture for the 3D shape processing is based on PointNet [18]. Instead of being transformed to regular structure, *i.e.*, 3D voxel grids or 2D aspect projection, the point cloud is fed into the network though its irregularity. The collections of point coordinates go directly through several feature extractors, local and global information aggregation layers, and collaborative alignment networks, which are used for extracting their useful components. Other argumentation strategies such as point permutation invariant are also taken into consideration. We pre-train this module on a classification task for better ability to extract 3D shape features and then fine-tune it in the overall unified training framework. The pre-training loss of this 3D shape network is very similar to the sketch network:

$$\mathcal{L}_{CE} = -\frac{1}{N_z}\sum_{i=1}^{N_z}\sum_{c=1}^{M_z} y_{ic} log(p_{ic}), \tag{5}$$

where N_z is the total number of shape samples, and M_z is the number of classes.

5.4.3. Relation Network

Inspired by the well-known Relation Network [22], we propose a relation distance encoder to construct the correlated relationship between sketch image and retrieved 3D shape features.

Firstly, after the aforementioned sketch and 3D shape modal-encoders, we obtain the global feature $\boldsymbol{q}$ of the queried sketch image and the global feature $\boldsymbol{z}_i$ of the retrieved 3D shapes by utilizing the output of the feature layer before classification. In order to obtain the distance matrix between each query point $\boldsymbol{q}$ and the retrieved 3D shapes $\boldsymbol{z}_i$, this model is also required to be capable of measuring the distance d_i between the query point $\boldsymbol{q}$ and each 3D shape sample $\boldsymbol{z}_i$. Herein, we simply concatenate the query point $\boldsymbol{q}$ with each 3D shape samples to form

several query-sample pairs $(\boldsymbol{q}, \boldsymbol{z}_i)$. For each query sample pair, following [22], we put it into a two-layer multi-layer perceptron (MLP). The activation function of the first layer is the linear rectification function(ReLU) and the sigmoid function is attached to the linear out put as the activation function of the second layer. In this way, the given algorithm can obtain the similarity score between the query point and the simulated retrieval samples $s_i \in (0, 1)$:

$$s_i = \sigma(\boldsymbol{W}_2(ReLU(\boldsymbol{W}_1(\boldsymbol{q}\|\boldsymbol{z}_i) + \boldsymbol{b}_1)) + \boldsymbol{b}_2), \tag{6}$$

where σ is the sigmoid function. $\boldsymbol{W}_1$, $\boldsymbol{W}_2$, $\boldsymbol{b}_1$, and $\boldsymbol{b}_2$ are the network parameters, and $\|$ is the concatenation operator. The objective of the relation distance network is modeled as:

$$\min_{\boldsymbol{W}_1,\boldsymbol{W}_2,\boldsymbol{b}_1,\boldsymbol{b}_2} \sum_{i=1}^{m} \|\boldsymbol{s}_i - \mathbf{1}(c(\boldsymbol{q}) = c(\boldsymbol{z}_i))\|^2, \tag{7}$$

where m is the number of sampled classes per batch, $c(\cdot)$ returns the class label of a datapoint, and $\mathbf{1}(\cdot)$ has value 1 when the condition is true, or has value 0, otherwise. This objective gives features of the same category a higher similarity score, while lower scores are achieved when features are from different classes. In this work, we use $d_i = 1 - s_i$ as the distance score between the retrieved samples and the query point, so that the query sample pairs with large (small) similarities have close (far) distances. The relation distance network can explore the correlation degree of features from different modalities.

5.4.4. Optimization

The overall optimization process of our method is simply summarized in these steps for the two tasks:

- Step 1: Pre-train the ResNet-50 on a sketch dataset using Eq. (4).
- Step 2: Pre-train the PointNet on a corresponding 3D shape dataset using Eq. (5).
- Step 3: Acquire the discriminative representations of the two modals with the trained two sub-networks.
- Step 4: Train the relation distance network with Eq. (7).

5.5. Team D: Suggestive Contours with Domain-aware Squeeze-and-Excitation for Sketch-based 3D Shape Retrieval

We propose a new sketch-based 3D shape retrieval model to bridge the gap between sketches and 3D point clouds. This network is divided into the suggestive contour of generating line drawing from 3D models, and the DASE network [23] using SE module and multiplicative Euclidean margin softmax loss.

Learning common feature representations for sketch and 3D point cloud domains is non-trivial. As shown in Figure 10, sketches are iconic and abstract with various deformation levels; meanwhile, photos are realistic images with shape information. It is distinct what edges are in an image by traditional image process algorithms such as Canny, but it is difficult to 3D point clouds. To draw an accurate line drawing of a 3D point cloud, the algorithm must understand the 3D shape, and select

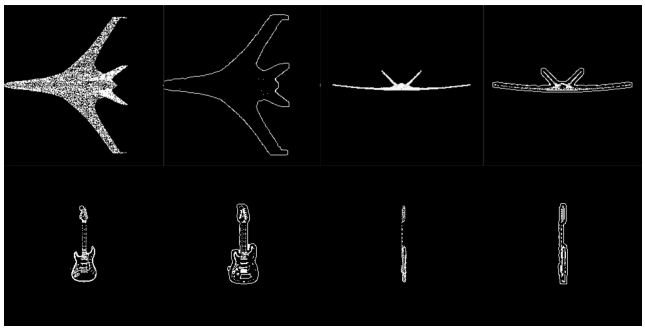


Fig. 10. Different views of the 3D point clouds. From left to right: frontal projection images of point cloud, suggestive contours images of frontal projection, side projection images of point cloud, suggestive contours images of side projection.

lines which can represent the shape of the 3D point cloud. We thus adopt a method [24] for generating line drawing from point clouds.

Our model learns a single network to analyze the projection of the 3D point clouds and sketches jointly rather than using independent sub-networks to process the projection images and sketches separately. The SiameseNet-like network is efficient in learning the embedding space across different domains. Liu *et al.* [25] proposed a semi-heterogeneous network, including a shared Siamese sub-network and an independent sub-network for each domain.

We add the Domain-Aware Squeeze-and-Excitation (DASE) module after each residual block, for learning an effective feature representation to provide an explicit mechanism for reweighting the importance of channels. As shown in Figure 11, the DASE module has an encoder-decoder structure and the sigmoid activation. And we append one binary value code indicates whether comes from the sketcher domain or the suggestive contours images of projection domain, to the low dimensional embedding. After the sigmoid activation, we can get the feature attention vector. Therefore, this conditional structure can help capture different features of the input image.

The SE module proposed by [26], using an encoder network

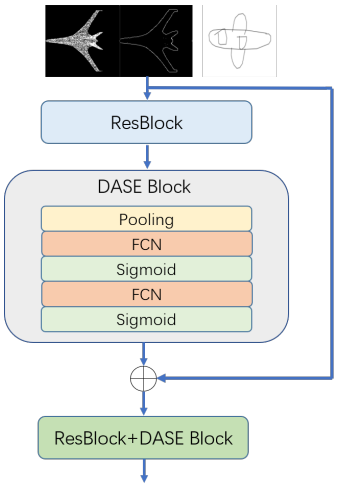


Fig. 11. The structure of the DASE module.

Table 2. Performance (in %) comparison on the STC benchmark for sketch-based 3D CAD model retrieval (Task 1).

Team	#Run	NN	FT	ST	E	DCG	mAP
Team B	1	92.25	85.89	92.06	48.50	95.11	89.28
	2	92.23	86.96	92.77	49.04	95.40	90.18
Team C	1	1.08	1.54	3.10	0.11	36.29	2.05
Team D	1	2.35	1.94	3.92	0.36	38.16	2.23
	2	2.63	1.80	3.62	0.16	37.57	1.99
	3	2.10	1.95	3.90	0.35	37.84	2.03
Baseline-MV	1	2.91	1.82	3.56	0.34	37.65	2.00
Baseline-PC	1	1.65	1.85	4.04	0.33	37.90	2.01

Table 3. Performance (in %) comparison on the STW benchmark for sketch-based realistic scanned model retrieval (Task 2). * The first 3 runs of Team B are just for reference as they use auxiliary data for training.

Team	#Run	NN	FT	ST	E	DCG	mAP
Team A	1	39.73	44.71	63.10	14.47	77.17	46.67
Team B	1*	1.19	12.55	22.63	4.78	60.57	13.43
	2*	1.05	7.97	18.69	1.84	57.37	10.26
	3*	0.34	7.67	16.94	1.09	56.5	9.99
	4	71.16	61.29	71.81	25.18	86.18	67.31
Team C	1	10.93	11.13	20.58	3.86	60.18	15.15
Team D	1	10.23	9.85	19.52	3.08	58.75	10.09
	2	9.18	9.67	19.51	3.18	58.75	10.05
	3	9.94	9.73	19.45	3.03	58.73	10.09
Baseline-MV	1	9.32	13.10	26.89	6.08	61.07	15.52
Baseline-PC	1	9.32	13.74	28.38	7.59	62.69	18.81

to squeeze the convolution features into a low dimensional embedding, and the characteristic responses of channels are adaptively recalibrated by explicitly simulating the interdependence between different channels. So the DASE module learned between two domains can explore the shared semantics in a common feature space.

The loss function plays an important role in optimizing the networks over the recognition tasks successfully, particularly in our cross-domain scenario. In order to embed photos and sketches into the shared space, we use a Multiplicative Euclidean Margin Softmax (MEMS) loss, which utilizes the strategy of maximizing intra-class distance and minimizing inter-class distance.

Specifically, the maximum intra-class distance can be defined by $\max_{x,x'\in D_y} x' \in (\cup_{y'\neq y} D_{y'})$and the minimum inter-class distance would be min $x \in D_y, x' \in (\cup_{y'\neq y} D_{y'}) d(x, x')$, where $d(\cdot)$ can be any kind of differentiable distance metric if the category y is given. Therefore, we optimize our framework by forcing the maximum intra-class distance to be less than the minimum inter-class distance to develop our objective.

$$\max_{\boldsymbol{x},\boldsymbol{x'}\in\mathcal{D}_y} \mathrm{d}\left(x, x'\right) \leq \min_{\boldsymbol{x}\in\mathcal{D}_y,\boldsymbol{x'}\in\left(\cup_{y'\neq y}\mathcal{D}_{y'}\right)} \mathrm{d}\left(x, x'\right) \tag{8}$$

We compute the prototypes $\{c_y\} \subset \chi$of each class, and characterize the distribution of instances in feature space. Instances will be closer to their corresponding prototypes than others in feature space if the MEMS loss is optimized well.

$$\forall\left(x, y_x\right) \in \mathcal{D},\ m \cdot \mathrm{d}\left(x, \boldsymbol{c}_{y_x}\right) \leq \mathrm{d}\left(x, \boldsymbol{c}_y\right) \tag{9}$$

where $m \geq 1$ is referred to as the margin constant.

We denote $R_{y,y'}$ as a region in the feature space for convenience, and R_y as a region such that:

$$x \in \mathcal{R}_{y,y'} \Leftrightarrow m \cdot \mathrm{d}\left(x, c_y\right) \leq \mathrm{d}\left(x, c_{y'}\right) \tag{10}$$

$$x \in \mathcal{R}_y \Leftrightarrow m \cdot \mathrm{d}\left(x, \boldsymbol{c}_y\right) \leq \mathrm{d}\left(x, \boldsymbol{c}_{y'}\right) \tag{11}$$

which considers different data distributions of each class. $\mathcal{R}_y = \bigcap_{y'\neq y} \mathcal{R}_{y,y'}$ is easy to prove. If Eq. (9) holds, we can derive a sufficient condition for Eq. (8):

$$\forall y \in \mathcal{Y},\ \max_{x,\boldsymbol{x'}\in\mathcal{R}_y} \mathrm{d}\left(x, x'\right) \leq \min_{x\in\mathcal{R}_y, x'\in\left(\cup_{y'\neq y}\mathcal{R}_{y'}\right)} \mathrm{d}\left(x, x'\right). \tag{12}$$

To incorporate this spirit and optimize Eq. (9) by the softmax loss, this gives us a novel loss function, *i.e.*, the MEMS loss:

$$\mathcal{L}_{MEMS} = \frac{1}{N}\sum_{i=1}^{N} -\log \frac{e^{-m^2\|x_i - c_{y_i}\|_2^2}}{e^{-m^2}\left\|x_i - c_{y_i}\right\|_2^2 + \sum_{j\neq y_i} e^{-\|x_i - c_j\|_2^2}} \tag{13}$$

where x_i indicates the feature extracted by the last layer of feature representation network. The center of j-th category is c_j. Rather than directly using the average feature center, we take the center c_j as the parameters and update c_j dynamically. We employ the negative squared Euclidean distance to measure the confidence of x_i being $\left(-\left\|x_i - c_j\right\|_2^2\right)$. Particularly, in binary classification, x_i is labelled as class 1 if $m\|x - c_1\|_2 < \|x - c_2\|_2$, and otherwise, as class 2.

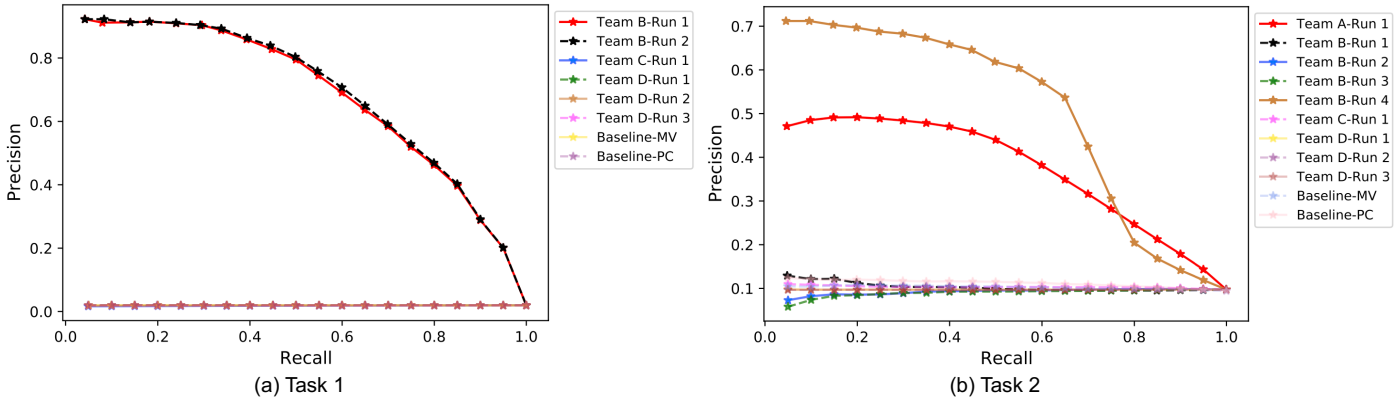


Fig. 12. Precision-recall curves for all the runs of the 4 teams in terms of (a) Task 1 and (b) Task 2.

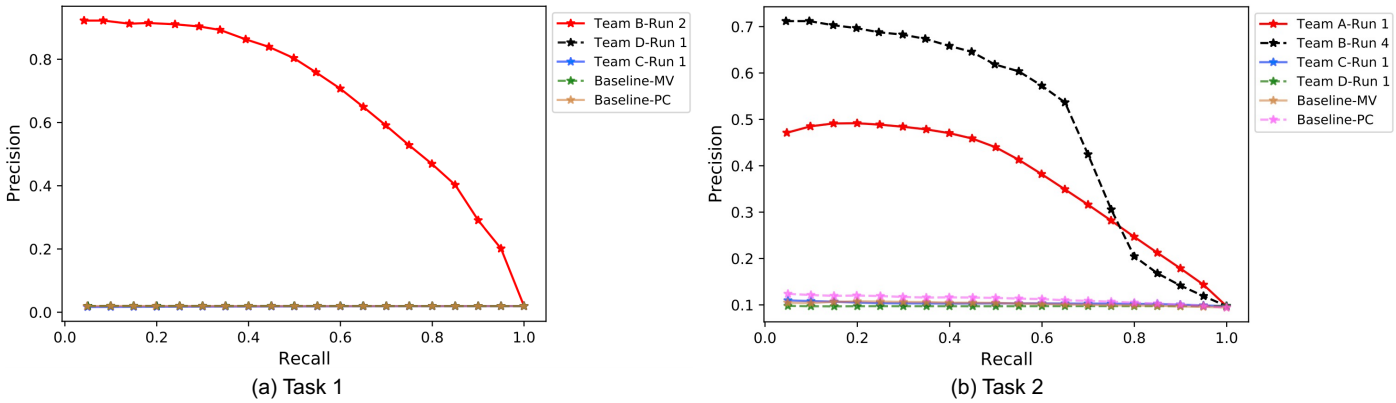


Fig. 13. Precision-recall curves for the best runs of the 4 teams in terms of (a) Task 1 and (b) Task 2.

6. Results and Discussions

In this section, we first show the comparative results of the two baseline methods on the STC and STW benchmarks, and then demonstrate the evaluation results for each task, respectively, based on the distance matrices submitted by the four teams. The retrieval performance is measured by the seven metrics (*i.e.*, NN, FT, ST, E, DCG, mAP, PR) as introduced in Section 3.1.

6.1. Baseline Results

We show the evaluation results of the two baseline methods for the two tasks in Tables 2 and 3, respectively. From the tables, we can see that both baselines achieve comparable but poor performance on the STC benchmark. This is probably due to the large scale of this dataset which consists of ~120k sketches in total. On the STW benchmark, Baseline-PC slightly outperforms Baseline-MV, indicating that point cloud based representations are relatively better than multi-view representations on this realistic dataset. We also show the precision-recall curves in Figures 12 and 13, where we have the similar observations. Moreover, the retrieval time during the test phase is reported in Fig. 14. We can see that Baseline-PC is much more efficient than Baseline-MV, by avoiding the time-consuming rendering process. For instance, Baseline-PC is around 5 times faster than Baseline-MV on STC. This indicates that directly representing 3D point clouds not only mimics the realistic setting but also is more practical in real applications.

6.2. Results for Task 1

Since Task 1 is of much larger scale than Task 2, there are only 3 teams participating in this challenging task. We show the detailed comparative results in Table 2 and Figures 12(a) and 13(a). As can be seen from the table and figures, Team C and Team D fail to obtain satisfactory performance, which is comparable to that of the two baseline methods. Team B performs the best and there exists a large performance gap between the team and the other two. For instance, their 2nd run achieves over 90% in mAP. This is probably due to that Team B employs a common embedding space of the same dimension with the number of categories of sketches/shapes. In other words, once the classification network is optimized, the common embedding is equivalent to the classification probabilities, and is thus very effective.

