TA'AM

A platform promoting the resale and reuse of clothes to reduce environmental impact

Graduation Project Representation

Team

- Saleh Adel Saleh
- Shehab Mostafa Fahmy
- Abdelaziz Hossam Abdelaziz
- Abdulrhman Hosny Muhammed
- Muhammed Mahros Muhammed

Supervisors

- Dr. Salsabil Amin
- TA. Ahmed Hatem

TABLE OF CONTENTS

Introduction

TA’AM
Team

Introduction

The clothing industry is a major polluter, causing **10%** of global carbon emissions and **20%** of wastewater. To help reduce this impact, we can wear clothes for longer and buy second-hand instead of new ones. TA'AM is a new app that makes it easy to buy and sell used clothes, combining style with eco-friendliness for a more sustainable fashion choice.

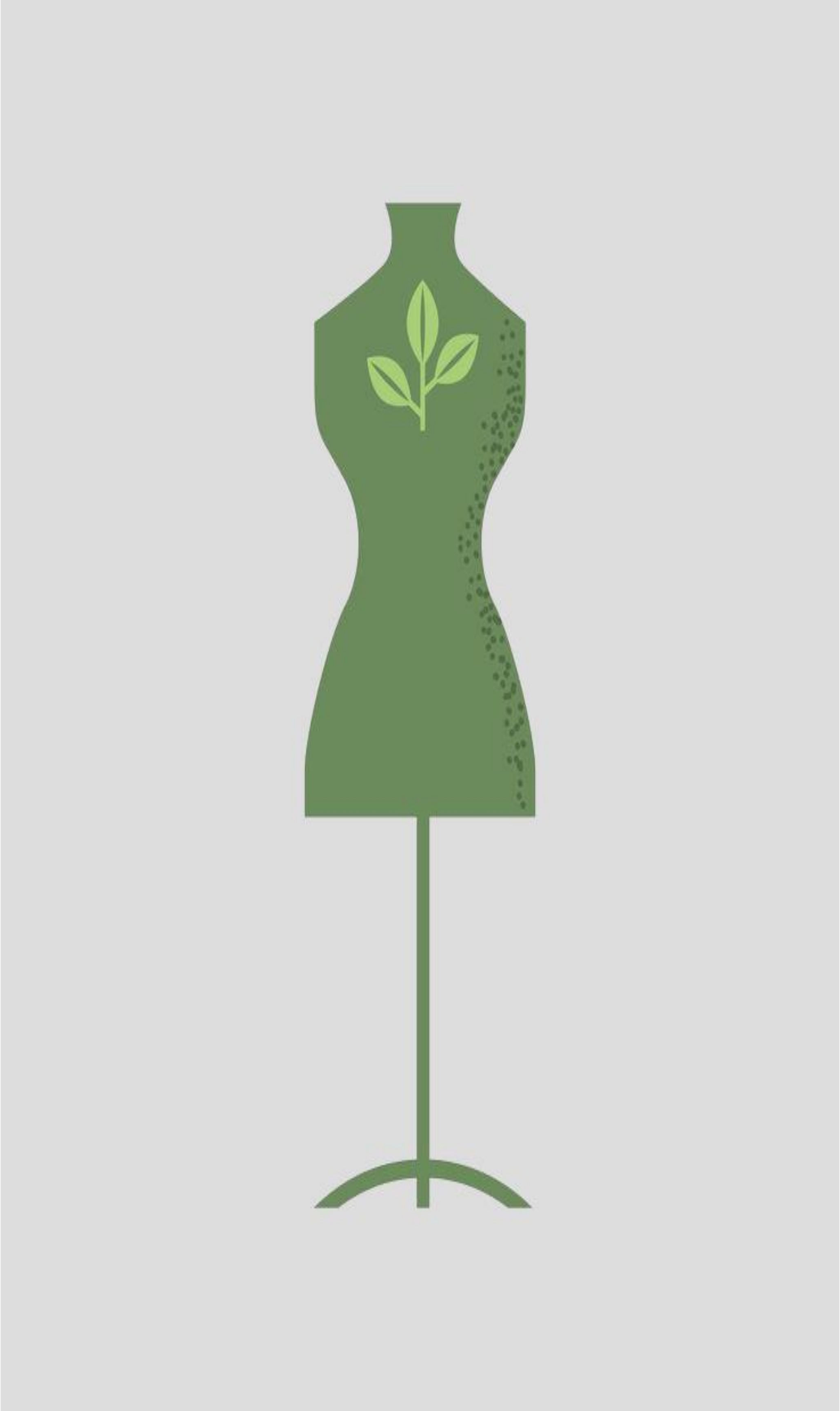
Motivation & Objective

TA’AM

Team

Motivation

- Reducing environmental impact resulting from excessive clothes manufacturing.
- Promotes sustainable fashion by extending the lifespan of clothes.
- Helping users buy clothes at a lower cost than the original ones.



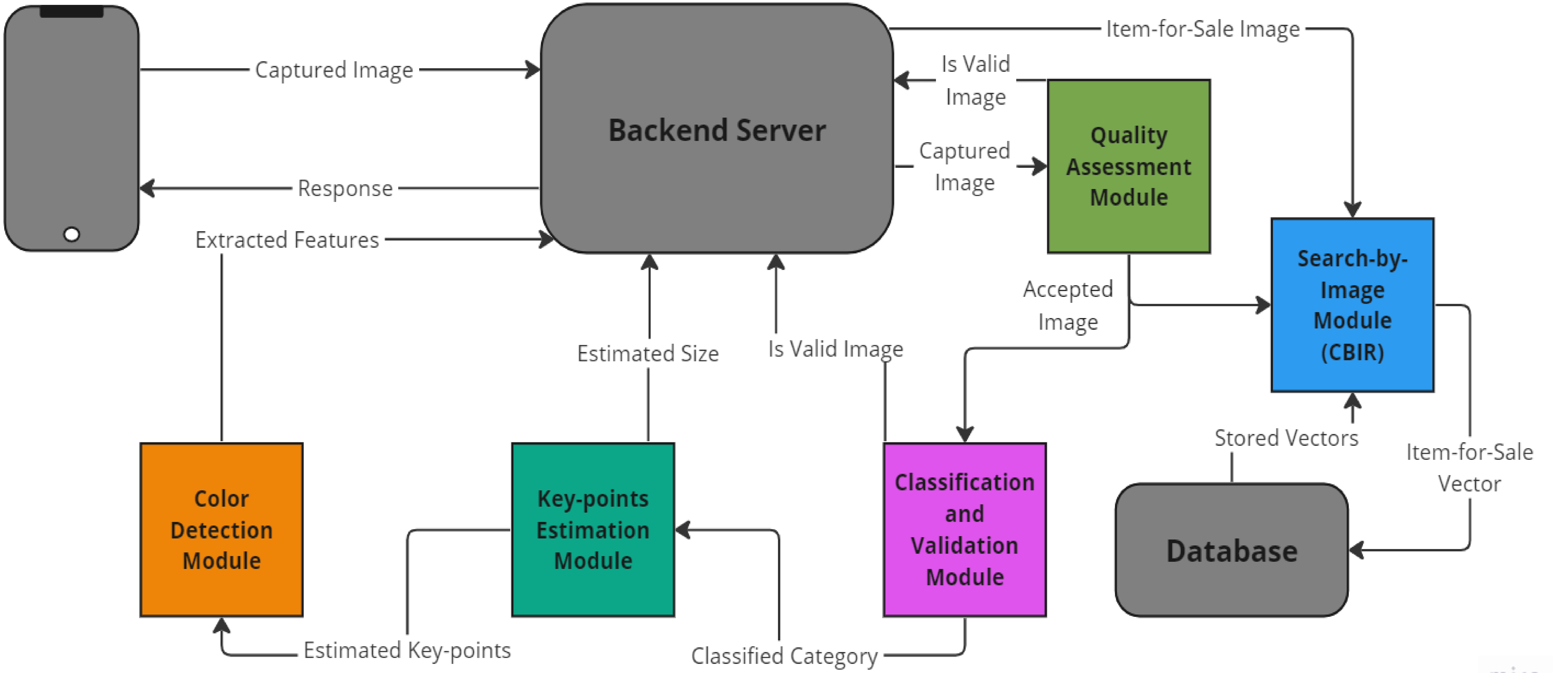
Objective

- Offer a specialized platform for users to allow them to showcase their surplus clothing and buy suitable ones at a relatively low cost.
- Enhance the user experience by implementing multiple machine learning models to assess image suitability, extract various attributes, and offer search capabilities.

