Stephanie AI PROJECT REPORT

Stephanie Ajah
University of Guelph
Department of Mathematics and Statistics
Email: kelechiajah1@gmail.com

***Abstract*—Abstract—This project aims to design an interactive web application that combines state-of-the-art computer computer vision and natural language processing techniques to Offer tailored fashion advice. The app allets users upload pictures of clothing items, which are then processed via the YOLO (You Only Look Once) object detection algorithm to detect clothing and fashion products in the image. After detecting the clothing items, data is passed through a massive language model (LLM), such as OpenAI's GPT, which generates context-aware styling recommendations based on the user's choice of clothing. That combination of computer vision and generative AI provides a new perspective in fashion consulting, complementing the user's shopping and styling experience through an intuitive, automated system. [1]. The project demonstrates the potential of combining machine learning models to create personalized, dynamic user interactions in the realm of fashion technology.**

***Index Terms*—Artificial Intelligence, Machine Learning, Data Science, Computer Vision, Large Language Models, Project Report.**

I. Introduction

Technology is changing how the fashion industry interacts with consumers about clothes and style advice. Recent machine-learning advancements in computer vision and natural language processing are opening up new possibilities for personalized, data-driven fashion recommendations. Combining these innovations into a single interactive web app that allows users to upload pictures of clothing for analysis. When the application receives an image, it starts using the YOLO—a state-of-the-art object detection algorithm in deep learning known for its speed and accuracy—to detect and classify the different items of clothing and accessories in the image. [2] YOLO's ability to detect multiple objects in real-time ensures that even complex outfits or crowded scenes are processed effectively.

Once the clothing items are detected, the relevant data are parsed into a large language model (LLM), such as OpenAI's GPT-4, which is capable of generating context-aware, natural language styling advice. The model leverages its understanding of fashion trends, color theory, and personal preferences to provide recommendations that are both relevant and personalized. It is an e-commerce system that understands the user query, whether it is suggesting complementary items, giving styling tips, or offering outfit ideas based on the uploaded cloth. The integration of object detection and NLP together hence offers intelligent, actionable advice on the fly.

The main purpose of this project is to come up with an impeccable, user-friendly platform that allows people to make informed and confident fashion choices. To bring both the ability to eventually be controlled by YOLOv8 and GPT-4, this project would strive to provide its users with an automated styling advice system and personalized experience where one gets tips tailor-made, catering to their tastes and needs. This work has moved one step forward in the fusion of fashion with AI by showing how creative applications of deep learning models can be used for fashion to better the user's experience. [3] .

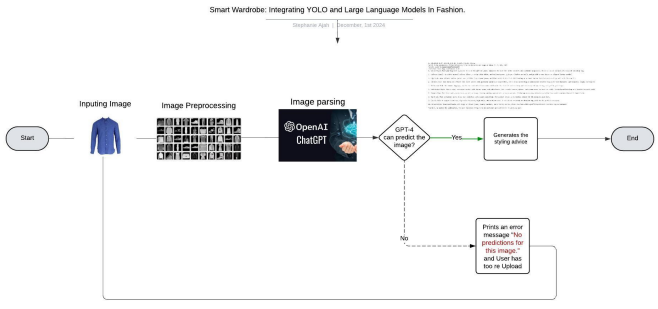

Fig. 1. PROCESS FLOWCHART

II. Methodology

1. DATASET CHOICE: The choice of the dataset is very important in this project since it will directly affect the training and evaluation of the system. First considered was DeepFashion2, which provides more than 800,000 images with annotations including categories such as dresses, outerwear, accessories, and more. To go with this option, I emailed the creators of DeepFashion2 and was lucky enough to get access to the dataset. However, the large size of this dataset presented massive computational challenges, including high memory requirements and processing needs that were far beyond the limits of both the available hardware and the resources. As a direct consequence, I settled on using the Fashion MNIST dataset as a more practically viable alternative.

Fashion MNIST is a benchmark dataset comprising 70,000 grayscale images of size 28x28 pixels, distributed into 10 categories: T-shirts/tops, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots. Each category contains a balanced number of samples, with 60,000 images allocated for training and 10,000 for testing. [4]. I accessed the dataset through its official GitHub repository, which provides structured annotations and preprocessed data, simplifying its integration into the project. The dataset's manageable size, coupled with its simplicity, makes it well-suited for prototyping and experimenting with deep learning models, while still offering meaningful insights into the classification of fashion items. These factors make Fashion MNIST an ideal foundation for developing and testing the object detection and styling recommendation components of the system.