6.3. Results for Task 2

Table 3 and Figures 12(b) and 13(b) illustrate the performance obtained by 9 runs of 4 teams. Note that the first 3 runs of Team B are just for reference as they use auxiliary data for training. From the table and figures, we can see that, once again, Team B outperforms all the other teams by a large margin. The model trained on the STW benchmark (*i.e.*, Team B-run 4) is

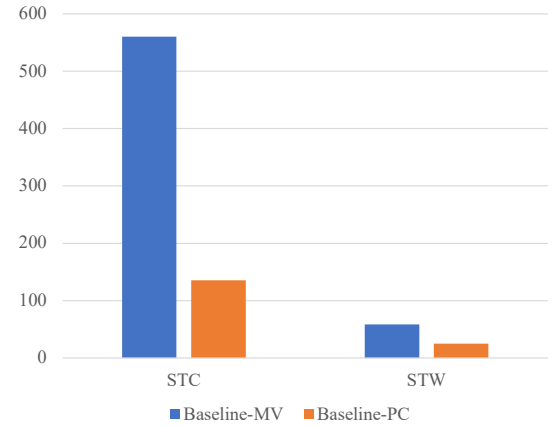


Fig. 14. Runtime (in seconds) comparison between the two baseline methods on the STC and STW benchmarks.

superior to those trained using auxiliary data (*i.e.*, Team B-run 1-3), showing the differences between our realistic dataset with others. Furthermore, Team B's performance for Task 2 is lower than Task 1, indicating the challenges of retrieving 3D shapes in the wild. Team C achieves comparable performance with Baseline-MV, but underperforms Baseline-PC. And Team D ranks the last - their performance is lower than both baseline methods. Interestingly, Team C and Team D achieve higher mAPs for Task 2 than the ones for Task 1. This is probably because the scale of the STW dataset is much smaller than that of STC, and more importantly, all the 3D models on STW are involved during both the training and the test processes.

6.4. Discussions

From the results above, we can see that Team C and Team D obtain unsatisfactory results for both tasks. In some cases, their results are lower than the point cloud based baseline method (*i.e.*, Baseline-PC). After looking into the data themselves, one cause of this defect may turn out to be the class imbalance, which is especially obvious on the STC benchmark. Team C relies on the constructed positive and negative pairs to train the relation network. Due to the imbalance, the positive pairs are overwhelmed by the negative ones, making the model hard to capture useful similarity information. One suggestion for them to improve the performance may be developing a better sampling method to generate appropriately balanced pairs, or simply changing the weighting factors between positive and negative pairs. Another reason that leads to the unsatisfactory performance could be that point clouds are too abstract and cannot represent the shape as detailed as meshes, which are available on previous SBSR datasets. In such cases, local geometric details may be lost after 2D projections. Therefore, we suggest Team D to further integrate 3D point cloud representations into their framework to make the overall representations more robust and discriminative. In addition to the challenging benchmarks themselves, the unsatisfactory performance may be also related to the design of their methods. In terms of Team C's method, the core component is the relation encoder that pays specific attention to the relationship between the two modalities. However, it unintentionally omits the discriminability of the individual modality. Although cross-entropy losses are used during pre-training, such classification losses are still needed to ensure the learned features from the corresponding modality to be more discriminative when training the whole DDRN network. As for Team D's approach, we observe that their framework focuses on learning discriminative feature representations for both modalities through a single network architecture. In this case, there is no interaction/alignment between the two modalities, thus leading to inferior results. In summary, both the challenging benchmarks and the proposed frameworks lead to the unsatisfactory results. We encourage them to focus on the discriminability of the single modality as well as the alignment between the two modalities when designing the SBSR frameworks in the future.

7. Conclusions

Encouraged by the success of previous SHREC tracks on sketch-based 3D shape retrieval, this track aims to further foster this important research theme by introducing two realistic and challenging tasks and the corresponding STC and STW benchmarks. Four teams have participated in the two tasks and contributed 15 runs of their proposed methods. Overall, this track provides large-scale benchmarks for evaluating existing sketch-based 3D shape retrieval methods and at the same time for fostering novel approaches and new research directions by mimicking real-world application scenarios.

In future works, from the methodology perspective, learning from imbalanced data deserves further exploration, especially for the first task as the number of CAD shapes per category varies significantly. Besides, how to learn a common/shared latent space that is able to align the representations of the two modalities well is also a possible direction for improving the retrieval performance. Last but not least, for the realistic 3D models, a possible way for performance improvement is to refine the 3D models by noise removal or data augmentation techniques. From the challenge perspective, we plan to expand the size of the STW dataset by adopting other realistic 3D datasets. Another future direction could be exploring the 2D-3D cross-modality retrieval in other application scenarios, such as scene retrieval. Furthermore, in addition to the retrieval task, the challenge can also be extended to other settings, such as 2D sketch-based 3D shape synthesis/generation.

Acknowledgments

We sincerely thank the organizers of SHREC 2022 and 3DOR 2022 for their help in the organization of this track. We thank the researchers who collected the sketch and 3D shape benchmarks used in this track. We thank the anonymous reviewers for providing constructive comments, which helped us to improve and clarify this work. We also thank Peng Zheng who helped us to perform the evaluation.

This work was partially supported by the Fundamental Research Funds for the Central Universities (No. NS2022083). This work was also partially supported by the NYUAD Center for Artificial Intelligence and Robotics (CAIR), funded by Tamkeen under the NYUAD Research Institute Award CG010. The

work of Shuaihang Yuan was supported by the NYUAD Global Ph.D. Student Fellowship, funded by the Inception Institute of Artificial Intelligence. The team from Changzhou University was supported by the National Natural Science Foundation of China (No. 61976028). The work of the teams from University of Science, VNU-HCM (HCMUS) was funded by Vingroup and supported by Vingroup Innovation Foundation (VINIF) under project code VINIF.2019.DA19.

References

[1] Li, B, Lu, Y, Godil, A, Schreck, T, Bustos, B, Ferreira, A, et al. A comparison of methods for sketch-based 3d shape retrieval. Computer Vision and Image Understanding 2014;119:57–80.

[2] Chen, J, Qin, J, Liu, L, Zhu, F, Shen, F, Xie, J, et al. Deep sketch-shape hashing with segmented 3d stochastic viewing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, p. 791–800.

[3] Chen, J, Fang, Y. Deep cross-modality adaptation via semantics preserving adversarial learning for sketch-based 3d shape retrieval. In: Proceedings of the European Conference on Computer Vision (ECCV). 2018, p. 605–620.

[4] Li, B, Lu, Y, Godil, A, Schreck, T, Aono, M, Johan, H, et al. Shrec'13 track: large scale sketch-based 3d shape retrieval. In: Eurographics Workshop on 3D Object Retrieval. 2013,.

[5] Li, B, Lu, Y, Li, C, Godil, A, Schreck, T, Aono, M, et al. Shrec'14 track: Extended large scale sketch-based 3d shape retrieval. In: Eurographics workshop on 3D object retrieval; vol. 2014. 2014, p. 121–130.

[6] Li, B, Lu, Y, Duan, F, Dong, S, Fan, Y, Qian, L, et al. Shrec'16 track: 3d sketch-based 3d shape retrieval. In: Eurographics workshop on 3D object retrieval. 2016,.

[7] Yuan, J, Li, B, Lu, Y, Bai, S, Bai, X, Bui, NM, et al. Shrec'18 track: 2d scene sketch-based 3d scene retrieval. In: Eurographics Workshop on 3D Object Retrieval. 2018, p. 29–36.

[8] Dey, S, Riba, P, Dutta, A, Llados, J, Song, YZ. Doodle to search: Practical zero-shot sketch-based image retrieval. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2019, p. 2179–2188.

[9] Sangkloy, P, Burnell, N, Ham, C, Hays, J. The sketchy database: learning to retrieve badly drawn bunnies. ACM Transactions on Graphics 2016;35(4):1–12.

[10] Eitz, M, Hays, J, Alexa, M. How do humans sketch objects? ACM Transactions on graphics 2012;31(4):1–10.

[11] Ha, D, Eck, D. A neural representation of sketch drawings. In: International Conference on Learning Representations. 2018,.

[12] Uy, MA, Pham, QH, Hua, BS, Nguyen, T, Yeung, SK. Revisiting point cloud classification: A new benchmark dataset and classification model on real-world data. In: Proceedings of the IEEE/CVF international conference on computer vision. 2019, p. 1588–1597.

[13] Wu, Z, Xiong, Y, Yu, SX, Lin, D. Unsupervised feature learning via non-parametric instance discrimination. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 3733–3742.

[14] Chang, AX, Funkhouser, T, Guibas, L, Hanrahan, P, Huang, Q, Li, Z, et al. Shapenet: An information-rich 3d model repository. arXiv preprint arXiv:151203012 2015;.

[15] Xiao, J, Hays, J, Ehinger, KA, Oliva, A, Torralba, A. Sun database: Large-scale scene recognition from abbey to zoo. In: 2010 IEEE computer society conference on computer vision and pattern recognition. IEEE; 2010, p. 3485–3492.

[16] Su, H, Maji, S, Kalogerakis, E, Learned-Miller, EG. Multi-view convolutional neural networks for 3d shape recognition. In: Proc. ICCV. 2015,.

[17] Simonyan, K, Zisserman, A. Very deep convolutional networks for large-scale image recognition. In: International Conference on Learning Representations. 2015,.

[18] Qi, CR, Su, H, Mo, K, Guibas, LJ. Pointnet: Deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2017, p. 652–660.

[19] He, K, Zhang, X, Ren, S, Sun, J. Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2016, p. 770–778.

[20] Tan, M, Le, QV. Efficientnetv2: Smaller models and faster training. In: Meila, M, Zhang, T, editors. Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event; vol. 139 of *Proceedings of Machine Learning Research*. PMLR; 2021, p. 10096–10106.

[21] Xu Ma Can Qin, HYHRYF. Rethinking network design and local geometry in point cloud: A simple residual MLP framework. In: ICLR. 2022, p. 1–15.

[22] Sung, F, Yang, Y, Zhang, L, Xiang, T, Torr, PH, Hospedales, TM. Learning to compare: Relation network for few-shot learning. In: Proceedings of the IEEE conference on computer vision and pattern recognition. 2018, p. 1199–1208.

[23] Lu, P, Huang, G, Lin, H, Yang, W, Guo, G, Fu, Y. Domain-aware se network for sketch-based image retrieval with multiplicative euclidean margin softmax 2018;.

[24] DeCarlo, D, Finkelstein, A, Rusinkiewicz, S, Santella, A. Suggestive contours for conveying shape. ACM Transactions on Graphics (Proc SIGGRAPH) 2003;22(3):848–855.

[25] Liu, L, Shen, F, Shen, Y, Liu, X, Shao, L. Deep sketch hashing: Fast free-hand sketch-based image retrieval. IEEE 2017;.

[26] Jie, H, Li, S, Gang, S, Albanie, S. Squeeze-and-excitation networks. IEEE Transactions on Pattern Analysis and Machine Intelligence 2017;PP(99).

SHREC 2022: Fitting and recognition of simple geometric primitives on point clouds

Chiara Romanengo[1, *], Andrea Raffo[1, *, †], Silvia Biasotti[1], Bianca Falcidieno[1], Vlassis Fotis[2], Ioannis Romanelis[2], Eleftheria Psatha[2], Konstantinos Moustakas[2], Ivan Sipiran[3], Quang-Thuc Nguyen[4,5,6], Chi-Bien Chu[4,6], Khoi-Nguyen Nguyen-Ngoc[4,6], Dinh-Khoi Vo[4,6], Tuan-An To[4,6], Nham-Tan Nguyen[4,6], Nhat-Quynh Le-Pham[4,6], Hai-Dang Nguyen[4,5,6], Minh-Triet Tran[4,5,6], Yifan Qie[7], and Nabil Anwer[7]

[1] *Istituto di Matematica Applicata e Tecnologie Informatiche "E. Magenes", Consiglio Nazionale delle Ricerche, Via de Marini 6, 16149 Genova, Italy*
[2] *Department of Electrical and Computer Engineering, University of Patras, Patras, 26504, Greece*
[3] *Department of Computer Science, University of Chile, Beauchef 851, Santiago, Chile*
[4] *University of Science, VNU-HCM, Ho Chi Minh City, Vietnam*
[5] *John von Neumann Institute, VNU-HCM, Ho Chi Minh City, Vietnam*
[6] *Vietnam National University, Ho Chi Minh City, Vietnam*
[7] *Automated Production Research Laboratory (LURPA), ENS Paris-Saclay, Université Paris-Saclay, 91190 Gif-sur-Yvette, France*

Abstract

This paper presents the methods that have participated in the SHREC 2022 track on the fitting and recognition of simple geometric primitives on point clouds. As simple primitives we mean the classical surface primitives derived from constructive solid geometry, i.e., planes, spheres, cylinders, cones and tori. The aim of the track is to evaluate the quality of automatic algorithms for fitting and recognising geometric primitives on point clouds. Specifically, the goal is to identify, for each point cloud, its primitive type and some geometric descriptors. For this purpose, we created a synthetic dataset, divided into a training set and a test set, containing segments perturbed with different kinds of point cloud artifacts. Among the six participants to this track, two are based on direct methods, while four are either fully based on deep learning or combine direct and neural approaches. The performance of the methods is evaluated using various classification and approximation measures.

Keywords: geometric primitives, primitive fitting, primitive recognition, primitive descriptors, SHREC.

1 Introduction

Fitting and recognition are of paramount importance in multiple application domains, such as reverse engineering and data compression. In most real-world scenarios, acquisition methods tend to produce data affected by noise and potentially by other imperfections (e.g., non-uniform or low sampling density, misalignment and missing data). The literature on methods for fitting and recognition of simple primitives is quite consolidated [1] and recently we have seen the growth of data-driven methods, see for instance [2, 3, 4, 5, 6] resulting in a growing demand for appropriate datasets for training and their evaluation. It is therefore worth analyzing and comparing the outcome of simple primitive fitting methods on a controlled dataset with different kinds of perturbations and primitive variations and in terms of different evaluation measures.

[*] Track organizer
[†] Corresponding author

1.1 Related datasets

Given the strong interest in the problem of fitting geometric primitives, several datasets are becoming available but, generally, they are not specifically designed for simple geometric primitives, possibly affected by perturbations, missing points and local variations. A first collection of one million CAD models targeted to data-driven problems was provided by [7], in which each object is associated to a ground truth for different quantities, as for example the indices of the patch segmentation and the analytic representation of surfaces and curves. A lot of surface types are considered and the models are represented as triangle meshes. Then, it is worth mentioning [8], a benchmark of point clouds representing CAD objects aimed at evaluating methods for detecting simple geometric primitives. Despite having some features similar to our purposes, such as sub-sampled clouds and missing parts, it is not big enough and was designed to test model segmentation methods. A dataset of mechanical components is proposed by [2], which generated the point samples from some models of the American National Standards Institute (ANSI) provided by TraceParts. In this collection, four types of primitives are considered (plane, sphere, cylinder, cone). Another synthetic dataset is presented in [9], but it proposes only cylinders, spheres and cubes as surface types. These existing datasets are not suitable for our purposes since our intent is to analyze the different methods for fitting geometric primitives on point clouds that present different types of specific artifacts. Indeed, in our dataset all point clouds have been artificially generated from exact surfaces and then optionally perturbed by simulating uniform or Gaussian noise of different intensity, undersampling, missing parts, small deformations, or a combination of such point cloud artifacts. In addition, we are not interested in the whole segmentation process, but we are focused on a sub-issue of it, that is the problem of classification and recognition of segments.

1.2 Contribution

The aim of this SHREC track is to evaluate the quality of automatic algorithms for fitting and recognising geometric primitives in point clouds. The goal is to identify, for each point cloud, its primitive type (i.e., planes, spheres, cylinders, cones and tori) and some geometric descriptors.

To this aim, a dataset of 3D segments, corresponding to parts of primitives and represented as point clouds, divided in training set and test set was generated. For each segment in the training set, its primitive type and an analytical representation of it were made available to the track participants. For each point cloud in the test set, the goal of the contest is twofold: to recognise the primitive type from the data and to provide an analytical representation of the primitive recognised. The relevance or non-relevance of the simple primitives has been evaluated on the basis of a ground truth that derived from the knowledge of the primitive that originated that segment. Finally, the performance of the methods is discussed using classification, recognition and approximation measures that quantify the adherence to the original primitives and the goodness of fit.

1.3 Organization

The remainder of this paper is organized as follows. Section 2 describes the characteristics of the proposed benchmark: the dataset, the ground truth, and the quality measures chosen to evaluate classification, approximation and recognition of primitives. Section 3 presents a description of the six automatic algorithms that participate to this SHREC track: two of them are fully-based on neural networks, two are direct methods, while the remaining ones use deep learning in an initial classification step and proceed by applying direct methods. Finally, Section 4 provides an accurate comparative analysis of the results obtained by the six methods.

2 The benchmark

The benchmark is available at `https://github.com/chiararomanengo/SHREC2022.git`.

2.1 Dataset and ground truth

The dataset is composed of 46,925 three-dimensional segments represented as point clouds; it is divided into a training set and a test set that contain, respectively, 46,000 and 925 point clouds. Each point cloud is provided in a TXT file listing one triplet per line.

The point clouds (see Figure 1) are generated by sampling classical surface primitives derived from constructive solid geometry (i.e., planes, cylinders, spheres, cones and tori), by means of the following procedure. First, point clouds are sampled by using parametric equations in their canonical form, i.e., centered at the origin of the coordinate axes and with rotational axis aligned with the $z-$axis; the geometric quantities that define each primitive (i.e., amplitudes and radii, if any) are assigned randomly. Second, to obtain segments of different shapes, several cuts are enforced by exploiting random planes. Third, we randomly apply translations and/or rotations to recover primitives in their general position. Lastly, each point cloud is (potentially) processed so that it can be:

- *A0 – Clean.* No perturbation is applied.
- *A1 – Perturbed by uniform noise of different intensities.* The noise is obtained by sampling uniform distributions of the form $\mathcal{U}(-\frac{1}{n}, \frac{1}{n})$, being $3 \leq n \leq 20$ random, and adding such perturbations to a random percentage of the points.
- *A2 – Perturbed by Gaussian noise of different intensities.* The noise is obtained by sampling normal distributions among $\mathcal{N}(-\frac{1}{n}, \frac{4}{n^2})$, being $10 \leq n \leq 30$ random, and adding such perturbations to a random percentage of the points.
- *A3 – Clean but affected by undersampling.* We randomly select a percentage of points to be removed.
- *A4 – Clean but affected by missing parts.* We randomly choose a point of the point cloud; then, we remove all points contained inside the sphere having the selected point as center and radius assigned randomly.
- *A5 – Perturbed by uniform noise of different intensities and undersampled.*
- *A6 – Perturbed by Gaussian noise of different intensities and undersampled.*
- *A7 – Perturbed by uniform noise of different intensities and affected by missing parts.*
- *A8 – Perturbed by Gaussian noise of different intensities and affected by missing parts.*
- *A9 – Clean but with local deformations.* We randomly select a point of the point cloud and we apply a bivariate Gaussian centered at the selected point with a random covariance matrix.

