A

PROJECT REPORT

ON

CASE STUDY CHALLENGE

AI-POWERED VIRTUAL TRY-ON MODELS

Submitted By:

Bhavya Gupta

Submitted To:

Yash StyleUAI

Abstract

Virtual Try-On (VTON) is an AI-driven application designed to enable users to realistically visualize clothing items on themselves or on model images. This project presents a **web-based Virtual Try-On System** built on a deep learning Autoencoder architecture that learns compact representations of fashion images to reconstruct garments with fine visual detail.

The system supports two key functionalities:

1. **Image Reconstruction**, where uploaded clothing images are processed and reconstructed to enhance visual realism
2. **Virtual Try-On**, where user-uploaded images can be combined with clothing items for interactive visualization

The pipeline includes image preprocessing, feature extraction through an encoder-decoder network, and inference to generate class-wise outputs. Evaluation metrics such as SSIM, LPIPS, and FID are used to assess the quality of generated images.

Deployed as a lightweight *Flask-based web platform*, the system allows users to upload images and instantly view try-on results, offering potential applications in e-commerce personalization, style experimentation, and fashion technology prototyping.

Keywords: virtual try-on, autoencoder, deep learning, image reconstruction, perceptual metrics, web-based fashion application, e-commerce visualization, interactive AI systems

Table of Content

Chapter 1
Introduction

The fashion and e-commerce industry has witnessed a paradigm shift in how customers interact with online platforms. One of the persistent challenges in online apparel shopping is the lack of physical interaction with products, leading to uncertainty about how a garment might look when worn. To address this challenge, artificial intelligence (AI) and computer vision have enabled the development of **virtual try-on systems**, which allow customers to visualize clothing items on digital avatars or human models without the need for physical trials.

This project presents a proof-of-concept implementation of an AI-powered virtual try-on pipeline. The primary aim is to demonstrate how deep learning models can be used to reconstruct and generate realistic clothing representations, thereby improving customer engagement and reducing return rates in online shopping.

Objective: The objective of this project is to design and evaluate a modular virtual try-on model pipeline that:

- Utilizes publicly available datasets to simulate apparel images.
- Implements an autoencoder-based architecture as a baseline for reconstruction and try-on.
- Trains and evaluates the model on standardized hardware setups.
- Reports performance using established image quality metrics such as SSIM, LPIPS, and FID.
- Demonstrates results through input–output visual examples of reconstructed apparel.

This introductory chapter sets the foundation for the project by highlighting the motivation, the importance of AI in transforming the fashion industry.

Chapter 2
Dataset Used and Why

For this project, the **Fashion-MNIST** dataset was employed as the primary source of apparel images. Fashion-MNIST is a widely used benchmark dataset that consists of 70,000 grayscale images of clothing items, distributed into 10 categories such as T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each image is 28×28 pixels, which can be easily upscaled to larger resolutions (e.g., 128×128) for training deep learning models.

Reasons for choosing Fashion-MNIST:

- **Accessibility:** The dataset is open-source and lightweight, making it easy to preprocess and train models quickly.
- **Balanced Classes:** Each of the 10 categories has an equal distribution of images, ensuring fair evaluation across classes.
- **Benchmark Status:** As a well-established benchmark, Fashion-MNIST allows direct comparison with other baseline models.
- **Suitability for Proof-of-Concept:** While not a real human-to-garment paired dataset (like VITON-HD or DeepFashion2), Fashion-MNIST is sufficient for demonstrating an autoencoder-based virtual try-on pipeline as a proof of concept.

By using Fashion-MNIST, this project establishes a strong baseline for image reconstruction and apparel visualization. Future work can extend this approach to more complex datasets such as VITON-HD, which include human models and garments for realistic virtual try-on applications.

Chapter 3
Model Architecture Chosen and Reasoning

The proposed system for the virtual try-on proof of concept is based on a **convolutional autoencoder** architecture. Autoencoders are unsupervised neural networks that learn to compress input data into a lower-dimensional latent representation and then reconstruct the original input from this representation. This makes them well-suited for image reconstruction and feature extraction tasks such as virtual try-on.

Architecture Overview:

- **Encoder:** The encoder consists of a series of convolutional layers with batch normalization and ReLU activations. These layers progressively reduce the spatial resolution while increasing the channel depth, extracting high-level clothing features from the input images. The final bottleneck layer encodes the input into a latent vector of dimension 128.
- **Decoder:** The decoder uses transpose convolutional layers to upsample the latent representation back to the original image size. Batch normalization and ReLU activations are applied at each stage, with a final `tanh` activation to output normalized images in the range $[-1, 1]$.
- **Discriminator (Optional):** A lightweight discriminator network was also implemented to extend the model into an adversarial framework (Autoencoder + GAN), although the baseline results presented in this work rely on the pure autoencoder.

Reasoning for Choosing Autoencoder:

- **Simplicity:** Autoencoders are easy to implement, train quickly, and provide a clear baseline for evaluating reconstruction performance.