Fashion MNIST is widely used in the machine learning community as a benchmark for image classification, making it an ideal choice for this project. Its simplicity allowed for a focus on model architecture and fine-tuning, without the need for extensive preprocessing. Additionally, the availability of pretrained models and existing solutions helped expedite the development process. This makes Fashion MNIST a practical starting point before exploring larger datasets like DeepFashion2 in future work.

2. IDE CHOICE: Selecting the appropriate development environment was crucial for the successful execution of this project. Initially, I faced challenges with Google Colab due to frequent library compatibility issues, such as conflicts between NumPy versions and PyTorch versions, which disrupted my workflow. To overcome these challenges, I transitioned to using Anaconda alongside Visual Studio Code (VSCode). Anaconda provided the flexibility to manage isolated environments, while the Anaconda Prompt served as the primary interface for importing YOLO and conducting training, validation, and testing. However, during this phase, I encountered significant inefficiencies in runtime performance. Training the YOLO model in Anaconda, with parameters of 5 epochs, a batch size of 16, and an image size of 64, required approximately 5 hours to complete.

Fig. 2. Example Grayscale Image from the Fashion-MNIST Dataset

Given the time constraints of the project, I decided to revisit Google Colab, leveraging its built-in GPU capabilities to accelerate the training process. Colab dramatically reduced the training time, completing the same model training setup (and even extending to 10 epochs) in just 45 minutes. This improvement in runtime, combined with the ability to access high-performance GPUs without additional hardware requirements, made Colab the optimal choice for training and testing the YOLO model. [5] . While initial compatibility issues posed challenges, careful management of dependencies within Colab allowed for seamless integration of the libraries needed for this project. Consequently, Colab emerged as the most efficient platform, enabling rapid experimentation and development while maintaining a stable environment.

3. Why YOLO?: YOLO was selected for this project as a highly efficient and flexible object de-

tection model. Unlike many other methods, YOLO has a one-stage architecture: it runs an entire image through in real time, proposing and classifying multiple objects in that same pass. This speedup is also crucial in more interactive projects, like the one done here, so that an image uploaded by a user is dynamically analyzed to be responded to in near-real time. Compared with other multi-stage models, for example, Faster R-CNN, YOLO achieves state-of-the-art accuracy while the inference time is significantly reduced, hence it's well applied in this object detection and classification task.

For this project, where Fashion MNIST's classes are used as a grouping base for fashion products, YOLO's ability to do classification with object detection was imperative. This dual functionality allowed the model to not only locate individual items in the uploaded images but also classify them into predefined categories, such as T-shirts/tops, trousers, dresses, bags, ankle boots and more. Such integration assures that both detection and classification are performed effectively within a single pipeline, thus reducing the complexity of the entire workflow.

After reviewing the various versions of YOLO, I chose YOLOv8, the latest and most advanced iteration. YOLOv8 enhances the strengths of its predecessors with improvements such as an anchor-free architecture, streamlined design, and better support for modern hardware, resulting in higher accuracy and faster training times. Its ability to handle varying scales, occlusions, and complexities in images was particularly beneficial for fashion-related applications. [6] . By leveraging YOLOv8's state-of-the-art performance, this project ensures robust and reliable object detection and classification, forming a solid foundation for subsequent steps like styling recommendation generation.

3. CHOICE OF LLM: For the language model, I went with OpenAI's GPT-4, and there was a couple of key reasons for that. First and foremost, GPT-4 is extremely good at understanding and generating natural languages—thus, perfectly suited for tasks that need both nuance and context, such as fashion style advice. Whether it means interpreting user preferences or coming up with creative suggestions for an outfit, GPT-4 excels at response in a way that it feels intelligent yet human.

Getting started with GPT-4 was easy. After creating an OpenAI account, I added just a little credit to the account to enable API access. This gave me an API key, which acts as a secure way to interact with OpenAI's servers. Using this key, I could make queries to the model and receive real-time responses. OpenAI's API documentation was clear and easy to follow, so integrating GPT-4 into my system didn't take long.