For the sake of brevity, we will often refer to these point cloud artifacts as *perturbation types*. For each perturbation type A0, ..., A8, training set and test set contain, respectively, 5,000 and 100 point clouds; for perturbation type A9, training set and test set count, respectively, 1,000 and 25 point clouds.

For each point cloud of the training set, the track participants were provided with a ground truth TXT file that includes the primitive type and the geometric parameters required to identify each segment. More precisely, each file contains a column vector $\mathbf{v}$ defined as follows:

- *Plane.* We list the primitive type (for planes, codified as 1), the unit normal vector[1] $\mathbf{a}$ and a point sampled on the plane $\mathbf{P}$. Then, the file provides the vector $\mathbf{v} = [1, \mathbf{a}^\mathsf{t}, \mathbf{P}^\mathsf{t}]^\mathsf{t}$.

[1] To be precise, the definite article "the" is here used improperly. Indeed, there are two unique unit normal vectors that can be used to defined the same plane, or rotational axis. In our case, we recover uniqueness by enforcing the first nonzero component to be positive.

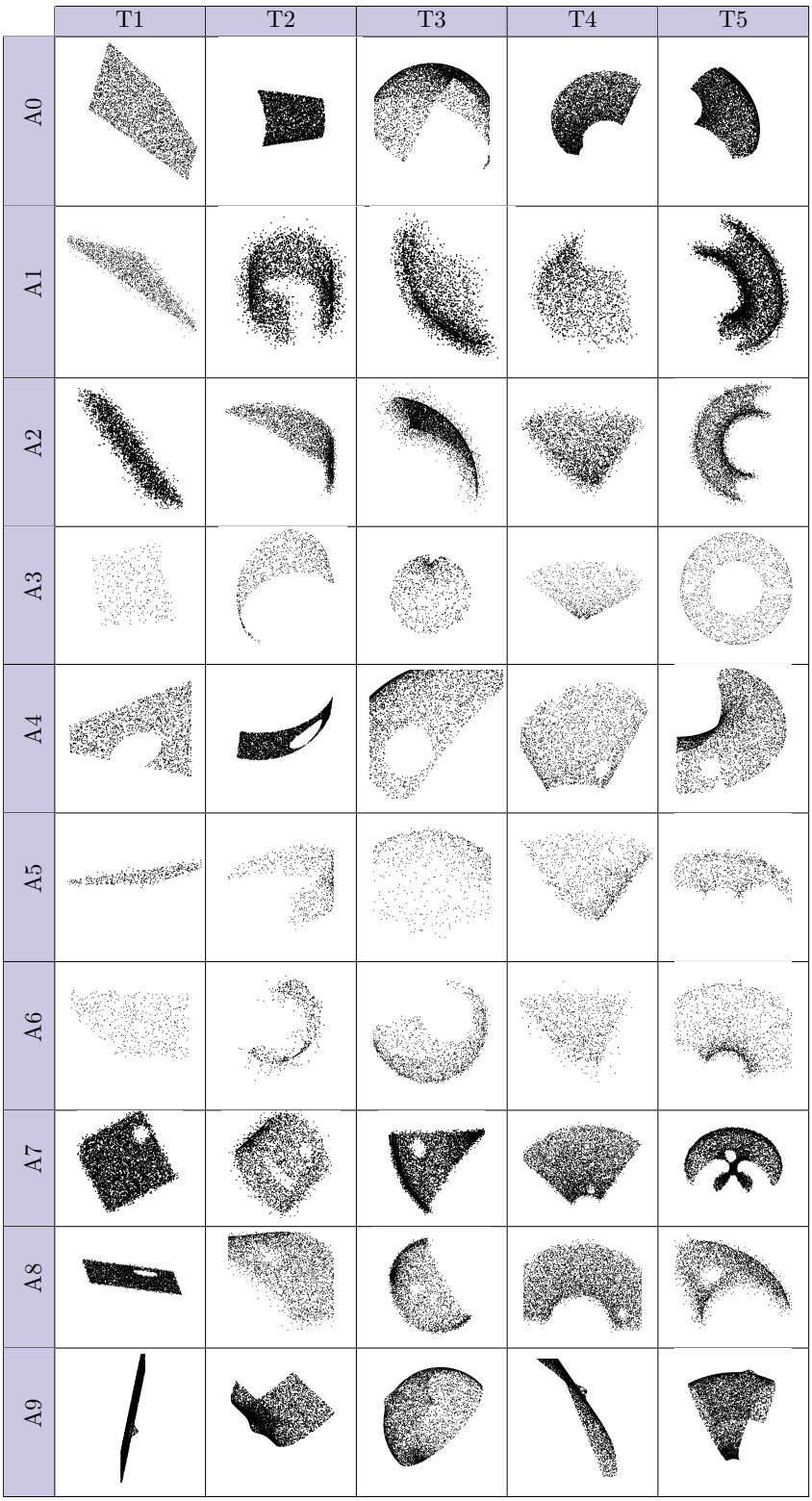


Figure 1: Examples of point clouds from the dataset. Columns identify primitive types: plane (T1), cylinder (T2), sphere (T3), cone (T4) and torus (T5). Rows correspond to point cloud artifacts: none (A0), uniform noise (A1), Gaussian noise (A2), undersampling (A3), missing data (A4), uniform noise + undersampling (A5), Gaussian noise + undersampling (A6), uniform noise + missing data (A7), Gaussian noise + missing data (A8), small deformations (A9).

- *Cylinder.* We list the primitive type (for cylinders, codified as 2), the radius r, the unit vector determining the rotational axis $\mathbf{a}$, and a point $\mathbf{P}$ sampled on the axis. In this case, the file contains the vector $\mathbf{v} = [2, r, \mathbf{a}^\mathbf{t}, \mathbf{P}^\mathbf{t}]^\mathbf{t}$.

- *Sphere.* We list the primitive type (for spheres, codified as 3), the radius r and the center $\mathbf{C}$. The ground truth corresponds to the vector $\mathbf{v} = [3,\, r,\, \mathbf{C}^{\mathsf{t}}]^{\mathsf{t}}$.

- *Cone.* We list the primitive type (for cones, codified as 4), half the aperture[2] α, the unit vector determining the rotational axis $\mathbf{a}$ and the vertex $\mathbf{C}$. Then, the file provides the vector $\mathbf{v} = [4,\, \alpha,\, \mathbf{a}^{\mathsf{t}},\, \mathbf{C}^{\mathsf{t}}]^{\mathsf{t}}$.

- *Torus.* We list the primitive type (for tori, codified as 5), the major and minor radii, respectively r and R, the unit vector determining the rotational axis $\mathbf{a}$ and the center $\mathbf{C}$. In this case, the file contains the vector $\mathbf{v} = [5,\, r,\, R,\, \mathbf{a}^{\mathsf{t}},\, \mathbf{C}^{\mathsf{t}}]^{\mathsf{t}}$.

For each point cloud, the track participants were required to return a TXT file per point cloud (of the test set), with the same syntax employed for the ground truth files of the training set.

2.2 Evaluation measures

Three tasks are considered in the evaluation phase:

- *Classification task.* Methods are evaluated on the basis of their capability for identifying, for each point cloud of the test set, its primitive type.

- *Recognition task.* Methods are studied in terms of their ability to recover the geometric descriptors (e.g, radii of a torus, vertex of a cone) behind each segment.

- *Fitting task.* Regardless of the primitive type, methods' performance is studied with respect to the approximation of each point cloud.

To quantify the performance of automatic algorithms in classifying, recognizing and fitting simple geometric primitives on point clouds under different conditions, we therefore consider a variety of classification and approximation measures. This analysis will be further specified according to the type of geometric primitive and point cloud artifacts.

2.2.1 Classification measures

The *confusion matrix* [10] is a popular way to visualize the classification performance of a method. A confusion matrix CM is a square matrix whose order equals the number of classes in the dataset under study. Diagonal elements count true positives, i.e., all those items which have been correctly labeled as members of their ground truth classes: precisely, $\mathrm{CM}(i,i)$ is the number of items that have been correctly predicted as elements of class i. Off-diagonal elements give the numbers of items mislabeled by the classifier; precisely, $\mathrm{CM}(i,j)$, with $j \neq i$, is the number of elements wrongly labeled as belonging to class j rather than to class i. Ideals classifiers have diagonal classification matrices.

True Positive and Negative Rates True Positive Rate (TPR) refers to the method's ability to correctly identify positives (e.g., the percentage of cats correctly classified as cats). Likewise, True Negative Rate (TNR) measures the classifier's ability to identify negatives (e.g., the percentage of non-cats correctly classified as non-cats). In statistics, TPR and TNR are also called *sensitivity* and *specificity*. A perfect classifier is 100% sensitive and 100% specific.

Positive and negative predicted values Besides TPR and TNR, to quantify the likelihood that a method returns a true positive (or true negative) rather than a false-positive (or a false-negative) we consider the Positive Predictive Value (PPV) and the Negative Predictive Value (NPV). PPV is the ratio of *true positives* to the number of items labeled as positives by the classifier. Similarly, NPV is the ratio of *true negatives* to the number of items classified as negatives.

[2]The aperture of a right circular cone, denoted by 2α, is the maximum angle between two generatrix lines; we here report half the aperture, i.e., α.

Accuracy Accuracy describes how often a classifier is correct in its predictions: precisely, it is defined as the proportion of predictions that the classifier got right.

Macro-average Macro-average permits to evaluate the overall performance of the classifier treating all classes equally, it is defined as the arithmetic mean of one of the metrics mentioned above, computed independently for each class.

2.2.2 Fitting and recognition measures

To measure the approximation accuracy of a specific primitive, we exploit the geometric descriptors predicted by each method and the corresponding parametric representation to sample densely the recognised surface primitive, say $\mathcal{S}_P$. Let us consider a point cloud $\mathcal{P}$ to be evaluated; we use the following two measures to evaluate the approximation accuracy:

- *Mean Fitting Error* (MFE):

$$\mathrm{MFE}(\mathcal{P}, \mathcal{S}_P) := \frac{1}{|\mathcal{P}|} \sum_{\mathbf{x} \in \mathcal{P}} d(\mathbf{x}, \mathcal{S}_P)/l, \tag{1}$$

  where d is the minimum Euclidean distance between $\mathbf{x}$ and every point in $\mathcal{S}_P$, and l is the diagonal of the minimum bounding box containing $\mathcal{P}$.

- *Directed Hausdorff distance*:

$$d_{\mathrm{dHaus}}(\mathcal{P}, \mathcal{S}_P) = \max_{\mathbf{x} \in \mathcal{P}} \min_{\mathbf{y} \in \mathcal{S}_P} d(\mathbf{x}, \mathbf{y}),$$

  with d the Euclidean distance.

Finally, the recognition accuracy is evaluated in terms of L^2 norm between the column vectors of the ground truth and that obtained by each method. Specifically, following the notation introduced in Section 2.1, we compare the geometric descriptors recognised with those of the ground truth considering the L^2 distance:

$$d(\mathbf{v}_G, \mathbf{v}_P) = \|\mathbf{v}_G - \mathbf{v}_P\|_2$$

where $\mathbf{v}_G$ and $\mathbf{v}_P$ here denote the ground truth and the prediction vectors, respectively, where we remove the first entry and, for planes and cylinders, the point (since they are not comparable).

3 Description of the methods

Ten groups initially subscribed to this track, with a final number of six methods submitted. In the remainder of this section we describe the methods, here simply referred to as M1, M2, ..., M6.

The six methods can be grouped as:

- Fully direct methods: accelerated Hough Transform (M1), and Histograms of local features + parameter fitting (M2).
- Fully neural-based methods: PointNet (M3) and 3D ShapeNets (M4).
- Mix of neural and direct methods: PoinNet + parameter fitting (M5); AlexNet + parameter fitting (M6).

3.1 M1: Accelerated Hough transform

This method is proposed by Chiara Romanengo, Andrea Raffo, Silvia Biasotti and Bianca Falcidieno. The method implementation can be found at `https://github.com/chiararomanengo/fitting_geometric_primitives.git` and it is introduced in [11].

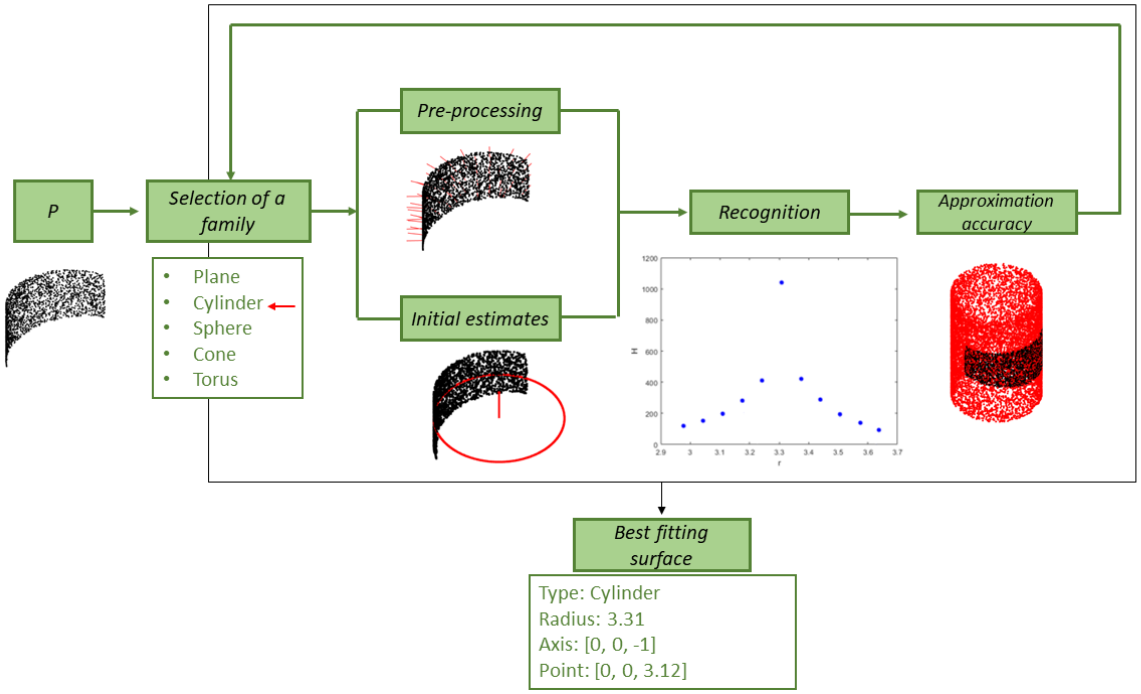


Figure 2: A graphical representation of the strategy adopted in method M1.

3.1.1 Overall strategy

The proposed approach is based on an accelerated version of the Hough transform, a well known technique in the pattern recognition community, which have been recently extended to algebraic objects, see [12]. The general (algebraic) HT-framework deals with the problem of finding a (hyper)surface, within a family of (hyper)surfaces dependent on a set of unknown parameters, that best approximates an input shape. More precisely, given an input point cloud and a family of parametrized primitives with some unknown parameters, the core idea is to consider the parameter space corresponding to the selected family of primitives, to discretize it into cells, and to vote a cell every time the Hough transform of an input point intersects it; the most voted cell uniquely identifies the optimal solution of the recognition problem.

Despite the robustness to various point cloud artifacts (e.g., noise, outliers, missing data and uneven sampling), a severe limitation of this paradigm is due to the curse of dimensionality; indeed, the dimension of the parameter space corresponds to the number of parameters that are meant to be estimated through the voting procedure. Only a few families of primitives can be considered in their general position, i.e., planes and spheres, but even for ellipsoids it is required that the cloud is centred in the origin of the Cartesian axes [13] to make the problem computationally affordable.

The proposed method takes in input a point cloud $\mathcal{P}$ and, for each family of primitives $\mathcal{F}$ (among planes, cylinders, spheres, cones and tori), it works in three main steps.

- *Pre-processing.* Independently from the primitive type, the normal vectors of the points are computed using the Hough transform technique and the family of planes in the Hesse normal form, as described in [14]. The adoption of a voting strategy makes this computation robust to most point cloud artifacts. Once the candidate tangent planes have been estimated, the process is specialized for each type of primitive. Specifically, the rotations and/or translations necessary to bring the point cloud in standard form are based on the geometric properties of the primitive type in question. The result of this process is a new point cloud $\mathcal{P}'$ and an initial estimates of the parameters of the primitive to be recognised. These parameters differ according to the type of family selected; for example in case of cylinders, tori or spheres, they correspond to the radii, while in case of cones they coincide with half the aperture.

- *HT-based surface recognition.* The new set of points $\mathcal{P}'$ is the input of the classical HT-based recognition algorithm. The estimates obtained in the pre-processing step allow to localize the search in a dimensionally-reduced parameter space. The method returns the parameters of the surface that best fits $\mathcal{P}'$ with respect to the given family of primitives. From these parameters, the geometric descriptors required by the track are obtained.

- *Evaluation of the approximation accuracy.* To evaluate the recognition accuracy of a specific primitive, the recognised surface is sampled from the parametric equation and the Mean Fitting Error as defined in Equation 1 is calculated between $\mathcal{P}'$ and the sampled surface. Finally, the initial roto-translation is applied backwards in order to obtain the geometric descriptors for the point cloud $\mathcal{P}$ in its original position.

The three steps here described are repeated for each family of surfaces (that is, planes, cylinders, spheres, cones and tori). Then, the primitive type and the geometric descriptors of the surface with the lowest MFE is returned by the method.

Figure 2 provides a graphical abstract of method M1.

3.1.2 Computational aspects

The tests were performed on a desktop equipped with an Intel Core i9 processor (at 3.6 GHz), in a Windows system 64 bits. The method was implemented in MATLAB. The execution time for each point cloud of the test set is, on average, 123 seconds.

3.2 M2: Histograms of local features and parameter fitting

This method is proposed by Chi-Bien Chu, Khoi-Nguyen Nguyen-Ngoc, Quang-Thuc Nguyen and Minh-Triet Tran. The method implementation can be found at `https://github.com/ccbien/SHREC22-Track2.git`.

3.2.1 Overall strategy

The proposed method pre-processes the input point cloud normalizing it in a unit cube. Then, the task is addressed with two main modules: a point cloud classification with $k-$nearest neighbours (k-NN) using a histogram of local features (Module 1), divided into several steps, and a shape parameter fitting (Module 2).