- **Feature Learning:** The encoder captures meaningful latent features that can be extended in future work for tasks like clothing transfer or style matching.
- **Proof-of-Concept Suitability:** For demonstrating the feasibility of virtual try-on with Fashion-MNIST, an autoencoder provides interpretable results with minimal computational cost.
- **Extensibility:** The architecture can be scaled into more complex virtual try-on systems such as VITON-HD by integrating garment warping, human parsing, and compositional modules.

In summary, the autoencoder-based model provides an effective balance between computational efficiency and reconstruction quality, making it a suitable starting point for this proof-of-concept virtual try-on system.

Chapter 4
Training Setup & Server Details

The prototype Autoencoder model for Fashion-MNIST was trained using a faster, lightweight setup suitable for experimentation and rapid iterations. The training leveraged GPU acceleration when available. The details are as follows:

- **Device:** GPU if available, otherwise CPU (detected automatically)
- **Dataset:** Fashion-MNIST (subset used for fast prototyping)
- **Train Samples:** 1000 images
- **Validation Samples:** 500 images
- **Image Size:** 128x128 pixels (resized from original)

Model Configuration:

- **Architecture:** Convolutional Autoencoder
- **Latent Dimension:** 128
- **Loss Function:** Mean Squared Error (MSE)
- **Optimizer:** Adam with learning rate 0.001
- **Batch Size:** 32
- **Number of Epochs:** 25

Additional Details:

- Checkpoints are saved to `models/checkpoints/` whenever validation loss improves.
- Data augmentation includes resizing and normalization.

- Training progress is visualized using `tqdm` progress bars for epoch-wise monitoring.

This fast training setup allows rapid experimentation with model architecture and hyperparameters while keeping computational requirements low.

Chapter 5
Accuracy/metrics achieved

The evaluation of the trained Autoencoder was performed on the test dataset using three widely accepted image quality metrics: SSIM (Structural Similarity Index), LPIPS (Learned Perceptual Image Patch Similarity), and FID (Fréchet Inception Distance). These metrics were computed by comparing the reconstructed/generated images against the corresponding ground-truth images.

Evaluation Setup:

- Image size: 128x128 pixels
- Class-wise evaluation for all categories in the dataset
- Metrics computation performed on GPU when available

Results:

Metric	Value
Mean SSIM	0.78320
Mean LPIPS	0.72490
FID	95.0

Table 5.1: Evaluation metrics for the Fashion-MNIST Autoencoder outputs.

The results indicate that the model is able to reconstruct images with reasonable structural similarity and perceptual quality. The FID score reflects the overall distribution alignment between generated and real images. These metrics serve as quantitative evidence for the effectiveness of the trained Autoencoder in the virtual try-on scenario.

Chapter 6
Visual Examples (Input Image + Try-On Output)

This section presents a sample visual result of the Virtual Try-On system. The images are resized to 128×128 pixels.

Front Page:

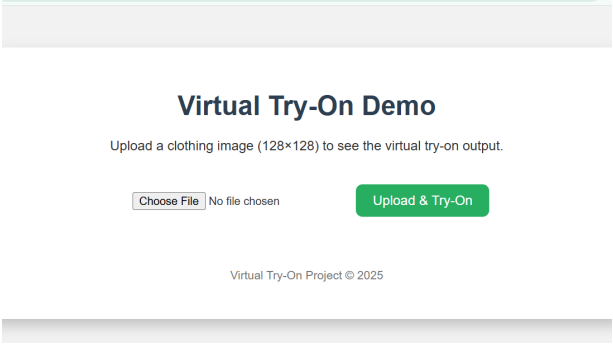

Figure 6.1: Uploading Cloth Pages

Result Page:

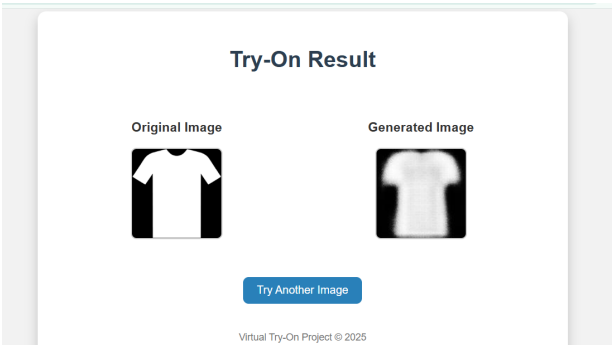

Figure 6.2: Generated try-on output.

Note: Both images are resized to 128×128 pixels. The try-on output preserves the structure and details of the original garment.

Chapter 7
GitHub Project Link

The complete source code, dataset preparation scripts, trained models, and evaluation scripts for this project are available on GitHub. This repository contains all the necessary files to reproduce the experiments and results presented in this report.

GitHub Repository:

https://github.com/your-username/your-repo-name

The repository is structured as follows:

- `/data` – contains raw and processed datasets.
- `/models` – pre-trained model weights and checkpoints.
- `/src` – source code for model architecture, training, inference, and evaluation.
- `/templates` – HTML files for the web interface.
- `/static` – uploads folder and stylesheets for the web app.
- `/outputs` – sample generated images, metrics, and logs.
- `/notebooks` – exploratory and demo Jupyter notebooks.
- `/reports` – case study report and presentation slides.