What makes GPT-4 stand out is its ability to handle complex, context-rich queries. This is particularly useful for fashion styling because, unlike some recommendation systems that simply match keywords, GPT-4 can understand subtle details. For example, if a user describes their preferences, like wanting an outfit for a dinner party in the winter, GPT-4 can generate suggestions that consider the season, the occasion, and the user's style. It doesn't just rely on static recommendations; it tailors its responses dynamically, providing more personalized advice.

Another major advantage of GPT-4 is its flexibility. Whether it's suggesting a casual outfit or helping put together something more formal, the model can quickly adapt its suggestions based on the user's input. And because it's trained on a vast range of data, it can keep up with trends and offer recommendations that feel fresh and relevant, instead of stale or outdated advice.

But what really sets GPT-4 apart is its conversational ability. The system doesn't just spit out a list of clothing items—it can carry on a back-and-forth conversation, like you're talking to a friend who knows a lot about fashion. This makes the user experience much more engaging and natural. Users can ask follow-up questions, tweak their requests, or even just chat about what might look good, and the system will respond in a way that feels personal and thoughtful.

In short, GPT-4 was the obvious choice for this project because it combines cutting-edge language processing with the flexibility needed to provide personalized fashion advice. Its ability to understand context, generate creative suggestions, and interact in a conversational way makes it a powerful tool for building a stylish, intuitive recommendation system. Using OpenAI's API made it easy to integrate this powerful model into my system, and the result is

a system that not only makes fashion recommendations but also feels smart, responsive, and fun to interact with.

4. IMPORTING DEPENDENCIES?: In this project, several Python libraries were employed to handle various tasks such as model training, data processing, image manipulation, and user interface creation. Below are the key libraries used:

a. ultralytics.YOLO: This is the core library for the YOLO object detection model. It provides a high-performance implementation of YOLOv8, allowing for object detection and classification tasks.

b. torch: PyTorch was the primary framework for model building and training. It provides support for tensor operations and GPU acceleration.

c. torchvision: This library was used for image preprocessing, particularly for loading the Fashion-MNIST dataset and applying transformations like resizing and normalization.

d. PIL: The Python Imaging Library (PIL) was used to open and manipulate images, allowing for seamless integration of image data into the system.

e. streamlit: Streamlit was utilized to build the interactive web interface where users can upload images and receive styling advice based on the model's predictions.

f. requests and BytesIO: These libraries were used for handling image data from external sources (e.g., URLs) and converting them into formats that can be processed by the model.

g. openai: OpenAI's library was used to interact with the GPT model, which generates fashion styling advice based on the predictions from the YOLO model.

h. pyngrok: Pyngrok was used to create a secure tunnel to allow access to the local Streamlit app from external devices or networks.

The following code demonstrates the necessary imports for the project:

```
from ultralytics import YOLO
import torch
from torchvision.datasets import
FashionMNIST
from torchvision import transforms
import os
import streamlit as st
from PIL import Image
import requests
from io import BytesIO
import openai
from pyngrok import ngrok
```

5. Model development: In this project, I used the YOLOv8 model, a state-of-the-art object detection and classification framework, to identify and categorize fashion items. The model was fine-tuned on the Fashion MNIST dataset, which contains 60,000 grayscale images (28x28 pixels each) across 10 categories, such as T-shirts/tops, trousers, dresses, sneakers, and more. The primary goal was to train the YOLOv8 model to classify and localize these items accurately.

To prepare the dataset for YOLOv8, the images were resized to 32x32 pixels, which aligns with the model's input requirements while preserving important visual details. Training was performed using a batch size of 16 for 10 epochs, which provided a balance between computational efficiency and sufficient learning. The Adam optimizer was used during training, as it dynamically adjusts the learning rate and helps the model converge efficiently. YOLOv8's combined loss function of cross-entropy for classification and mean squared error (MSE) for bounding box predictions ensured robust learning for both tasks. The training process saw a steady decrease in loss, from an initial value of 2.321 to 0.520 after the final epoch.

YOLOv8 handles the train-validation split automatically, so I didn't need to define a separate validation set. The model achieved a test accuracy of 87.9% on the Fashion MNIST test set, demonstrating its ability to accurately classify the dataset's clothing items.