In the first step of Module 1, all objects are classified into two types: planar and non-planar segments. The latter are then classified into subcategories following two strategies: one considers cone and cylinder as two categories (Strategy 1), while the other on considers them as one category (Strategy 2). Both of them consider sphere and torus as one category. Finally, the primitive types are further specialized (to distinguish: tori from spheres for Strategy 1; cones, cylinders, tori and spheres for Strategy 2).

- *Module 1: object shape classification.* In the first step, both strategies search for all planes. The Histogram of Oriented Normal Vectors (HONV, see [15]) is used to represent a point cloud, specifically, a 2D histogram (25 x 25). Each cell with a value greater than or equal to 80% of the maximum value in the histogram is shifted to the center of the 2D histogram. In this way, a multiple 2D histogram representation of a single object is created and the $k-$NN, with $k = 3$, is used to classify planar and non-planar objects. If a point cloud is classified as plane, the PCA is applied to further verify its planar property and to remove false-positive cases.

  In the second step, the aim is to classify non-planar instances. Once the categories are defined accordingly to the strategy mentioned before, the HONV feature is used to represent the point cloud and the $k-$NN method is applied with $k = 5$.

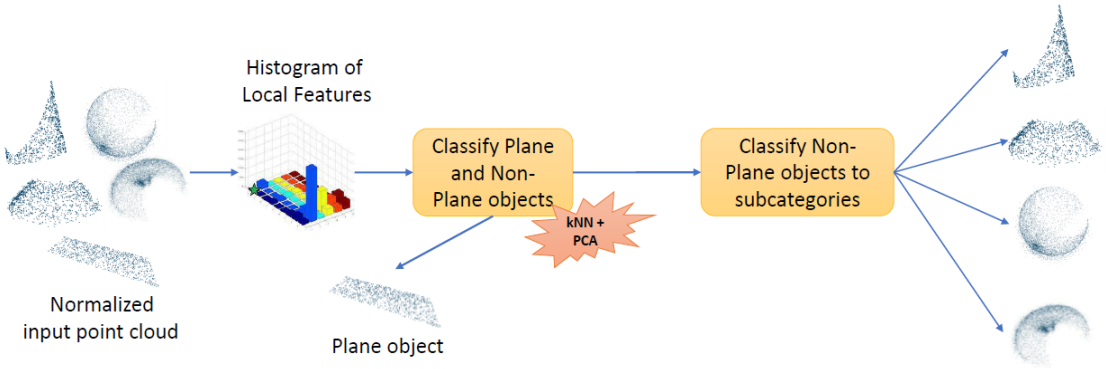


Figure 3: A graphical representation of the strategy adopted in method M2.

To classify sphere and torus, the Surflet-pair-relation histograms [16] are used in place of HONVs: in this way, each object can be represented with only one histogram. Finally, the $k-$NN with $k = 15$ is used to distinguish spheres from tori.

In Strategy 2, an additional step is required to classify cylinder and cone. In particular, the HONV feature is used to represent the point cloud and the $k-$NN with $k = 5$ is employed to differentiate between cylinder and cone.

- *Module 2: finding shape parameters.* For each primitive type, different approaches are considered to estimate and fine tune the shape parameters.
  - *Plane.* The average value of all the points considered to be on the surface is calculated. Then, PCA is used to find the eigenvector corresponding to the smallest eigenvalue, which is taken as approximation of the normal vector.
  - *Sphere.* A least squares sphere fit algorithm[3] is employed to find the center and radius of a sphere. In this case, only 30% of points are used and the process loops 100 times to find the parameters. Using the estimated parameter values, the number of points belonging to the surface of a sphere is computed, through the difference between the distance from the point to the center and the radius. If the difference is larger than 5% of the radius, that point is considered as an outlier. Finally, the set of parameter values with the most number of inliers is chosen.
  - *Cylinder.* A cylinder fitting approach[4] is used to find the rotational axis and the radius by minimizing the common least squares loss.
  - *Cone.* The RANSAC method proposed in [17] is followed to estimate the parameters for conic segments.
  - *Torus.* The parameter values are initialised: the center in the origin, the rotation axis equivalent to the $z-$axis, the main radius 1 and the minor radius 0.1. First, the points are fitted like a 3D circle (find center, radius, axis of rotation) by optimizing the L^2 loss function. Then, the minor radius is calculated by averaging the distance of that circle to the set of points.

Figure 3 provides a graphical abstract of method M2.

3.2.2 Computational aspects

The classification method needs at most 20 seconds on AMD Ryzen 5 3550H, when applying $k-$NN approach to classify all objects in each step of Strategy 1 and Strategy 2; this corresponds to an average of 0.02 seconds per model. Each point cloud might need multiple steps, and it takes at

[3] jekel.me/2015/Least-Squares-Sphere-Fit
[4] https://github.com/xingjiepan/cylinder_fitting

most 1 second to get the final primitive type prediction for an object. The shape parameter fitting methods take approximately 60 minutes on an Intel i5 10510U for all the 925 objects, which means about 3.89 seconds per model, on average.

3.3 M3: PointNet

This method is proposed by Ivan Sipiran. The method implementation can be found at `https://github.com/ivansipiran/shrec2022-geometric-primitives.git`.

3.3.1 Overall strategy

As for the previous method, M3 is based on a point cloud pre-processing, consisting of a centering and a scaling, and two consecutive steps: shape classification and parameter regression.

This method considers the PointNet [18] architecture as backbone for all networks. A first neural network is used to return the class probabilities for the five classes: plane, cylinder, sphere, cone, and torus.

Depending on the predicted class, each point cloud is fed into one of five specific networks to obtain the primitive parameters. Networks differ in the final layer, whose outputs must depend on the primitive type; the last layer of the backend consists of 256 linear neurons, which means that the input for the last layer of the regression is a 256-dimensional vector. Each primitive parameter is thus generated by a different MLP head which consists of a linear layer. For example, the plane network has two heads in the final layer: a 256×3 linear layer for the normal and a 256×3 linear layer for the point in plane parameter.

When it comes to the classification task, the loss function used to train the network is the cross-entropy loss between the one-hot activation functions from ground-truth and the network output. For the regression step, each primitive type has its own loss, defined as the sum of simpler functions:

- The *scalar loss* $\mathcal{L}_{x,\hat{x}}$ compares a ground truth scalar parameters x with its prediction $\hat{x}$, and is defined as the mean squared error. It can be used for radii and angles.
- The *point loss* $\mathcal{L}_{\mathbf{p},\hat{\mathbf{p}}}$ is defined as the Euclidean distance between ground-truth point $\mathbf{p}$ and output point $\hat{\mathbf{p}}$. It is used for points, e.g., for the sphere centers and the cone vertices.
- The *vector loss* controls the direction between a ground-truth vector $\mathbf{v}_G$ and the output vector $\mathbf{v}_O$, and is defined as follow:

  $$\mathcal{L}_{\mathbf{v}_G,\mathbf{v}_O} = 1 - \cos^8(\mathbf{v}_G, \mathbf{v}_O) + \|\mathbf{v}_G - \mathbf{v}_O\|_2^2.$$

  It is used for normal vectors and rotational axes. The L^2 regularization between $\mathbf{v}_G$ and $\mathbf{v}_O$ enforces the componentwise similarity between vectors. The exponent of the cosine function is meant to improve the learning of normal directions.

The total loss of each primitive type is defined by summing the above-defined losses:

$$\begin{aligned}
\mathcal{L}_{\text{plane}} &= \mathcal{L}_{\mathbf{p},\hat{\mathbf{p}}} + \mathcal{L}_{\mathbf{n},\hat{\mathbf{n}}} \\
\mathcal{L}_{\text{cylinder}} &= \mathcal{L}_{\mathbf{p},\hat{\mathbf{p}}} + \mathcal{L}_{\mathbf{a},\hat{\mathbf{a}}} + \mathcal{L}_{r,\hat{r}} \\
\mathcal{L}_{\text{sphere}} &= \mathcal{L}_{\mathbf{c},\hat{\mathbf{c}}} + \mathcal{L}_{r,\hat{r}} \\
\mathcal{L}_{\text{cone}} &= \mathcal{L}_{\mathbf{v},\hat{\mathbf{v}}} + \mathcal{L}_{r,\hat{r}} + \mathcal{L}_{\alpha,\hat{\alpha}} \\
\mathcal{L}_{\text{torus}} &= \mathcal{L}_{\mathbf{c},\hat{\mathbf{c}}} + \mathcal{L}_{\mathbf{a},\hat{\mathbf{a}}} + \mathcal{L}_{r_{\min},\hat{r}_{\min}} + \mathcal{L}_{r_{\max},\hat{r}_{\max}}
\end{aligned}$$

Finally, the parameters are transformed with respect to the inverse translations and scaling applied in the preprocessing.

The training is performed after splitting the original training set is split into two parts: a (sub-)training set (80%) and a validation set (20%).

Figure 4 provides a graphical abstract of method M3.

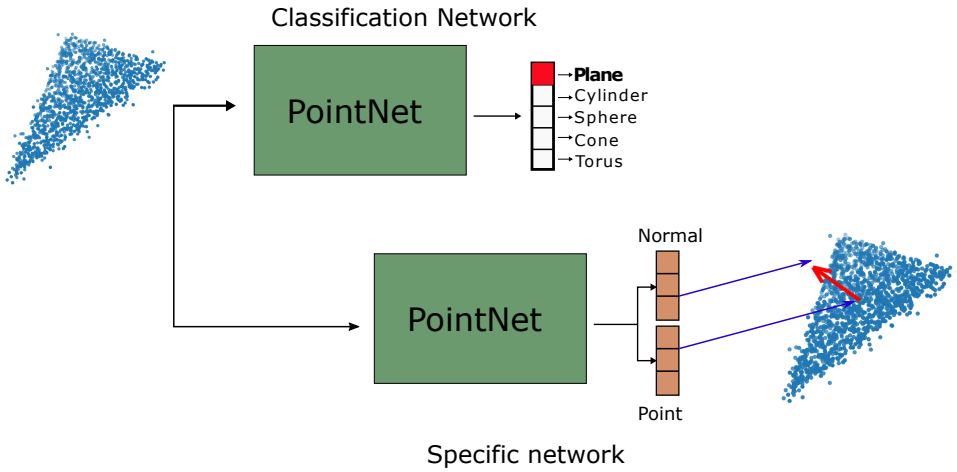


Figure 4: A graphical representation of the strategy adopted in method M3.

3.3.2 Computational aspects

This method was tested by using a 64-bit Linux OS, with 32GB RAM and a GPU NVIDIA RTX 2080 of 8GB. The method was tested with Python 3.6, by implementing NN architectures via Pytorch. All the training tasks were performed considering 200 epochs. The training time was about 4 hours for the classification step and approximately 6 hours for the regression part, which means an average of 72 and 108 seconds per epoch. The average prediction time, per point cloud, is 280 ms.

3.4 M4: 3D ShapeNets

This method is proposed by Vlassis Fotis, Ioannis Romanelis, Eleftheria Psatha and Konstantinos Moustakas. The method implementation can be found at `https://github.com/JohnRomanelis/SHREC2022_PrimitiveRecognition.git`.

3.4.1 Overall strategy

An initial preprocessing is applied to the input point clouds. First, they are translated to the origin, by subtracting the respective centroids. Following that, a normalization is applied, so that all points lie within the unit sphere. Random rotations about the x-, y- and z-axes are applied, for data augmentation purposes. The point clouds are finally converted into volumetric grids.

The backbone neural network for all models was originally proposed in the Minkowski Engine [19] as a candidate for the ModelNet classification challenge [20]. It is based on sparse voxel convolution, which minimizes the memory requirements associated with voxel based methods.

The network comprises of a voxel-wise input MLP and an input convolution. This is followed by three strided convolutions that progressively reduce the resolution of the voxel grid. After that, the output of the convolutional layers are projected on the initial grid, concatenated and finally passed through three strided convolutional layers. To extract the feature vector, Max and Mean pooling is applied to the resulting grid and the results are concatenated. All layers are followed by Batch Normalization and LeakyReLU activation function. For this track a 3-layer MLP head was used, with the number of output parameters changing according to the task.

M2's strategy involves two consequent steps:

- *Classification step.* An instance of the model described above, trained for classification, is tasked to find which primitive the input points where sampled from.
- *Fitting step.* Five more copies of the same model, each one trained on a specific type of primitive, are considered for the fitting task. During this step, the primitive type is known,

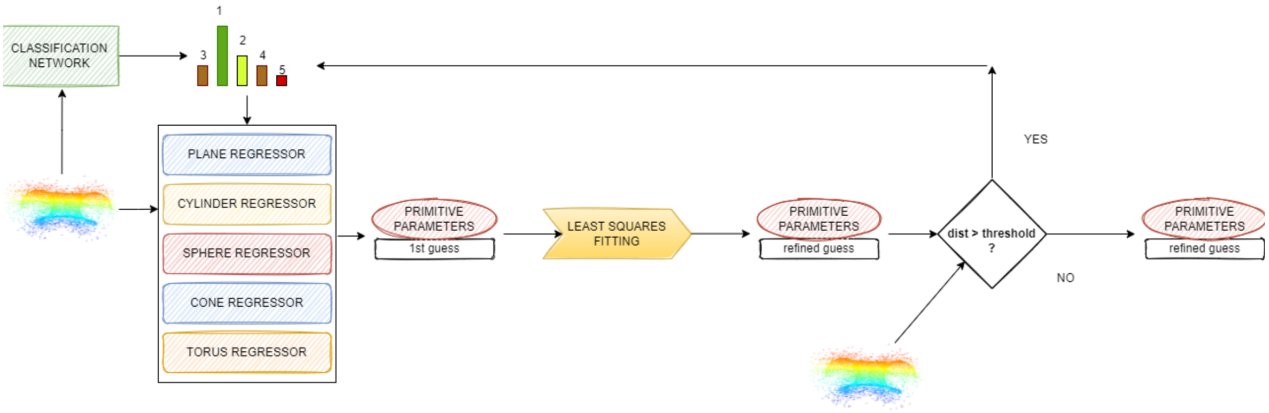


Figure 5: A graphical representation of the strategy adopted in method M4.

as it was predicted in the previous step. The model corresponding to that type of primitive is activated to regress the primitive's parameters. Although the network's performance is generally quite good, it can be increased even further by performing least-squares fitting. This is accomplished by using the network's prediction as an excellent initial guess, thereby making the fitting process very quick. For the least squares fitting, an existing python library `geomfitty`[5] is used and extended to include cones.

For the classification model, M2 minimizes the cross entropy loss. For the regression models, M4 uses a weighted combination of four basic losses, depending on the type of parameters to be predicted for a given primitive type:

- *Axis to axis loss.* This loss is applied to predict direction vectors. Then, given two such vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ the loss is defined as:

$$\mathcal{L}_1 = 1 - \frac{(\mathbf{v}_1 \cdot \mathbf{v}_2)^2}{\|\mathbf{v}_1\| \, \|\mathbf{v}_2\|}.$$

  Vectors are forced to have the same direction by minimizing the angle between them.

- *Point to axis loss.* This loss is used for anchor points that help to define the axes of various shapes. It is defined as the distance between the query point and the line that passes through the shape's axis.

- *Point to point loss.* This loss is used for the centers of spheres and tori. For a given point and a ground truth center it is defined as the Euclidean distance between them.

- *Scalar to scalar loss.* The remaining parameters such as radii and angles are simply scalars, so this loss is calculated via the mean squared error.

It is important to note that neural networks perform very poorly when tasked to predict numbers that are distributed over a wide range of values. The unit sphere normalization applied to the input helps with this problem, as it restricts said range to roughly $(0, 1)$ for most types of parameters. In order to map these values back to the correct range we apply the inverse transformations that we applied to the input points.

The network was trained and evaluated on a $85\% - 15\%$ training-validation split of the given dataset and was found to perform very well, reaching over 95% validation accuracy. To further improve the performance, a *backtracking strategy* is implemented to deal with heavily perturbed primitives. After the first shape's parameters are regressed, the distance between the shape and the input points is measured. If the distance exceeds a pre-defined threshold, then it is assumed that

[5] https://github.com/mark-boer/geomfitty

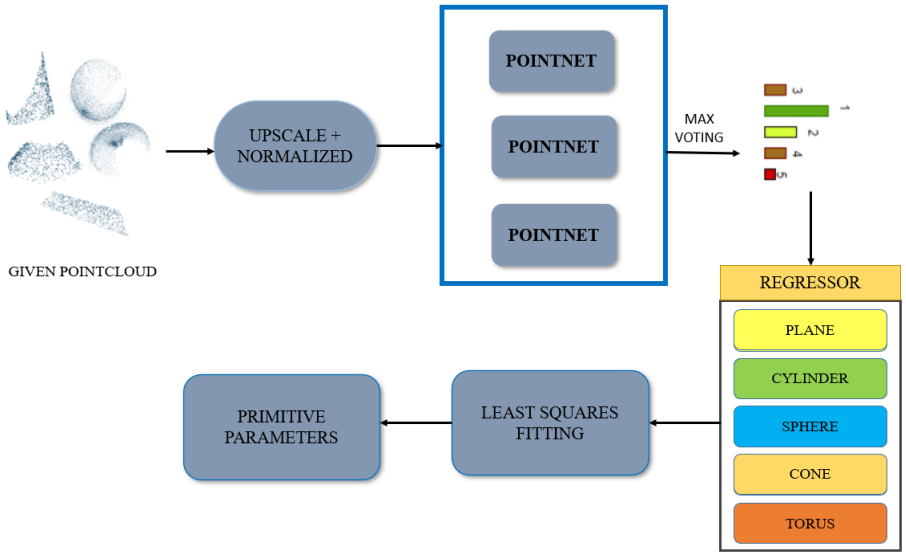


Figure 6: A graphical representation of the strategy adopted in method M5.

our initial classification guess was wrong, so the network's guess with the second highest probability is taken and the process is repeated. For peculiar cases where the least-squares fitting diverges, the regressor network's predicted parameters is used as the final estimation.

Figure 5 provides a graphical abstract of method M4.

3.4.2 Computational aspects

After experimenting on the training set, the voxel size was set to 0.05. For the minimization problem, M4 consider the optimization algorithm Adam [21]. The training of each model was performed on a machine equipped with a single Nvidia RTX 3090 and a Ryzen 7 3800x processor with 32GB of RAM. The classification network required 80 epochs to converge, while the regressor networks required 100 epochs. The average training times, per epoch, are: 123 seconds for the classification network, 14 seconds for the plane and cylinder regressors and 15, 16, 22 seconds for the sphere, cone and torus regressors respectively. The entire pipeline takes 39 seconds to complete processing the test set. No backtracking was required on the test set. The average execution times, per point cloud, for the classification network, regression network, and least squares fitting are 0.010, 0.009 and 0.021 seconds, respectively.