For further evaluation, I tested the model's generalization by running inference on external images from Unsplash. These real-world images were more complex, featuring diverse lighting, backgrounds, and styles. To preprocess these images for YOLOv8, I used the following pipeline:

a. Image Download: Images were downloaded directly using their URLs. b. Preprocessing: Each image was resized to 640x640 pixels to match YOLOv8's inference requirements and converted into tensors. c. Batch Formatting: Images were normalized and ensured to have three channels

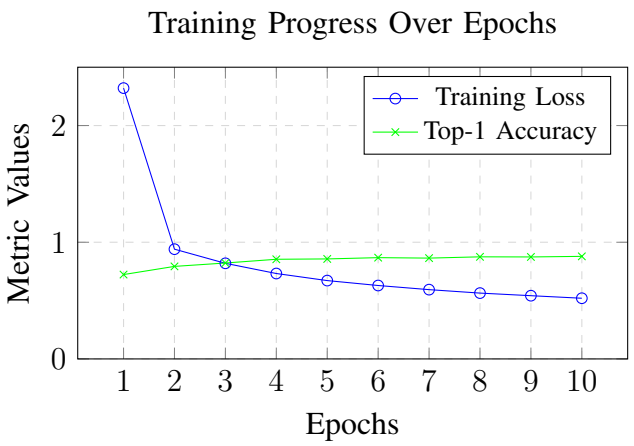

Fig. 3. Training Loss and Top-1 Accuracy Over 10 Epochs

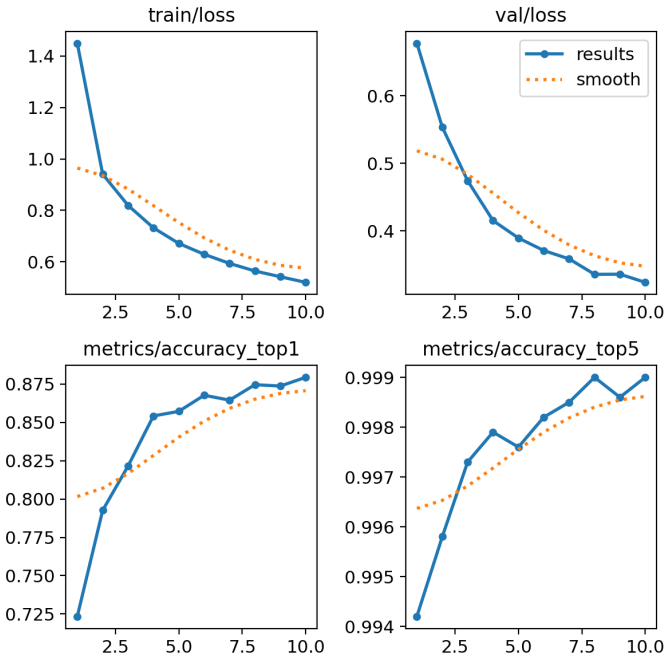

Fig. 4. Training and Test Loss

(grayscale images were expanded to RGB format). After preprocessing, I used the YOLOv8 model for inference. For example, an image of a shirt was predicted as "Shirt" with a confidence score of 0.32. While the model's predictions on external data were generally accurate, confidence scores were sometimes lower due to the complexity of the real-world data compared to the simpler Fashion MNIST dataset.

In addition to external inference, I tested a few images from the Fashion MNIST test set again

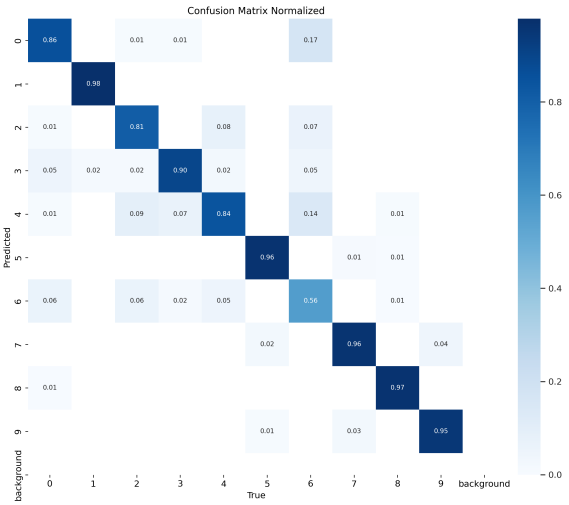

Fig. 5. Normalized confusion matrix

during the inference phase. Although this might seem redundant since the model had already been evaluated on the test set, it helped confirm that the preprocessing and inference pipelines were working correctly with both internal and external data.