3.5 M5: PointNet bundle and parameter estimation with least square fitting

This method is proposed by Dinh-Khoi Vo, Tuan-An To, Nham-Tan Nguyen, Nhat-Quynh Le-Pham, Hai-Dang Nguyen and Minh-Triet Tran. The method implementation can be found at `https://github.com/ToTuanAn/SHREC2022.git`.

3.5.1 Overall strategy

The input point clouds are pre-processed; specifically, each segment is re-sampled in order to obtain the same number of points (8100 points), as the number of points in each object is not constant. This strategy augments the data by holding two of the three (x, y, z) position and adds a small shift in the remainder coordinate to create new data points. Finally, normalization and random rotations about the $z-$axis are applied to increase the effectiveness of the training process.

Similarly to M3, the core component for point cloud classification is the PointNet [18] architecture. This architecture has three main modules: the max pooling layer to aggregate information symmetrically from all the points symmetrically, a global and local feature combination structure

and two joint alignment networks. The symmetrical max pooling layer helps to make model invariant to the orders of the input data. Input points are fed in a collection of multi-layer perceptron networks to extract global features. After computing the global point cloud feature vector, data are fed back to concatenate the global feature with each of the point features. M5's strategy works in two main steps:

- *Classification step.* The data is trained by the above model with different parameters and the top three models are obtained with the highest accuracy in the validation test. The final label is returned after considering the majority votes from the models.
- *Shape parameter fitting.* For each primitive type, different ways are considered to estimate the shape parameters of each point cloud.
  - *Plane.* The regression with least mean square loss is used to fit a 3D plane on the input point cloud.
  - *Sphere.* The least mean square method is used to calculate the most appropriate sphere to the point cloud.
  - *Cylinder.* The least square fitting method is used to fit the point clouds to an infinitely long cylinder, parameterized by an anchor point, the direction and the radius of the cylinder.
  - *Cone.* Tangent planes are computed to approximate the vertex of the cone; then the circle that fits the set of points having the same distance from the vertex is used to find the rotational axis and half the aperture.
  - *Torus.* The `SciPy` library is used to determine the torus that best fits the point cloud.

  For the least squares fitting, the python library `geomfitty` is used (see method M4).

The networks are trained and evaluated by splitting the initial dataset in two parts, training set (80%) and validation set (20%); the validation accuracy is above 80%.

Figure 6 provides a graphical abstract of method M5.

3.5.2 Computational aspects

In the experiments, a learning rate of 2.5×10^{-4} is used, and Adam is used for optimization. The training is run on Google Colab Pro - GPU P100, for a total of 10 epochs. The average training times, per epoch, are: 50 minutes for the PointNet network and 8 min for all 3D geometry regressors. The execution time is, on average, 36 seconds for the classification of the 925 test segments, while it is 10 minutes to estimate their shape parameters.

3.6 M6: AlexNet and ISO reconstruction standards

This method is proposed by Yifan Qie and Nabil Anwer. The method implementation can be found at `https://github.com/YifanQie/SHREC22_fitting_LURPA.git`.

3.6.1 Overall strategy

Input point clouds are first preprocessed by centering them to the origin and scaling them into a unit sphere. Then, they are rotated so that the first principal axis of each point cloud lies along $z-$axis.

The primitive type identification of this method is based on AlexNet [22] which is a convolutional neural network architecture with 60 million parameters and 650,000 neurons. Segments are turned into images as input for training the neural network; images are collected from the principal directions for feature extraction and recognition. All the layers, except for the last three layers, are extracted form AlexNet. The updated last three layers consists of a fully connected layer, a softmax

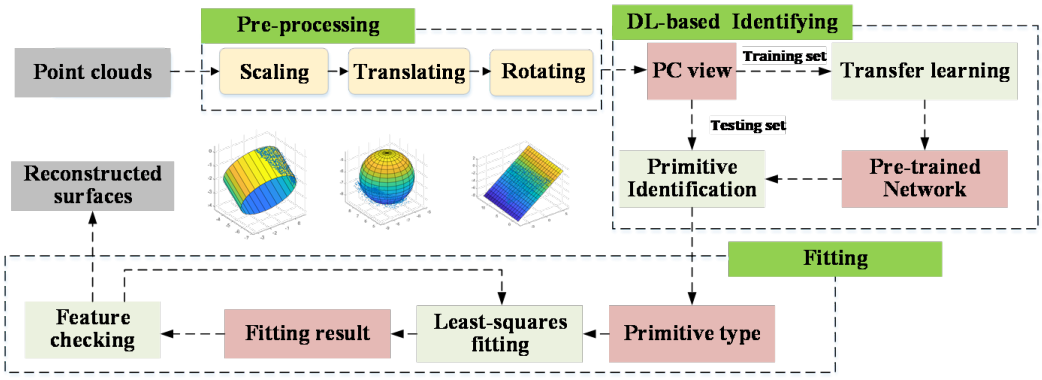


Figure 7: A graphical representation of the strategy adopted in method M6.

layer, and a classification output layer, where the fully connected layer is set to return as many numbers as the number of primitive types defined in contest (i.e., 5). Regarding the specificity of the input point clouds, transfer learning is used in the approach since the first layers in the pretrained networks have proved to be able to address simple pattern recognition in the images.

This strategy involves two main steps:

- *Classification step.* The pre-trained neural network is used for the test set to provide the primitive type that identify the input point cloud.
- *Fitting step.* The surface reconstruction is tested using the methods considered in the ISO standards on geometrical product specifications and verification (GPS), detailed in [23]. Once the primitive type of a point cloud, that is the invariance class in ISO GPS terminology, is identified by the fine-tuned AlexNet, an ideal feature defined by a parametrized equation can be guaranteed using least square fitting to obtain a reconstructed surface, see [24].

The network was trained and evaluated on a $95\% - 5\%$ training-validation split of the given dataset, reaching over 87% validation accuracy.

Figure 7 provides a graphical abstract of method M6.

3.6.2 Computational aspects

All tests are performed on a workstation equipped with a 1.70 GHz Intel(R) Xeon(R) CPU, an 16 GB RAM, and the Windows 10 operating system. The classification network required 6 epochs to converge. The training process takes about 50 minutes per epoch. The execution time for each model of the test set is about 0.5 second on average.

4 Comparative analysis

The performance of each method presented in Section 3 is here quantitatively evaluated on the basis of the measures described in Section 2.2.

For the sake of conciseness (and of the readers), we here report only tables referring to the whole test set; additional tables obtained by focusing on specific point cloud artifacts are provided as additional material in A and B.

4.1 Classification task

Figure 8 shows the confusion matrices over the whole test set. All the six methods have a high number of true positives, being cones and tori the most difficult primitive types to be recognized. A more thorough analysis is provided by Table 1, which contains the following information: Positive Predicted Value (PPV), Negative Predicted Value (NPV), True Positive Rate (TPR), True Negative Rate (TNR), Accuracy (ACC) and their macro-averages. We can notice that:

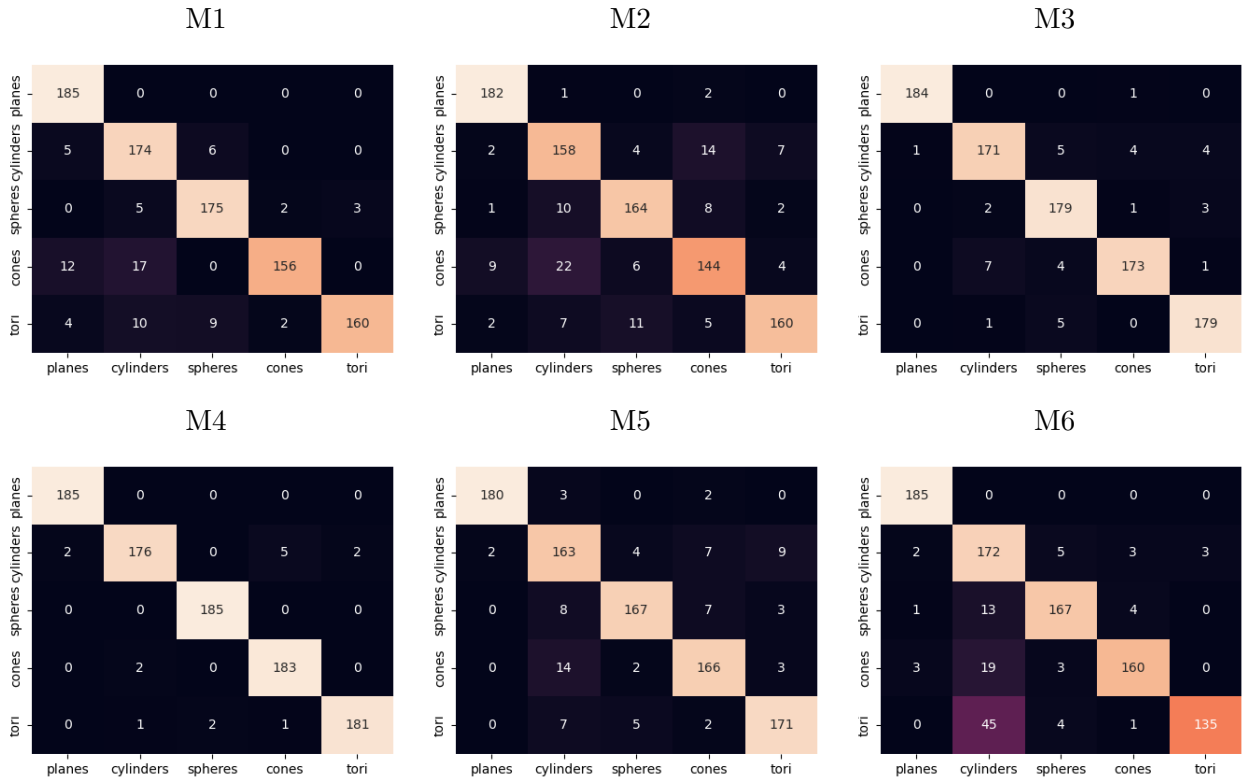


Figure 8: Confusion matrices CM for the whole test set, with respect to each primitive type. Here, the entry $\mathrm{CM}(i, j)$ indicates the number of samples with true label being the i-th class and predicted label being the j-th class.

- Methods M2, M3, M4 and M5 have NPV (slightly) higher than PPV: it is more likely for these methods to be right when reporting a negative rather than a positive. Methods M1 and M6 have mixed outcomes.
- Methods M3 and M5 have TNR higher than TPR, making them more reliable in correctly finding true negatives rather than true positives. Methods M1, M2, M4 and M6 presents, for some primitive types, a TPR higher than the TNR: this means that, in some cases, the classifier is more likely to identify correctly true positives rather than true negatives.
- All methods have the accuracy over 90%, with the slightly lower scores for cylinders, cones and tori. Macro-averaged values of accuracy are always above 95%, with two of the three best scores reached by neural methods.

Figure 9 clarifies the (global) impact of different point cloud artifacts upon the six methods, with respect to the classification measures. We observe that:

- M1, M2, M4 and M6 reach their best performance on clean data. Surprisingly enough, M2, M4 and M6 perform best on data with small deformations: these three approaches appear to be able to extract some additional hidden information to improve their prediction capability.
- M1 suffers the most from undersampled data. On the other hand, coherently with the Hough paradigm on which the method relies on, it is not much affected by missing data.
- M3 and M4 have their lowest performance, respectively, in case of uniform and Gaussian noise. However, the grouped bar charts suggest these two methods are the least prone to misclassification. This result confirms the power of neural methods in handling classification tasks.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8981	0.8447	0.9211	0.9750	0.9816	0.9241
	M2	0.9286	0.7980	0.8865	0.8324	0.9249	0.8741
	M3	0.9946	0.9448	0.9275	0.9665	0.9572	0.9581
	M4	0.9893	0.9832	0.9893	0.9683	0.9891	0.9838
	M5	0.9890	0.8359	0.9382	0.9022	0.9194	0.9169
	M6	0.9686	0.6908	0.9330	0.9524	0.9783	0.9046
NPV	M1	1.0000	0.9847	0.9864	0.9621	0.9672	0.9801
	M2	0.9959	0.9629	0.9716	0.9455	0.9668	0.9685
	M3	0.9986	0.9812	0.9918	0.9839	0.9919	0.9895
	M4	1.0000	0.9879	1.0000	0.9973	0.9946	0.9960
	M5	0.9933	0.9699	0.9759	0.9744	0.9811	0.9789
	M6	1.0000	0.9808	0.9759	0.9670	0.9365	0.9720
TPR	M1	1.0000	0.9405	0.9459	0.8432	0.8649	0.9189
	M2	0.9838	0.8541	0.8865	0.7784	0.8649	0.8735
	M3	0.9946	0.9243	0.9676	0.9351	0.9676	0.9578
	M4	1.0000	0.9514	1.0000	0.9892	0.9784	0.9838
	M5	0.9730	0.8811	0.9027	0.8973	0.9243	0.9157
	M6	1.0000	0.9297	0.9027	0.8649	0.7297	0.8854
TNR	M1	0.9716	0.9568	0.9797	0.9946	0.9959	0.9797
	M2	0.9811	0.9459	0.9716	0.9608	0.9824	0.9684
	M3	0.9986	0.9865	0.9811	0.9919	0.9892	0.9895
	M4	0.9973	0.9959	0.9973	0.9919	0.9973	0.9959
	M5	0.9973	0.9568	0.9851	0.9757	0.9797	0.9789
	M6	0.9919	0.8959	0.9838	0.9892	0.9959	0.9714
ACC	M1	0.9773	0.9535	0.9730	0.9643	0.9697	0.9676
	M2	0.9816	0.9276	0.9546	0.9243	0.9589	0.9494
	M3	0.9978	0.9741	0.9784	0.9805	0.9849	0.9831
	M4	0.9978	0.9870	0.9978	0.9914	0.9935	0.9935
	M5	0.9924	0.9416	0.9686	0.9600	0.9686	0.9663
	M6	0.9935	0.9027	0.9676	0.9643	0.9427	0.9542

Table 1: Classification metrics for the whole test set, with respect to each primitive type: T1=plane, T2=cylinder, T3=sphere, T4=cone, T5=torus. Here, gold, silver and bronze identify the first, second and third best performance. The last column contains the macro averages.

- In spite of having, in most cases, the lowest classification performance, M2 has still no measure that goes below 80%. Compared to the other method using PointNet (M3), here the implementation seems to overfit way more the training data.
- M5 has a relatively high performance on undersampled data, when compared to other point cloud artifacts.
- The combination of noise and undersampling has proved to be the most challenging perturbation for M6 which, however, exhibits one of the best overall performance on small deformations.

By checking which segments are misclassified, it can be additionally observed that the segments which results being more problematic are, in many cases, the small ones.

Tables 3-12 in A provide a complete overview of classification measures, specified with respect to the primitive type.

4.2 Recognition and fitting tasks

Table 2 provides various statistics of the measures introduced in Section 2.2.2, when the whole test set is considered. It contains the following information: first, second and third quartiles, mean value and standard deviation. More specifically, these quartiles split a set of sorted real numbers into four parts of approximately equal cardinality: the first (Q1) and the third (Q3) quartiles are defined as the values such that, respectively, 25% and 75% of the numbers lie below them; the second quartile (Q2) is the median of the set. Quartiles' significance is due to their ability to identify possible outliers. Based on quartiles, we can notice that:

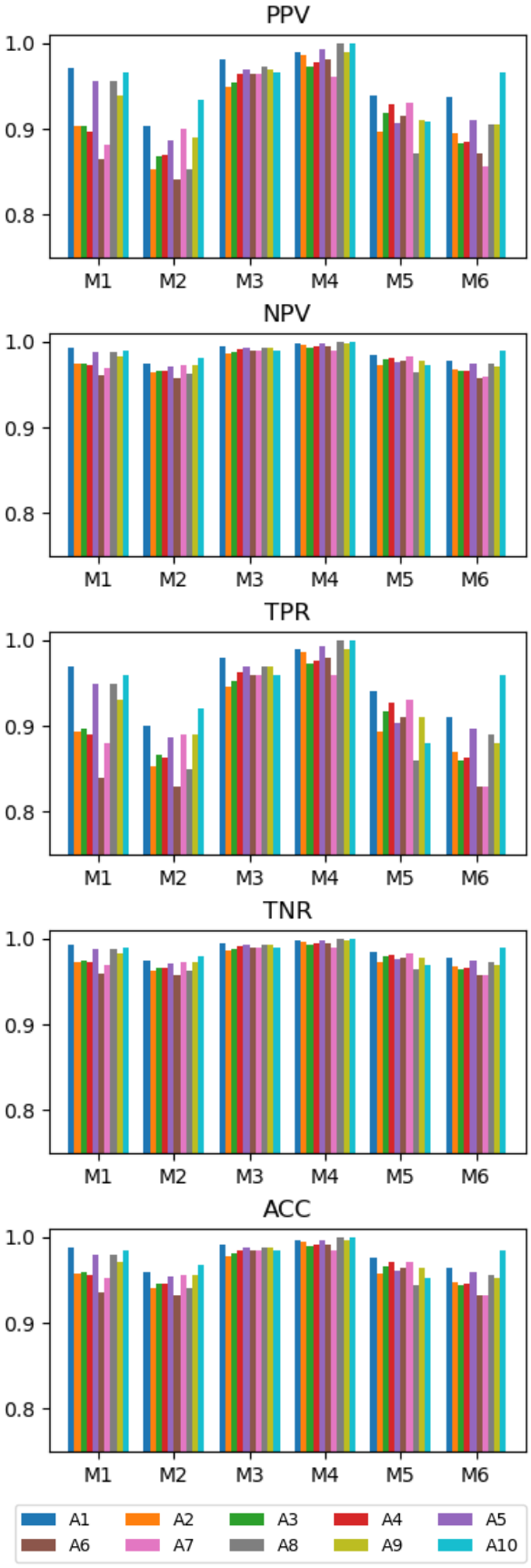


Figure 9: Bar graphs of the macro averages for the classification measures, grouped by method. The legend reflects the color encoding of the perturbation types.

- The L^2 distance provides low values for all methods with regard to the three quartiles. The lowest Q1 has an order of magnitude of -14 and is obtained by method M6; the highest order of magnitude is that of method M3, and corresponds to -1. Q2 and Q3 have order of magnitudes between -3 and -1.
- With the sole exception of the third quartile of M5, quartiles of the MFEs lie under the order of magnitude of -2. The lowest values are obtained by M2.
- When it comes to quartiles, the Hausdorff distance the order of magnitude is generally (non-strictly) lower than -1; the only exceptions fare the third quartiles of methods M3 and M5.

Based on the arithmetic mean and standard deviation, we can conclude that:

- Methods M2, M5 and M6 are affected by numerical instability issues, which result in the presence of outliers; it is worth noting, however, this problem is limited to the top 25% of the errors.
- M2 presents 11 outliers corresponding to the recognition of cones in which the estimated half-angle estimate is close to zero and the estimated vertex is far away from the origin of the coordinate axes.
- M5 provides different outliers depending on the type of task. Specifically, it has 61 outliers in the parameter recognition of various cylinders, cones and tori; however, for these clouds, fitting errors are low. We therefore conclude that, in all these cases, the approximation is successful but at the cost of a poor recognition. From the fitting task point of view, MFE and directed Hausdorff distance have high values in correspondence of 18 outliers correctly identified as planes: here, the normal is correctly identified (thus, the low value of L^2 distance); on the other hand, the method fails in returning a point passing through each returned plane.
- M6 presents 30 outliers, which correspond to point clouds classified as cones; the problem these estimates suffer from are analogue to that of M2.
- M1 and M4 alternate the first and second places in terms of means and standard deviations, suggesting that direct and neural methods can both provide competitive solutions to the problem under study.

By checking the bar graphs of the log mean errors with respect to the perturbation type, see Figure 10, it is possible to note a generally low variation of the (average) performance of each method, with the exception of M2 and M6; for these results, the logarithmic scale was preferred because of the presence of outliers. It is indicative that, when the methods (i.e., all but M2 and M6) are right in predicting the primitive type, their estimates are not much influenced by the perturbation type. Moreover:

- Methods M2 and M6 have better recognition and fitting performance when dealing with clean data or with point clouds having just small deformations. However, it is worth noticing once more that the poor performance mostly depends on the presence of outliers (see Table 2).
- M5 is on average better in fitting than in recognition: despite being the mean L^2 errors rather high, mean MFEs and directed Hausdorff distances are lower.
- M1 and M4 behave similarly in both recognition and fitting tasks.

As for classification measures, tables that specify recognition and fitting measures to the single point cloud artifacts are left to B (see Tables 13-22).

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.16e-02	6.82e-02	1.78e-01	2.06e-01	3.71e-01
	M2	8.95e-10	5.17e-03	9.27e-02	1.21e+11	2.78e+12
	M3	1.77e-01	4.73e-01	8.73e-01	6.03e-01	5.54e-01
	M4	2.37e-07	1.54e-02	4.43e-02	5.18e-02	1.84e-01
	M5	6.25e-04	2.15e-02	9.01e-01	2.46e+02	1.64e+03
	M6	1.49e-14	1.49e-02	1.05e-01	1.28e+06	7.54e+06
MFE	M1	3.71e-03	5.96e-03	1.05e-02	9.19e-03	9.42e-03
	M2	0.00	1.94e-03	5.11e-03	1.39e+10	3.51e+11
	M3	1.71e-02	4.49e-02	8.67e-02	6.57e-02	7.46e-02
	M4	2.32e-03	3.59e-03	5.87e-03	5.82e-03	1.43e-02
	M5	3.45e-03	1.49e-02	1.70e-01	2.62e+00	4.46e+01
	M6	2.04e-03	3.43e-03	9.51e-03	2.56e+05	1.46e+06
d_{dHaus}	M1	9.49e-02	1.67e-01	3.05e-01	2.50e-01	2.58e-01
	M2	6.12e-02	9.25e-02	2.68e-01	1.07e+11	2.60e+12
	M3	3.72e-01	6.62e-01	1.09e+00	8.13e-01	6.34e-01
	M4	5.82e-02	7.90e-02	1.51e-01	2.23e-01	4.74e-01
	M5	7.18e-02	2.98e-01	4.24e+00	3.87e+01	5.77e+02
	M6	5.85e-02	8.95e-02	2.53e-01	1.31e+06	7.63e+06

Table 2: Statistics of the fitting errors for the whole test set. Here, gold, silver and bronze identify the first, second and third best performance.

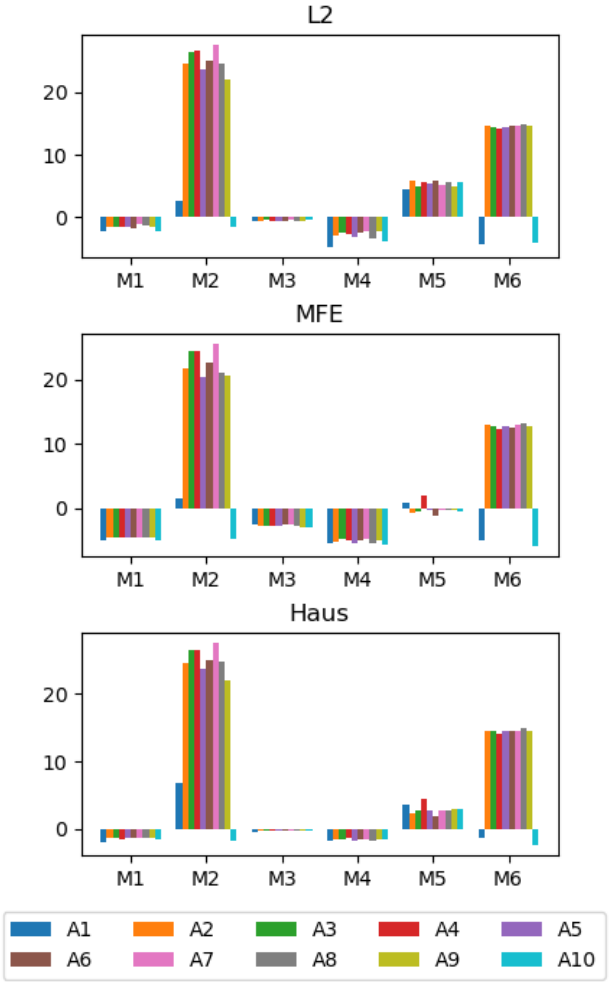


Figure 10: Bar graphs of the log macro averages for the recognition and fitting measures, grouped by method. The legend reflects the color encoding of the perturbation types.

5 Concluding remarks

Six methods were submitted for evaluation to this SHREC track on the fitting and recognition of simple geometric primitives on point clouds. Two approaches – M1 and M2 – base their whole pipeline on direct methods; M3 and M4 are fully based in neural networks; M5 and M6 exploits both methodologies, that is, neural architectures for the classification task and direct methods for the fitting and recognition tasks.

The dataset was specifically designed to allow the training of data-driven methods, with a training set of 46000 primitives and a test set of 925 models. Nine types of perturbations were randomly applied to each type of primitive in the ratio of 5000 training examples and 100 test samples, with the exception of local deformations that count 1000 training examples and 25 test samples. The outcome of the methods shows that, on simple primitives, data-driven methods are reaching a satisfactory level of maturity, obtaining the best scores both in primitive classification and in the parameter estimation. Direct methods remain fiercely competitive, and seem to suffer more in classification than in fitting and recognition. We believe that a similar comparative analysis will be possible in the future on models with multiple primitives.

As a possible future development, to better assess the potential of these methods in a reverse engineering context, it would be interesting to create a dataset with real data, e.g., captured with a high quality industrial 3D scanner, and derive the ground truth coming from the specifications used in the fabrication of the targets. To determine which approach (or which paradigm, between direct and neural methods) has the best generalization capability, it would be certainly worthwhile to keep the training set introduced in this track and use real data just in the test set. This direction is however rather challenging, as it requires having sufficient diversity, in terms of models to be scanned, between the primitives and parameters used for training and testing sets.

Acknowledgments

This work has been developed in the CNR IMATI research activities DIT.AD004.100, DIT.AD021.080.001 and DIT.AD021.125. Ivan Sipiran was funded by the Agencia Nacional de Investigación y Desarrollo (ANID Chile), under grant number 11220211. Nguyen Quang Thuc was funded by Vingroup JSC and supported by the Master, Ph.D. Scholarship Programme of Vingroup Innovation Foundation (VINIF), Institute of Big Data, code VINIF.2021.ThS.JVN.06. The teams from University of Science, VNU-HCM, were funded by Gia Lam Urban Development and Investment Company Limited, Vingroup and supported by Vingroup Innovation Foundation (VINIF) under project code VINIF.2019.DA19. Yifan Qie's work benefited from the financial support of China Scholarship Council (Yifan QIE), under Grant NO.201806020187.

References

[1] A. Kaiser, J. A. Ybanez Zepeda, and T. Boubekeur, "A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data," *Computer Graphics Forum*, vol. 38, no. 1, pp. 167–196, 2019.

[2] L. Li, M. Sung, A. Dubrovina, L. Yi, and L. J. Guibas, "Supervised Fitting of Geometric Primitives to 3D Point Clouds," in *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 2647–2655, 2019.

[3] A. Raffo, O. J. Barrowclough, and G. Muntingh, "Reverse engineering of CAD models via clustering and approximate implicitization," *Computer Aided Geometric Design*, vol. 80, p. 101876, 2020.

[4] G. Sharma, D. Liu, S. Maji, E. Kalogerakis, S. Chaudhuri, and R. Mech, "ParSeNet: A Parametric Surface Fitting Network for 3D Point Clouds," in *Computer Vision - ECCV 2020 - 16th European Conference, Glasgow, UK, August 23-28, 2020, Proceedings, Part VII* (A. Vedaldi,

H. Bischof, T. Brox, and J. Frahm, eds.), vol. 12352 of *Lecture Notes in Computer Science*, pp. 261–276, Springer, 2020.

[5] S. Yan, Z. Yang, C. Ma, H. Huang, E. Vouga, and Q.-X. Huang, "HPNet: Deep Primitive Segmentation Using Hybrid Representations," *2021 IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 2733–2742, 2021.

[6] T. Saporta and A. Sharf, "Unsupervised recursive deep fitting of 3D primitives to points," *Computers & Graphics*, vol. 102, pp. 289–299, 2022.

[7] S. Koch, A. Matveev, Z. Jiang, F. Williams, A. Artemov, E. Burnaev, M. Alexa, D. Zorin, and D. Panozzo, "ABC: A Big CAD Model Dataset For Geometric Deep Learning," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] C. Romanengo, A. Raffo, Y. Qie, N. Anwer, and B. Falcidieno, "Fit4CAD: A point cloud benchmark for fitting simple geometric primitives in CAD objects," *Computers & Graphics*, vol. 102, pp. 133–143, 2022.

[9] G. Sharma, R. Goyal, D. Liu, E. Kalogerakis, and S. Maji, "CSGNet: Neural Shape Parser for Constructive Solid Geometry," in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2018.

[10] K. Kuhn, Max;Johnson, *Applied Predictive Modeling*. Springer New York, 2018.

[11] A. Raffo, C. Romanengo, B. Falcidieno, and S. Biasotti, "Fitting and recognition of geometric primitives in segmented 3D point clouds using a localized voting procedure," *Computer Aided Geometric Design*, 2022.

[12] M. C. Beltrametti and L. Robbiano, "An algebraic approach to Hough transforms," *Journal of Algebra*, vol. 37, pp. 669–681, 2012.

[13] M. Beltrametti, J. Sendra, J. Sendra, and M. Torrente, "Moore–Penrose approach in the Hough transform framework," *Applied Mathematics and Computation*, vol. 375, p. 125083, 2020.

[14] F. A. Limberger and M. M. Oliveira, "Real-time detection of planar regions in unorganized point clouds," *Pattern Recognition*, vol. 48, no. 6, pp. 2043–2053, 2015.

[15] S. Tang, X. Wang, X. Lv, T. X. Han, J. Keller, Z. He, M. Skubic, and S. Lao, "Histogram of oriented normal vectors for object recognition with a depth sensor," *Computer Vision – ACCV 2012*, pp. 525–538, 2013.

[16] E. Wahl, U. Hillenbrand, and G. Hirzinger, "Surflet-pair-relation histograms: a statistical 3D-shape representation for rapid classification," in *Fourth International Conference on 3-D Digital Imaging and Modeling, 2003. 3DIM 2003. Proceedings.*, pp. 474–481, 2003.

[17] R. Schnabel, R. Wahl, and R. Klein, "Efficient RANSAC for point-cloud shape detection," *Comput. Graph. Forum*, vol. 26, pp. 214–226, 06 2007.

[18] R. Q. Charles, H. Su, M. Kaichun, and L. J. Guibas, "PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation," in *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 77–85, 2017.

[19] C. Choy, J. Gwak, and S. Savarese, "4D Spatio-Temporal ConvNets: Minkowski Convolutional Neural Networks," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[20] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, and J. Xiao, "3D ShapeNets: A Deep Representation for Volumetric Shapes," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[21] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings* (Y. Bengio and Y. LeCun, eds.), 2015.

[22] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "ImageNet Classification with Deep Convolutional Neural Networks," *Communications of the ACM*, vol. 60, pp. 84–90, may 2017.

[23] Y. Qie, N. Anwer, P. J. Scott, J. X. Jiang, and V. Srinivasan, "Toward a Mathematical Definition of Reconstruction Operation for ISO GPS Standards," *Procedia CIRP*, vol. 92, pp. 152–157, 2020.

[24] C. Shakarji, "Least-Squares Fitting Algorithms of the NIST Algorithm Testing System," *Journal of research of the National Institute of Standards and Technology*, 1998-12-01 1998.

A Classification metrics

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	1.0000	0.9524	0.9524	0.9500	1.0000	0.9710
	M2	0.9524	0.8333	1.0000	0.7826	0.9474	0.9031
	M3	1.0000	1.0000	0.9524	0.9500	1.0000	0.9805
	M4	1.0000	1.0000	1.0000	0.9524	1.0000	0.9905
	M5	1.0000	0.9048	0.9524	0.8947	0.9474	0.9398
	M6	1.0000	0.6897	1.0000	1.0000	1.0000	0.9379
NPV	M1	1.0000	1.0000	1.0000	0.9875	0.9756	0.9926
	M2	1.0000	0.9390	0.9877	0.9740	0.9753	0.9752
	M3	1.0000	0.9877	1.0000	0.9875	1.0000	0.9950
	M4	1.0000	0.9877	1.0000	1.0000	1.0000	0.9975
	M5	1.0000	0.9873	1.0000	0.9630	0.9753	0.9851
	M6	1.0000	1.0000	0.9877	0.9756	0.9302	0.9787
TPR	M1	1.0000	1.0000	1.0000	0.9500	0.9000	0.9700
	M2	1.0000	0.7500	0.9500	0.9000	0.9000	0.9000
	M3	1.0000	0.9500	1.0000	0.9500	1.0000	0.9800
	M4	1.0000	0.9500	1.0000	1.0000	1.0000	0.9900
	M5	1.0000	0.9500	1.0000	0.8500	0.9000	0.9400
	M6	1.0000	1.0000	0.9500	0.9000	0.7000	0.9100
TNR	M1	1.0000	0.9875	0.9875	0.9875	1.0000	0.9925
	M2	0.9875	0.9625	1.0000	0.9375	0.9875	0.9750
	M3	1.0000	1.0000	0.9875	0.9875	1.0000	0.9950
	M4	1.0000	1.0000	1.0000	0.9875	1.0000	0.9975
	M5	1.0000	0.9750	0.9875	0.9750	0.9875	0.9850
	M6	1.0000	0.8875	1.0000	1.0000	1.0000	0.9775
ACC	M1	1.0000	0.9900	0.9900	0.9800	0.9800	0.9880
	M2	0.9900	0.9200	0.9900	0.9300	0.9700	0.9600
	M3	1.0000	0.9900	0.9900	0.9800	1.0000	0.9920
	M4	1.0000	0.9900	1.0000	0.9900	1.0000	0.9960
	M5	1.0000	0.9700	0.9900	0.9500	0.9700	0.9760
	M6	1.0000	0.9100	0.9900	0.9800	0.9400	0.9640

Table 3: Classification metrics for clean segments.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8571	0.7941	0.8906	0.9792	1.0000	0.9042
	M2	0.8955	0.7812	0.8596	0.8113	0.9153	0.8526
	M3	0.9836	0.9483	0.9194	1.0000	0.8939	0.9490
	M4	0.9836	1.0000	1.0000	0.9677	0.9836	0.9870
	M5	0.9831	0.7761	0.9464	0.8947	0.8852	0.8971
	M6	0.9231	0.6667	0.9423	0.9455	1.0000	0.8955
NPV	M1	1.0000	0.9741	0.9873	0.9484	0.9600	0.9740
	M2	1.0000	0.9576	0.9547	0.9312	0.9751	0.9637
	M3	1.0000	0.9793	0.9874	0.9717	0.9957	0.9868
	M4	1.0000	0.9836	1.0000	1.0000	1.0000	0.9967
	M5	0.9917	0.9657	0.9713	0.9630	0.9749	0.9733
	M6	1.0000	0.9815	0.9556	0.9673	0.9375	0.9684
TPR	M1	1.0000	0.9000	0.9500	0.7833	0.8333	0.8933
	M2	1.0000	0.8333	0.8167	0.7167	0.9000	0.8533
	M3	1.0000	0.9167	0.9500	0.8833	0.9833	0.9467
	M4	1.0000	0.9333	1.0000	1.0000	1.0000	0.9867
	M5	0.9667	0.8667	0.8833	0.8500	0.9000	0.8933
	M6	1.0000	0.9333	0.8167	0.8667	0.7333	0.8700
TNR	M1	0.9583	0.9417	0.9708	0.9958	1.0000	0.9733
	M2	0.9708	0.9417	0.9667	0.9583	0.9792	0.9633
	M3	0.9958	0.9875	0.9792	1.0000	0.9708	0.9867
	M4	0.9958	1.0000	1.0000	0.9917	0.9958	0.9967
	M5	0.9958	0.9375	0.9875	0.9750	0.9708	0.9733
	M6	0.9792	0.8833	0.9875	0.9875	1.0000	0.9675
ACC	M1	0.9667	0.9333	0.9667	0.9533	0.9667	0.9573
	M2	0.9767	0.9200	0.9367	0.9100	0.9633	0.9413
	M3	0.9967	0.9733	0.9733	0.9767	0.9733	0.9787
	M4	0.9967	0.9867	1.0000	0.9933	0.9967	0.9947
	M5	0.9900	0.9233	0.9667	0.9500	0.9567	0.9573
	M6	0.9833	0.8933	0.9533	0.9633	0.9467	0.9480