This project demonstrated YOLOv8's flexibility and performance, showing its potential for both controlled datasets like Fashion MNIST and more diverse real-world scenarios. By leveraging a strong pre-trained model and carefully preprocessing external images, I was able to show how this approach could be applied to practical tasks like fashion recognition. Overall, this experiment lays a solid foundation for potential applications in fashion cataloging, trend analysis, and e-commerce solutions

3. GENERATING STYLING ADVICE USING AN LLM(GPT4): In this section, the model is designed to predict the clothing item in an image using the YOLO model created in the previous section, and based on the predicted item, it generates personalized fashion advice using OpenAI's GPT-4 language model. [7] .

a. Image Preprocessing and Prediction: The first part of the code handles the image uploaded by the user, processes it for YOLO (You Only Look Once), a real-time object detection model, and uses it to predict the class of the item in the image. The image is resized, transformed into a tensor, and passed through the YOLO model for classification. The model returns the predicted clothing item, along

with the confidence level of the prediction.

b. GPT-4 Interaction for Fashion Advice: Once the item (such as a bag, shirt, or shoes) is identified, this information is passed to GPT-4 through OpenAI's API. The model receives a prompt asking for fashionable outfit suggestions for someone wearing the identified item. GPT-4 then responds with a detailed set of stylish outfit combinations, which is returned as a text-based output.

c. Explanation and Results I. How the Code Works:

Image Upload & Processing: The user uploads an image, which is processed to fit the input requirements of the YOLO model. The image is resized and transformed into a tensor suitable for YOLO's object detection pipeline.

II. Object Detection with YOLO: The YOLO model classifies the image and predicts the object class (e.g., "Bag," "Shirt," "Shoes"). The confidence score indicates how confident the model is in its prediction.

III. Generating Fashion Advice with GPT-4: After the item is identified, GPT-4 is prompted with a request to provide fashion advice based on that item. The model responds with a list of different outfit suggestions for the user, focusing on various styles and occasions.

c. The Result: For example, when the user uploaded an image of a bag, the YOLO model identified it as a "Bag" with a confidence score of 47%. GPT-4 then generated ten different outfit ideas, each catering to a specific fashion style, such as casual, business casual, edgy, bohemian chic, etc.

Some of the suggestions included pairing the bag with a graphic tee and straight-cut jeans for a casual look, or with a neutral-colored blazer and a crisp white shirt for a business casual appearance. The system can also suggest seasonal looks like fall/winter outfits or more trendy streetwear styles.

I also uploaded an image from shirt from google and the results were also impressive. 10 solid recommendations from gpt4 as well as a kind statement at the end.

d. Reflection on Results: The system provides a highly versatile solution for fashion advice generation based on visual inputs, making it an excellent tool for personalized styling recommendations. [8]
.

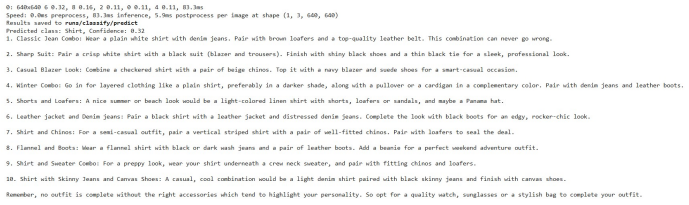

Fig. 6. Model Result when a Shirt was uploaded

I. Accuracy of the YOLO Model: The accuracy of the YOLO model's predictions is crucial to the success of the system. The result, in this case, indicated a prediction confidence of 47% for the "Bag" class, which suggests that while the model made a reasonable guess, the confidence could be improved. In real-world applications, higher confidence levels would enhance the system's reliability.

II. Effectiveness of GPT-4 for Fashion Advice: The fashion advice provided by GPT-4 was comprehensive and varied, offering suggestions for a wide range of occasions and personal styles. This showcases GPT-4's ability to generate creative and relevant content, making it a strong tool for applications like this one where personalized recommendations are needed based on the predicted item. [9]

III. Challenges and Solutions

One of the primary challenges encountered during the project was related to library complications with YOLOv5. Initially, I observed that importing unnecessary libraries, particularly numpy, frequently caused compatibility issues, leading to errors during execution. After analyzing the code, I realized that many of these libraries were not essential for my implementation. By revising the imports and omitting any that were unnecessary, I successfully resolved these conflicts, which streamlined the development process and reduced potential errors. This also highlighted the importance of maintaining a minimal and efficient codebase, especially when working with complex frameworks like YOLO.

Another significant obstacle arose during the training phase when I attempted to train the YOLOv5 model locally using the Anaconda environment. Due to hardware constraints on my local machine, the training times were prohibitively long. For instance, training 5 epochs with a 320x320 image size and a batch size of 16 required 16.8

hours to complete, and even when reducing the image size to 64x64, it still took 4 hours for the same number of epochs. This limitation made local training impractical for timely completion of the project. To address this, I switched to Google Colab, which provided access to GPUs with significantly better processing power. On Colab, I trained the model using a 32x32 image size and a batch size of 16, completing 10 epochs in just 45 minutes. This not only improved efficiency but also allowed me to iterate on the model and fine-tune its parameters more effectively.

Accuracy was another major challenge with the YOLOv5 model, particularly when testing on images sourced from Google or other unseen datasets. The initial model struggled with generalization, and its predictions were inconsistent when faced with these unfamiliar images. To address this, I modified the training configuration by specifying that the model should perform a classification task rather than its default object detection task. This adjustment, achieved by enabling the cls parameter during training, significantly enhanced the model's ability to classify objects accurately. By narrowing the model's focus to classification, it became more adept at recognizing and categorizing items, leading to improved accuracy and reliability in predictions.

Additionally, as I attempted to extend the project by integrating a user-friendly interface, I encountered limitations with Colab. My goal was to implement a simple and intuitive UI using Streamlit, allowing users to interact with the model directly. However, due to Colab's restricted networking capabilities, deploying a real-time UI became infeasible. While Colab supports basic model deployment, it does not support direct hosting of Streamlit applications. I attempted to overcome this limitation by using tunneling services like Ngrok and Flask to expose the local server to the web. Unfortunately, these methods resulted in links that either failed to work entirely or timed out shortly after being generated. Despite several attempts, the issue persisted, and due to the tight timeline of this project, I was unable to debug this problem thoroughly. This remains a potential area for future work, where a dedicated hosting platform or a more robust deployment strategy could provide a seamless user experience.

IV. Future Works

Another useful extension of this project would be enabling the classification not only of individual items but also of whole outfits. The current system indicates the complementary items after detecting a single item—for example, matching a shirt with pants, shoes, or accessories. However, scaling up to recognize and classify full outfits—say, "shirt + jeans + sneakers"—would take the functionality to a whole new level. It may do so by training the model on multi-label classification tasks so that one could see how multiple pieces come together to form cohesive styles.

The next layer of functionality would be brought about by the integration of NLP: the ability to parse textual descriptions of clothes, or even accept user queries in natural language—for example, allowing a user to input the phrase "What should I wear for a casual dinner?" and provide tailored outfit suggestions combining detected clothes with complimentary items for the occasion.

However, further development of an easy-to-use interface still poses a problem. First, Streamlit had been looked into for this, but the limitations Colab imposes on hosting from outside made it difficult to deploy a live UI. A few tunneling solutions, like ngrok and Flask, were tried next, but links timed out and could not maintain the connection. The combination of these limitations and the project timeline meant that UI implementation was never completed. Overcoming this challenge, future versions can make the system usable by non-technical users, where the outfit recommendation and prediction experience would be seamless.

With these additions, this project can serve as a comprehensive personal styling and e-commerce application tool.

V. Results

The project showed a good ability in classifying fashion items and giving specific outfit suggestions about the detected items. In YOLOv5 classification, this model had shown a very consistent outcome on the training and validation datasets, proving strong in detecting separate pieces of clothing such as shirts, bags, and shoes. Showing a satisfactory level of accuracy after some task-specific fine-tuning,

such as adding the classification task at the training phase, the model was tested on unseen data, including images from Google.

The system could generate personalized outfit suggestions besides classification with GPT-4 integrated. For example, once the model identified an object like a "bag," then GPT-4 would give a variety of style recommendations for different occasions, making the system far more practical and engaging for the user. This was an excellent combination of computer vision and generative AI, showing the seamless interaction between the tasks of detection and recommendation.