Table 4: Classification metrics for segments suffering from uniform noise.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8451	0.8333	0.9167	0.9600	0.9623	0.9035
	M2	0.9344	0.8254	0.8209	0.8750	0.8868	0.8685
	M3	1.0000	0.9310	0.8939	0.9661	0.9828	0.9548
	M4	0.9836	0.9655	0.9677	0.9672	0.9828	0.9734
	M5	0.9828	0.8413	0.9444	0.9062	0.9180	0.9186
	M6	0.9836	0.6463	0.8871	0.9434	0.9524	0.8826
NPV	M1	1.0000	0.9786	0.9792	0.9520	0.9636	0.9747
	M2	0.9874	0.9662	0.9785	0.9549	0.9474	0.9669
	M3	0.9959	0.9752	0.9957	0.9876	0.9876	0.9884
	M4	1.0000	0.9835	1.0000	0.9958	0.9876	0.9934
	M5	0.9876	0.9705	0.9634	0.9915	0.9833	0.9793
	M6	1.0000	0.9679	0.9790	0.9595	0.9225	0.9658
TPR	M1	1.0000	0.9167	0.9167	0.8000	0.8500	0.8967
	M2	0.9500	0.8667	0.9167	0.8167	0.7833	0.8667
	M3	0.9833	0.9000	0.9833	0.9500	0.9500	0.9533
	M4	1.0000	0.9333	1.0000	0.9833	0.9500	0.9733
	M5	0.9500	0.8833	0.8500	0.9667	0.9333	0.9167
	M6	1.0000	0.8833	0.9167	0.8333	0.6667	0.8600
TNR	M1	0.9542	0.9542	0.9792	0.9917	0.9917	0.9742
	M2	0.9833	0.9542	0.9500	0.9708	0.9750	0.9667
	M3	1.0000	0.9833	0.9708	0.9917	0.9958	0.9883
	M4	0.9958	0.9917	0.9917	0.9917	0.9958	0.9933
	M5	0.9958	0.9583	0.9875	0.9750	0.9792	0.9792
	M6	0.9958	0.8792	0.9708	0.9875	0.9917	0.9650
ACC	M1	0.9633	0.9467	0.9667	0.9533	0.9633	0.9587
	M2	0.9767	0.9367	0.9433	0.9400	0.9367	0.9467
	M3	0.9967	0.9667	0.9733	0.9833	0.9867	0.9813
	M4	0.9967	0.9800	0.9933	0.9900	0.9867	0.9893
	M5	0.9867	0.9433	0.9600	0.9733	0.9700	0.9667
	M6	0.9967	0.8800	0.9600	0.9567	0.9267	0.9440

Table 5: Classification metrics for segments suffering from Gaussian noise.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8955	0.7910	0.8710	0.9623	0.9608	0.8961
	M2	0.9524	0.7361	0.8361	0.8431	0.9811	0.8698
	M3	1.0000	0.9355	0.9500	0.9828	0.9500	0.9636
	M4	0.9836	1.0000	0.9836	0.9375	0.9831	0.9776
	M5	1.0000	0.8594	0.9444	0.9322	0.9062	0.9285
	M6	0.9524	0.6463	0.9138	0.9623	0.9545	0.8859
NPV	M1	1.0000	0.9700	0.9748	0.9636	0.9558	0.9728
	M2	1.0000	0.9693	0.9623	0.9317	0.9676	0.9662
	M3	1.0000	0.9916	0.9875	0.9876	0.9875	0.9908
	M4	1.0000	0.9796	1.0000	1.0000	0.9917	0.9943
	M5	0.9959	0.9788	0.9634	0.9793	0.9915	0.9818
	M6	1.0000	0.9679	0.9711	0.9636	0.9297	0.9664
TPR	M1	1.0000	0.8833	0.9000	0.8500	0.8167	0.8900
	M2	1.0000	0.8833	0.8500	0.7167	0.8667	0.8633
	M3	1.0000	0.9667	0.9500	0.9500	0.9500	0.9633
	M4	1.0000	0.9167	1.0000	1.0000	0.9667	0.9767
	M5	0.9833	0.9167	0.8500	0.9167	0.9667	0.9267
	M6	1.0000	0.8833	0.8833	0.8500	0.7000	0.8633
TNR	M1	0.9708	0.9417	0.9667	0.9917	0.9917	0.9725
	M2	0.9875	0.9208	0.9583	0.9667	0.9958	0.9658
	M3	1.0000	0.9833	0.9875	0.9958	0.9875	0.9908
	M4	0.9958	1.0000	0.9958	0.9833	0.9958	0.9942
	M5	1.0000	0.9625	0.9875	0.9833	0.9750	0.9817
	M6	0.9875	0.8792	0.9792	0.9917	0.9917	0.9658
ACC	M1	0.9767	0.9300	0.9533	0.9633	0.9567	0.9560
	M2	0.9900	0.9133	0.9367	0.9167	0.9700	0.9453
	M3	1.0000	0.980	0.9800	0.9867	0.9800	0.9853
	M4	0.9967	0.9833	0.9967	0.9867	0.9900	0.9907
	M5	0.9967	0.9533	0.9600	0.9700	0.9733	0.9707
	M6	0.9900	0.8800	0.9600	0.9633	0.9333	0.9453

Table 6: Classification metrics for segments suffering from undersampling.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.9231	0.8551	1.0000	1.0000	1.0000	0.9556
	M2	0.9077	0.8500	0.9455	0.8167	0.9167	0.8873
	M3	1.0000	0.9344	0.9831	0.9661	0.9672	0.9702
	M4	1.0000	0.9836	0.9836	1.0000	1.0000	0.9934
	M5	0.9828	0.7846	0.9194	0.8983	0.9464	0.9063
	M6	0.9677	0.7308	0.9483	0.9298	0.9778	0.9109
NPV	M1	1.0000	0.9957	1.0000	0.9600	0.9836	0.9879
	M2	0.9957	0.9625	0.9673	0.9542	0.9792	0.9718
	M3	1.0000	0.9874	0.9917	0.9876	0.9958	0.9925
	M4	1.0000	1.0000	1.0000	0.9959	0.9959	0.9983
	M5	0.9876	0.9617	0.9874	0.9710	0.9713	0.9758
	M6	1.0000	0.9865	0.9793	0.9712	0.9373	0.9749
TPR	M1	1.0000	0.9833	1.0000	0.8333	0.9333	0.9500
	M2	0.9833	0.8500	0.8667	0.8167	0.9167	0.8867
	M3	1.0000	0.9500	0.9667	0.9500	0.9833	0.9700
	M4	1.0000	1.0000	1.0000	0.9833	0.9833	0.9933
	M5	0.9500	0.8500	0.9500	0.8833	0.8833	0.9033
	M6	1.0000	0.9500	0.9167	0.8833	0.7333	0.8967
TNR	M1	0.9792	0.9583	1.0000	1.0000	1.0000	0.9875
	M2	0.9750	0.9625	0.9875	0.9542	0.9792	0.9717
	M3	1.0000	0.9833	0.9958	0.9917	0.9917	0.9925
	M4	1.0000	0.9958	0.9958	1.0000	1.0000	0.9983
	M5	0.9958	0.9417	0.9792	0.9750	0.9875	0.9758
	M6	0.9917	0.9125	0.9875	0.9833	0.9958	0.9742
ACC	M1	0.9833	0.9633	1.0000	0.9667	0.9867	0.9800
	M2	0.9767	0.9400	0.9633	0.9267	0.9667	0.9547
	M3	1.0000	0.9767	0.9900	0.9833	0.9900	0.9880
	M4	1.0000	0.9967	0.9967	0.9967	0.9967	0.9973
	M5	0.9867	0.9233	0.9733	0.9567	0.9667	0.9613
	M6	0.9933	0.9200	0.9733	0.9633	0.9433	0.9587

Table 7: Classification metrics for segments suffering from missing data.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8000	0.6957	0.8261	1.0000	1.0000	0.8643
	M2	0.9524	0.6400	0.7895	0.8235	1.0000	0.8411
	M3	1.0000	0.9500	1.0000	1.0000	0.8696	0.9639
	M4	1.0000	1.0000	1.0000	0.9091	1.0000	0.9818
	M5	1.0000	0.7826	0.9412	0.9474	0.9091	0.9160
	M6	0.9091	0.5667	0.9375	0.9444	1.0000	0.8715
NPV	M1	1.0000	0.9481	0.9870	0.9412	0.9302	0.9613
	M2	1.0000	0.9467	0.9383	0.9277	0.9756	0.9577
	M3	1.0000	0.9875	0.9877	0.9756	1.0000	0.9902
	M4	1.0000	0.9756	1.0000	1.0000	1.0000	0.9951
	M5	0.9877	0.9740	0.9518	0.9753	1.0000	0.9778
	M6	1.0000	0.9571	0.9405	0.9634	0.9302	0.9583
TPR	M1	1.0000	0.8000	0.9500	0.7500	0.7000	0.8400
	M2	1.0000	0.8000	0.7500	0.7000	0.9000	0.8300
	M3	1.0000	0.9500	0.9500	0.9000	1.0000	0.9600
	M4	1.0000	0.9000	1.0000	1.0000	1.0000	0.9800
	M5	0.9500	0.9000	0.8000	0.9000	1.0000	0.9100
	M6	1.0000	0.8500	0.7500	0.8500	0.7000	0.8300
TNR	M1	0.9375	0.9125	0.9500	1.0000	1.0000	0.9600
	M2	0.9875	0.8875	0.9500	0.9625	1.0000	0.9575
	M3	1.0000	0.9875	1.0000	1.0000	0.9625	0.9900
	M4	1.0000	1.0000	1.0000	0.9750	1.0000	0.9950
	M5	1.0000	0.9375	0.9875	0.9875	0.9750	0.9775
	M6	0.9750	0.8375	0.9875	0.9875	1.0000	0.9575
ACC	M1	0.9500	0.8900	0.9500	0.9500	0.9400	0.9360
	M2	0.9900	0.8700	0.9100	0.9100	0.9800	0.9320
	M3	1.0000	0.9800	0.9900	0.9800	0.9700	0.9840
	M4	1.0000	0.9800	1.0000	0.9800	1.0000	0.9920
	M5	0.9900	0.9300	0.9500	0.9700	0.9800	0.9640
	M6	0.9800	0.8400	0.9400	0.9600	0.9400	0.9320

Table 8: Classification metrics for segments suffering from uniform noise and undersampling.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.9091	0.8095	0.8500	0.8947	0.9444	0.8816
	M2	0.9524	0.8571	0.7500	0.9412	1.0000	0.9001
	M3	1.0000	0.9500	0.8696	1.0000	1.0000	0.9639
	M4	0.9524	1.0000	0.9524	0.9524	0.9500	0.9614
	M5	1.0000	0.9048	0.9444	0.9048	0.9000	0.9308
	M6	0.9524	0.6071	0.8571	0.9444	0.9167	0.8556
NPV	M1	1.0000	0.9620	0.9625	0.9630	0.9634	0.9702
	M2	1.0000	0.9747	0.9737	0.9518	0.9639	0.9728
	M3	1.0000	0.9875	1.0000	0.9877	0.9756	0.9902
	M4	1.0000	0.9639	1.0000	1.0000	0.9875	0.9903
	M5	1.0000	0.9873	0.9634	0.9873	0.9750	0.9826
	M6	1.0000	0.9583	0.9747	0.9634	0.8977	0.9588
TPR	M1	1.0000	0.8500	0.8500	0.8500	0.8500	0.8800
	M2	1.0000	0.9000	0.9000	0.8000	0.8500	0.8900
	M3	1.0000	0.9500	1.0000	0.9500	0.9000	0.9600
	M4	1.0000	0.8500	1.0000	1.0000	0.9500	0.9600
	M5	1.0000	0.9500	0.8500	0.9500	0.9000	0.9300
	M6	1.0000	0.8500	0.9000	0.8500	0.5500	0.8300
TNR	M1	0.9750	0.9500	0.9625	0.9750	0.9875	0.9700
	M2	0.9875	0.9625	0.9250	0.9875	1.0000	0.9725
	M3	1.0000	0.9875	0.9625	1.0000	1.0000	0.9900
	M4	0.9875	1.0000	0.9875	0.9875	0.9875	0.9900
	M5	1.0000	0.9750	0.9875	0.9750	0.9750	0.9825
	M6	0.9875	0.8625	0.9625	0.9875	0.9875	0.9575
ACC	M1	0.9800	0.9300	0.9400	0.9500	0.9600	0.9520
	M2	0.9900	0.9500	0.9200	0.9500	0.9700	0.9560
	M3	1.0000	0.9800	0.9700	0.9900	0.9800	0.9840
	M4	0.9900	0.9700	0.9900	0.9900	0.9800	0.9840
	M5	1.0000	0.9700	0.9600	0.9700	0.9600	0.9720
	M6	0.9900	0.8600	0.9500	0.9600	0.9000	0.9320

Table 9: Classification metrics for segments suffering from Gaussian noise and undersampling.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.9091	0.8696	1.0000	1.0000	1.0000	0.9557
	M2	0.8696	0.8421	0.9412	0.7143	0.9000	0.8534
	M3	1.0000	0.9091	1.0000	1.0000	0.9524	0.9723
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	0.6800	0.9500	0.8421	0.8824	0.8709
	M6	0.9091	0.7692	0.9474	0.9000	1.0000	0.9051
NPV	M1	1.0000	1.0000	1.0000	0.9524	0.9877	0.9880
	M2	1.0000	0.9506	0.9518	0.9367	0.9750	0.9628
	M3	1.0000	1.0000	0.9877	0.9756	1.0000	0.9927
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	0.9877	0.9600	0.9875	0.9506	0.9398	0.9651
	M6	1.0000	1.0000	0.9753	0.9750	0.9195	0.9740
TPR	M1	1.0000	1.0000	1.0000	0.8000	0.9500	0.9500
	M2	1.0000	0.8000	0.8000	0.7500	0.9000	0.8500
	M3	1.0000	1.0000	0.9500	0.9000	1.0000	0.9700
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	0.9500	0.8500	0.9500	0.8000	0.7500	0.8600
	M6	1.0000	1.0000	0.9000	0.9000	0.6500	0.8900
TNR	M1	0.9750	0.9625	1.0000	1.0000	1.0000	0.9875
	M2	0.9625	0.9625	0.9875	0.9250	0.9750	0.9625
	M3	1.0000	0.9750	1.0000	1.0000	0.9875	0.9925
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	0.9000	0.9875	0.9625	0.9750	0.9650
	M6	0.9750	0.9250	0.9875	0.9750	1.0000	0.9725
ACC	M1	0.9800	0.9700	1.0000	0.9600	0.9900	0.9800
	M2	0.9700	0.9300	0.9500	0.8900	0.9600	0.9400
	M3	1.0000	0.9800	0.9900	0.9800	0.9900	0.9880
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	0.9900	0.8900	0.9800	0.9300	0.9300	0.9440
	M6	0.9800	0.9400	0.9700	0.9600	0.9300	0.9560

Table 10: Classification metrics for segments suffering from uniform noise and missing data.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	0.8696	0.8261	1.0000	1.0000	1.0000	0.9391
	M2	0.9048	0.9000	0.9000	0.8947	0.8500	0.8899
	M3	1.0000	0.9500	0.9500	1.0000	0.9500	0.9700
	M4	1.0000	1.0000	0.9524	1.0000	1.0000	0.9905
	M5	0.9474	0.8421	0.9000	0.9091	0.9500	0.9097
	M6	1.0000	0.6429	0.9474	1.0000	0.9375	0.9055
NPV	M1	1.0000	0.9870	1.0000	0.9524	0.9756	0.9830
	M2	0.9873	0.9750	0.9750	0.9630	0.9625	0.9726
	M3	1.0000	0.9875	0.9875	1.0000	0.9875	0.9925
	M4	1.0000	1.0000	1.0000	1.0000	0.9877	0.9975
	M5	0.9753	0.9506	0.9750	1.0000	0.9875	0.9777
	M6	1.0000	0.9722	0.9753	0.9639	0.9405	0.9704
TPR	M1	1.0000	0.9500	1.0000	0.8000	0.9000	0.9300
	M2	0.9500	0.9000	0.9000	0.8500	0.8500	0.8900
	M3	1.0000	0.9500	0.9500	1.0000	0.9500	0.9700
	M4	1.0000	1.0000	1.0000	1.0000	0.9500	0.9900
	M5	0.9000	0.8000	0.9000	1.0000	0.9500	0.9100
	M6	1.0000	0.9000	0.9000	0.8500	0.7500	0.8800
TNR	M1	0.9625	0.9500	1.0000	1.0000	1.0000	0.9825
	M2	0.9750	0.9750	0.9750	0.9750	0.9625	0.9725
	M3	1.0000	0.9875	0.9875	1.0000	0.9875	0.9925
	M4	1.0000	1.0000	0.9875	1.0000	1.0000	0.9975
	M5	0.9875	0.9625	0.9750	0.9750	0.9875	0.9775
	M6	1.0000	0.8750	0.9875	1.0000	0.9875	0.9700
ACC	M1	0.9700	0.9500	1.0000	0.9600	0.9800	0.9720
	M2	0.9700	0.9600	0.9600	0.9500	0.9400	0.9560
	M3	1.0000	0.9800	0.9800	1.0000	0.9800	0.9880
	M4	1.0000	1.0000	0.9900	1.0000	0.9900	0.9960
	M5	0.9700	0.9300	0.9600	0.9800	0.9800	0.9640
	M6	1.0000	0.8800	0.9700	0.9700	0.9400	0.9520

Table 11: Classification metrics for segments suffering from Gaussian noise and missing data.

	Method	T1	T2	T3	T4	T5	Avg
PPV	M1	1.0000	1.0000	0.8333	1.0000	1.0000	0.9667
	M2	1.0000	0.8333	0.8333	1.0000	1.0000	0.9333
	M3	1.0000	1.0000	0.8333	1.0000	1.0000	0.9667
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	1.0000	0.8333	0.7143	1.0000	0.9095
	M6	1.0000	0.8333	1.0000	1.0000	1.0000	0.9667
NPV	M1	1.0000	1.0000	1.0000	1.0000	0.9524	0.9905
	M2	1.0000	1.0000	1.0000	0.9524	0.9524	0.9810
	M3	1.0000	1.0000	1.0000	1.0000	0.9524	0.9905
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	0.9091	1.0000	1.0000	0.9524	0.9723
	M6	1.0000	1.0000	1.0000	1.0000	0.9524	0.9905
TPR	M1	1.0000	1.0000	1.0000	1.0000	0.8000	0.9600
	M2	1.0000	1.0000	1.0000	0.8000	0.8000	0.9200
	M3	1.0000	1.0000	1.0000	1.0000	0.8000	0.9600
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	0.6000	1.0000	1.0000	0.8000	0.8800
	M6	1.0000	1.0000	1.0000	1.0000	0.8000	0.9600
TNR	M1	1.0000	1.0000	0.9500	1.0000	1.0000	0.9900
	M2	1.0000	0.9500	0.9500	1.0000	1.0000	0.9800
	M3	1.0000	1.0000	0.9500	1.0000	1.0000	0.9900
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	1.0000	0.9500	0.9000	1.0000	0.9700
	M6	1.0000	0.9500	1.0000	1.0000	1.0000	0.9900
ACC	M1	1.0000	1.0000	0.9600	1.0000	0.9600	0.9840
	M2	1.0000	0.9600	0.9600	0.9600	0.9600	0.9680
	M3	1.0000	1.0000	0.9600	1.0000	0.9600	0.9840
	M4	1.0000	1.0000	1.0000	1.0000	1.0000	1.0000
	M5	1.0000	0.9200	0.9600	0.9200	0.9600	0.9520
	M6	1.0000	0.9600	1.0000	1.0000	0.9600	0.9840

Table 12: Classification metrics for segments suffering from local deformations.