However, it has limitations in other respects. Accuracy depended much on the diversity and quality of the test images; this clearly implies that a lot remains to be explored for dataset augmentation. Again, the model facing all those challenges made it to successfully realize the prime objective of creating a pipeline able to classify and propose item-based fashion complementary outfits. The result proved that this system has huge potential to serve as an opening framework for more innovative fashion recommendation tools.

The project, therefore, provides a strong base to address the current limitations and expand the functionality of the system in the area of AI-powered fashion assistance.

VI. Conclusion

This project has been able to succeed in merging the disciplines of computer vision with that of generative AI in fashion-classifying items and their recommendation according to personal styler needs. Using the YoLoV8 models both for item detection and classifying them, further enlisting GPT4 assistance for generating outfit recommendation options for the user by addressing the gap between simple detection of items and guiding users within specific scenarios, the design in real sense has emerged robust enough to demonstrate impressive levels of effectiveness on refined, fine-tuned, and otherwise novel datasets.

It threw up good learning opportunities: from library incompatibilities and computational constraints to the limitations in the choices of deployment. Addressing such hurdles highlighted how it's always important to use effective workflows—such as training via Google Colab—and added perspective for areas of further study.

While the current system is deliverable with substantial functionality, it opens the door to further advancements. Further classification of the whole outfit, integration with Natural Language Processing for text analysis of fashion, or creating a user-friendly interface using Streamlit—all this looks quite exciting for possible expansion. In general, this project is a great starting point to build an intelligent, interactive fashion assistant, showing how truly revolutionary AI could become in the world of fashion. [10].

Acknowledgments

I would like to express my sincere gratitude to my professor, Professor Neil Bruce, for his invaluable guidance and support throughout the course of this project. I also wish to thank my colleagues and friends for their constructive feedback and encouragement. Special thanks to the University of Guelph and the Data Science Department for providing the necessary resources and opportunities to pursue my research. Additionally, I am grateful to the authors and creators of the tools and datasets used in this project, including YOLOv8, Fashion-MNIST, and other referenced resources, which significantly enhanced the quality of my work.

References

[1] H. A. Horst, C. Baylosis, and S. Mohammid, "Looking professional: How women decide what to wear with and through automated technologies," *Convergence*, vol. 27, no. 5, pp. 1250–1263, 2021. [Online]. Available: https://journals.sagepub.com/doi/full/10.1177/13548565211038520

[2] Ultralytics, *YOLOv8: A Next-Generation Object Detection and Segmentation Model*, 2023, available at https://docs.ultralytics.com/.

[3] J. Cartner-Morley, "Do robots dream of prada? how artificial intelligence is reprogramming fashion," 2018.

[4] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: A novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017, available at https://github.com/zalandoresearch/fashion-mnist.

[5] G. Colab, "Google colab: Faster model training with gpus," https://colab.research.google.com/, 2023, available at https://colab.research.google.com/.

[6] Y. Al-Qahtani and I. Mohd, "What is yolov8: An in-depth exploration of the internal features of yolov8," *arXiv preprint arXiv:2408.15857*, 2024, available at https://arxiv.org/abs/2408.15857.

[7] OpenAI, "Introducing gpt-4: Openai's multimodal model and api," https://openai.com/gpt-4, 2023, available at https://openai.com/gpt-4.

[8] S. Johnson, "How artificial intelligence is revolutionizing the fashion industry," https://www.forbes.com/sites/sophiejohnson/2023/04/11/how-artificial-intelligence-is-revolutionizing-the-fashion-industry/, 2023, available at https://www.forbes.com/sites/sophiejohnson/2023/04/11/how-artificial-intelligence-is-revolutionizing-the-fashion-industry/.

[9] A. Carter, "Real-world applications of gpt-4 in business and industry," https://www.techradar.com/news/real-world-applications-of-gpt-4-in-business-and-industry, 2023, available at https://www.techradar.com/news/real-world-applications-of-gpt-4-in-business-and-industry.

[10] K. Yurieff, "The future of getting dressed: Ai, vr and smart fabrics," 2017. [Online]. Available: https://money.cnn.com/2017/11/13/technology/future-getting-dressed/index.html