B Fitting metrics

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	1.73e-02	4.10e-02	1.23e-01	1.24e-01	2.10e-01
	M2	4.11e-10	5.80e-10	8.86e-10	1.46e+01	1.35e+02
	M3	1.24e-01	4.14e-01	6.92e-01	5.19e-01	5.11e-01
	M4	8.20e-09	2.86e-08	1.55e-07	8.72e-03	2.07e-02
	M5	4.41e-15	1.36e-13	1.81e-02	9.58e+01	4.12e+02
	M6	6.30e-15	2.86e-13	1.64e-05	1.26e-02	3.79e-02
MFE	M1	3.22e-03	4.99e-03	8.03e-03	6.35e-03	5.29e-03
	M2	0.00	1.16e-03	2.57e-03	4.53e+00	3.20e+01
	M3	1.68e-02	4.45e-02	7.74e-02	7.25e-02	9.88e-02
	M4	1.87e-03	2.56e-03	4.72e-03	4.37e-03	8.76e-03
	M5	2.22e-03	5.31e-03	1.35e-01	2.16e+00	1.45e+01
	M6	1.48e-03	2.21e-03	4.40e-03	7.31e-03	2.07e-02
d_{dHaus}	M1	7.44e-02	1.16e-01	1.95e-01	1.56e-01	1.41e-01
	M2	3.60e-02	5.69e-02	7.92e-02	9.86e+02	9.44e+03
	M3	3.17e-01	5.98e-01	9.64e-01	7.14e-01	5.06e-01
	M4	4.83e-02	6.14e-02	1.24e-01	1.68e-01	2.52e-01
	M5	5.32e-02	1.04e-01	3.24e+00	3.43e+01	2.05e+02
	M6	3.55e-02	5.40e-02	7.63e-02	2.71e-01	9.81e-01

Table 13: Statistics of the fitting errors for clean segments.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.40e-02	8.30e-02	2.21e-01	2.38e-01	4.07e-01
	M2	2.67e-03	1.74e-02	1.73e-01	4.40e+10	3.45e+11
	M3	1.68e-01	4.81e-01	8.83e-01	6.03e-01	5.43e-01
	M4	4.39e-03	1.50e-02	4.02e-02	5.20e-02	1.92e-01
	M5	4.66e-03	2.14e-02	9.07e-01	3.59e+02	2.16e+03
	M6	2.08e-03	2.33e-02	1.53e-01	2.09e+06	9.54e+06
MFE	M1	4.02e-03	6.78e-03	1.17e-02	1.06e-02	1.11e-02
	M2	0.00	1.93e-03	4.89e-03	2.47e+09	3.31e+10
	M3	1.51e-02	5.33e-02	9.30e-02	6.97e-02	7.46e-02
	M4	2.35e-03	3.30e-03	4.99e-03	5.15e-03	1.24e-02
	M5	3.52e-03	1.69e-02	1.74e-01	5.02e-01	1.57e+00
	M6	2.11e-03	3.34e-03	1.47e-02	3.81e+05	1.72e+06
d_{dHaus}	M1	1.13e-01	1.99e-01	3.57e-01	2.89e-01	3.12e-01
	M2	7.07e-02	1.02e-01	2.83e-01	4.03e+10	3.20e+11
	M3	3.74e-01	6.90e-01	1.16e+00	8.36e-01	5.93e-01
	M4	5.66e-02	7.35e-02	1.38e-01	2.13e-01	3.95e-01
	M5	7.08e-02	2.97e-01	4.44e+00	1.09e+01	3.73e+01
	M6	6.59e-02	1.14e-01	2.97e-01	1.93e+06	9.11e+06

Table 14: Statistics of the fitting errors for segments suffering from uniform noise.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.86e-02	8.42e-02	2.00e-01	2.38e-01	4.15e-01
	M2	7.30e-03	5.34e-02	2.47e-01	3.32e+11	4.89e+12
	M3	2.04e-01	5.19e-01	8.97e-01	6.48e-01	6.09e-01
	M4	2.43e-02	4.17e-02	8.15e-02	9.64e-02	2.53e-01
	M5	1.98e-02	7.73e-02	1.57e+00	1.41e+02	6.27e+02
	M6	1.84e-02	7.39e-02	2.48e-01	1.95e+06	9.27e+06
MFE	M1	4.22e-03	6.39e-03	1.28e-02	1.04e-02	9.74e-03
	M2	0.00	3.16e-03	8.53e-03	4.05e+10	6.16e+11
	M3	1.97e-02	4.52e-02	8.72e-02	6.49e-02	7.05e-02
	M4	3.70e-03	5.36e-03	7.66e-03	8.02e-03	1.99e-02
	M5	5.35e-03	1.77e-02	1.88e-01	5.96e-01	2.67e+00
	M6	3.27e-03	6.27e-03	1.53e-02	3.42e+05	1.68e+06
d_{dHaus}	M1	1.24e-01	2.25e-01	3.64e-01	2.88e-01	2.28e-01
	M2	9.35e-02	1.60e-01	4.54e-01	2.90e+11	4.55e+12
	M3	3.89e-01	6.92e-01	1.21e+00	8.65e-01	7.66e-01
	M4	7.78e-02	1.12e-01	2.16e-01	2.35e-01	3.11e-01
	M5	1.08e-01	3.21e-01	4.25e+00	1.58e+01	7.22e+01
	M6	9.40e-02	1.91e-01	4.48e-01	1.86e+06	9.12e+06

Table 15: Statistics of the fitting errors for segments suffering from Gaussian noise.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.24e-02	6.95e-02	2.28e-01	2.25e-01	3.89e-01
	M2	9.75e-10	7.64e-03	1.37e-01	3.59e+11	4.90e+12
	M3	2.03e-01	5.48e-01	9.30e-01	6.06e-01	5.02e-01
	M4	2.11e-07	1.96e-02	5.05e-02	6.98e-02	2.50e-01
	M5	8.45e-04	2.47e-02	1.70e+00	3.18e+02	1.94e+03
	M6	1.23e-14	1.74e-02	1.11e-01	1.46e+06	8.36e+06
MFE	M1	3.68e-03	6.38e-03	1.18e-02	9.75e-03	9.54e-03
	M2	0.00	2.09e-03	5.83e-03	4.19e+10	6.16e+11
	M3	2.16e-02	4.50e-02	9.29e-02	6.80e-02	7.36e-02
	M4	2.38e-03	3.56e-03	5.72e-03	7.07e-03	2.08e-02
	M5	3.52e-03	1.97e-02	1.66e-01	6.36e+00	7.78e+01
	M6	2.07e-03	3.53e-03	1.04e-02	2.24e+05	1.33e+06
d_{dHaus}	M1	9.58e-02	1.86e-01	3.02e-01	2.46e-01	2.35e-01
	M2	6.05e-02	9.62e-02	2.85e-01	3.10e+11	4.56e+12
	M3	3.85e-01	7.00e-01	1.12e+00	8.16e-01	5.44e-01
	M4	5.81e-02	7.67e-02	1.60e-01	2.53e-01	6.06e-01
	M5	7.32e-02	3.66e-01	4.01e+00	8.27e+01	1.00e+03
	M6	5.81e-02	9.43e-02	2.48e-01	1.26e+06	7.78e+06

Table 16: Statistics of the fitting errors for segments suffering from under sampling.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.23e-02	6.94e-02	1.68e-01	2.33e-01	4.39e-01
	M2	7.42e-10	4.80e-03	8.95e-02	1.71e+10	2.44e+11
	M3	1.51e-01	4.65e-01	8.34e-01	5.86e-01	5.44e-01
	M4	1.70e-07	1.36e-02	4.19e-02	4.82e-02	1.65e-01
	M5	1.75e-11	1.87e-02	8.00e-01	2.18e+02	1.44e+03
	M6	1.64e-14	1.87e-02	1.05e-01	1.67e+06	8.42e+06
MFE	M1	3.68e-03	5.76e-03	1.06e-02	9.54e-03	1.05e-02
	M2	0.00	2.03e-03	5.40e-03	6.97e+08	7.00e+09
	M3	1.45e-02	4.36e-02	8.22e-02	5.95e-02	6.08e-02
	M4	2.35e-03	3.57e-03	5.75e-03	4.54e-03	3.30e-03
	M5	3.72e-03	1.48e-02	1.95e-01	6.66e-01	2.83e+00
	M6	2.08e-03	3.39e-03	9.47e-03	3.67e+05	1.74e+06
d_{dHaus}	M1	9.15e-02	1.67e-01	3.22e-01	2.64e-01	2.91e-01
	M2	6.24e-02	9.26e-02	2.39e-01	1.80e+10	2.32e+11
	M3	3.72e-01	6.00e-01	1.03e+00	7.74e-01	5.52e-01
	M4	5.83e-02	8.06e-02	1.41e-01	1.93e-01	3.85e-01
	M5	7.35e-02	3.30e-01	4.31e+00	1.63e+01	7.46e+01
	M6	6.01e-02	8.75e-02	2.57e-01	1.86e+06	8.83e+06

Table 17: Statistics of the fitting errors for segments suffering from missing data.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.66e-02	9.42e-02	2.21e-01	1.95e-01	3.17e-01
	M2	3.78e-03	1.86e-02	2.17e-01	8.50e+10	4.19e+11
	M3	2.51e-01	5.46e-01	9.40e-01	6.11e-01	5.09e-01
	M4	6.35e-03	1.87e-02	4.66e-02	8.46e-02	3.12e-01
	M5	5.98e-03	2.14e-02	1.84e+00	3.59e+02	2.06e+03
	M6	4.40e-03	2.26e-02	1.75e-01	2.30e+06	1.11e+07
MFE	M1	3.86e-03	7.25e-03	1.42e-02	1.12e-02	1.10e-02
	M2	0.00	1.89e-03	6.05e-03	6.17e+09	5.66e+10
	M3	2.40e-02	6.00e-02	9.63e-02	7.20e-02	7.07e-02
	M4	2.50e-03	3.46e-03	4.99e-03	6.86e-03	2.07e-02
	M5	3.52e-03	1.35e-02	1.39e-01	3.30e-01	1.31e+00
	M6	2.20e-03	3.44e-03	1.58e-02	2.95e+05	1.46e+06
d_{dHaus}	M1	1.21e-01	2.09e-01	3.02e-01	2.82e-01	2.83e-01
	M2	6.63e-02	1.08e-01	3.28e-01	7.06e+10	3.83e+11
	M3	4.26e-01	6.83e-01	1.14e+00	8.39e-01	5.44e-01
	M4	5.78e-02	7.19e-02	1.45e-01	2.22e-01	3.43e-01
	M5	6.83e-02	3.14e-01	3.82e+00	6.87e+00	2.74e+01
	M6	6.72e-02	1.31e-01	3.50e-01	1.91e+06	1.01e+07

Table 18: Statistics of the fitting errors for segments suffering from uniform noise and undersampling.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	3.05e-02	1.14e-01	3.75e-01	3.33e-01	5.11e-01
	M2	8.27e-03	4.91e-02	2.76e-01	9.65e+11	8.35e+12
	M3	2.21e-01	5.99e-01	1.00e+00	6.48e-01	5.44e-01
	M4	2.80e-02	4.17e-02	9.50e-02	1.13e-01	2.95e-01
	M5	2.32e-02	8.68e-02	1.71e+00	1.78e+02	6.26e+02
	M6	1.26e-02	7.35e-02	2.80e-01	2.26e+06	9.63e+06
MFE	M1	4.60e-03	7.55e-03	1.31e-02	1.11e-02	1.00e-02
	M2	0.00	3.69e-03	8.65e-03	1.20e+11	1.07e+12
	M3	2.15e-02	4.37e-02	1.05e-01	7.38e-02	7.75e-02
	M4	3.49e-03	5.20e-03	7.54e-03	8.90e-03	2.72e-02
	M5	5.01e-03	1.60e-02	2.15e-01	6.76e-01	2.13e+00
	M6	3.32e-03	6.34e-03	1.75e-02	3.78e+05	1.77e+06
d_{dHaus}	M1	1.18e-01	2.33e-01	3.74e-01	2.95e-01	2.38e-01
	M2	9.09e-02	1.55e-01	4.39e-01	8.58e+11	7.88e+12
	M3	3.49e-01	7.23e-01	1.22e+00	8.45e-01	5.70e-01
	M4	7.68e-02	9.62e-02	1.99e-01	2.33e-01	3.35e-01
	M5	1.07e-01	3.09e-01	4.02e+00	1.62e+01	5.59e+01
	M6	9.06e-02	1.93e-01	4.62e-01	1.87e+06	8.80e+06

Table 19: Statistics of the fitting errors for segments suffering from Gaussian noise and undersampling.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.49e-02	8.86e-02	2.51e-01	3.08e-01	5.28e-01
	M2	2.78e-03	1.89e-02	1.72e-01	4.94e+10	4.30e+11
	M3	1.17e-01	4.84e-01	8.34e-01	5.71e-01	5.60e-01
	M4	3.81e-03	1.48e-02	3.46e-02	3.40e-02	8.47e-02
	M5	5.08e-03	2.30e-02	2.81e-01	3.07e+02	1.70e+03
	M6	2.43e-03	2.58e-02	1.10e-01	2.69e+06	1.08e+07
MFE	M1	4.50e-03	7.60e-03	1.20e-02	1.14e-02	1.26e-02
	M2	0.00	2.68e-03	8.35e-03	1.26e+09	8.80e+09
	M3	1.34e-02	4.23e-02	8.47e-02	6.21e-02	6.77e-02
	M4	2.42e-03	3.12e-03	4.81e-03	3.98e-03	2.33e-03
	M5	4.13e-03	2.45e-02	2.46e-01	7.23e-01	2.10e+00
	M6	2.27e-03	3.11e-03	1.92e-02	5.51e+05	2.11e+06
d_{dHaus}	M1	1.05e-01	2.00e-01	3.59e-01	2.97e-01	3.46e-01
	M2	7.09e-02	1.31e-01	7.10e-01	5.05e+10	4.00e+11
	M3	3.47e-01	5.77e-01	9.78e-01	7.51e-01	5.69e-01
	M4	5.35e-02	7.01e-02	1.27e-01	1.67e-01	2.38e-01
	M5	7.18e-02	4.47e-01	4.53e+00	1.54e+01	5.21e+01
	M6	6.26e-02	1.04e-01	3.56e-01	2.73e+06	1.06e+07

Table 20: Statistics of the fitting errors for segments suffering from uniform noise and missing data.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	2.27e-02	7.24e-02	1.49e-01	2.06e-01	4.09e-01
	M2	6.60e-03	4.49e-02	1.50e-01	3.81e+09	3.59e+10
	M3	2.08e-01	4.70e-01	8.34e-01	6.25e-01	5.82e-01
	M4	2.15e-02	4.11e-02	7.82e-02	1.03e-01	2.64e-01
	M5	1.77e-02	1.09e-01	1.68e+00	1.57e+02	6.74e+02
	M6	2.23e-02	8.31e-02	2.25e-01	2.38e+06	9.84e+06
MFE	M1	3.81e-03	5.96e-03	1.24e-02	1.00e-02	1.01e-02
	M2	0.00	2.58e-03	6.35e-03	8.36e+08	8.36e+09
	M3	1.90e-02	4.52e-02	8.17e-02	5.61e-02	5.01e-02
	M4	3.77e-03	5.09e-03	7.13e-03	5.98e-03	3.50e-03
	M5	5.64e-03	2.20e-02	1.84e-01	7.40e-01	3.97e+00
	M6	3.22e-03	5.72e-03	1.40e-02	3.53e+05	1.57e+06
d_{dHaus}	M1	1.25e-01	1.96e-01	3.45e-01	2.82e-01	2.35e-01
	M2	9.05e-02	1.55e-01	3.54e-01	3.39e+09	3.39e+10
	M3	3.87e-01	6.34e-01	1.21e+00	8.11e-01	5.30e-01
	M4	7.95e-02	1.15e-01	1.99e-01	2.22e-01	2.59e-01
	M5	1.21e-01	4.93e-01	4.31e+00	2.00e+01	1.07e+02
	M6	9.45e-02	2.02e-01	3.79e-01	2.09e+06	9.26e+06

Table 21: Statistics of the fitting errors for segments suffering from Gaussian noise and missing data.

	Metho	Q1	Q2	Q3	mean	std
L^2	M1	1.63e-02	4.11e-02	1.26e-01	1.23e-01	1.94e-01
	M2	1.38e-03	3.16e-03	9.06e-03	2.37e-01	7.69e-01
	M3	1.89e-01	6.40e-01	1.18e+00	7.56e-01	7.14e-01
	M4	5.23e-03	1.82e-02	2.39e-02	2.36e-02	2.55e-02
	M5	2.44e-03	1.19e-02	1.96e+00	3.02e+02	7.86e+02
	M6	2.67e-03	1.15e-02	2.37e-02	1.87e-02	2.19e-02
MFE	M1	3.67e-03	4.76e-03	7.53e-03	6.53e-03	5.46e-03
	M2	1.31e-03	1.80e-03	3.07e-03	8.41e-03	1.90e-02
	M3	1.00e-02	3.24e-02	7.65e-02	5.31e-02	5.88e-02
	M4	1.74e-03	2.50e-03	3.67e-03	3.31e-03	2.78e-03
	M5	2.88e-03	3.32e-02	2.46e-01	6.27e-01	1.89e+00
	M6	1.40e-03	1.83e-03	2.86e-03	2.73e-03	2.63e-03
d_{dHaus}	M1	7.34e-02	1.15e-01	2.01e-01	2.23e-01	3.32e-01
	M2	6.12e-02	7.21e-02	1.44e-01	1.75e-01	2.65e-01
	M3	4.37e-01	5.95e-01	1.00e+00	7.18e-01	4.59e-01
	M4	4.75e-02	7.19e-02	9.06e-02	2.24e-01	4.98e-01
	M5	6.55e-02	1.10e+00	3.82e+00	1.94e+01	6.76e+01
	M6	5.47e-02	6.82e-02	7.80e-02	9.45e-02	9.76e-02

Table 22: Statistics of the fitting errors for segments suffering from small deformations.