System Architecture

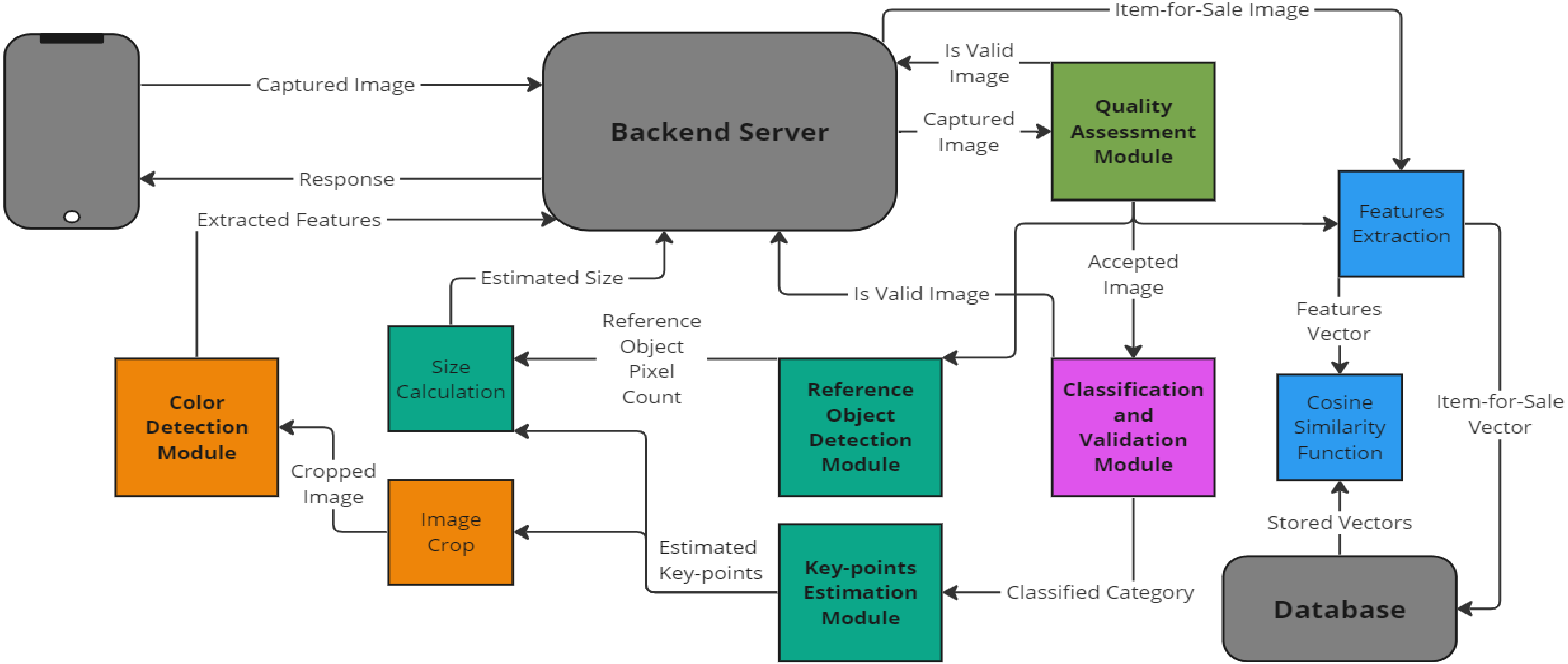
TA'AM
Team

System Architecture



TA'AM Team

System Architecture



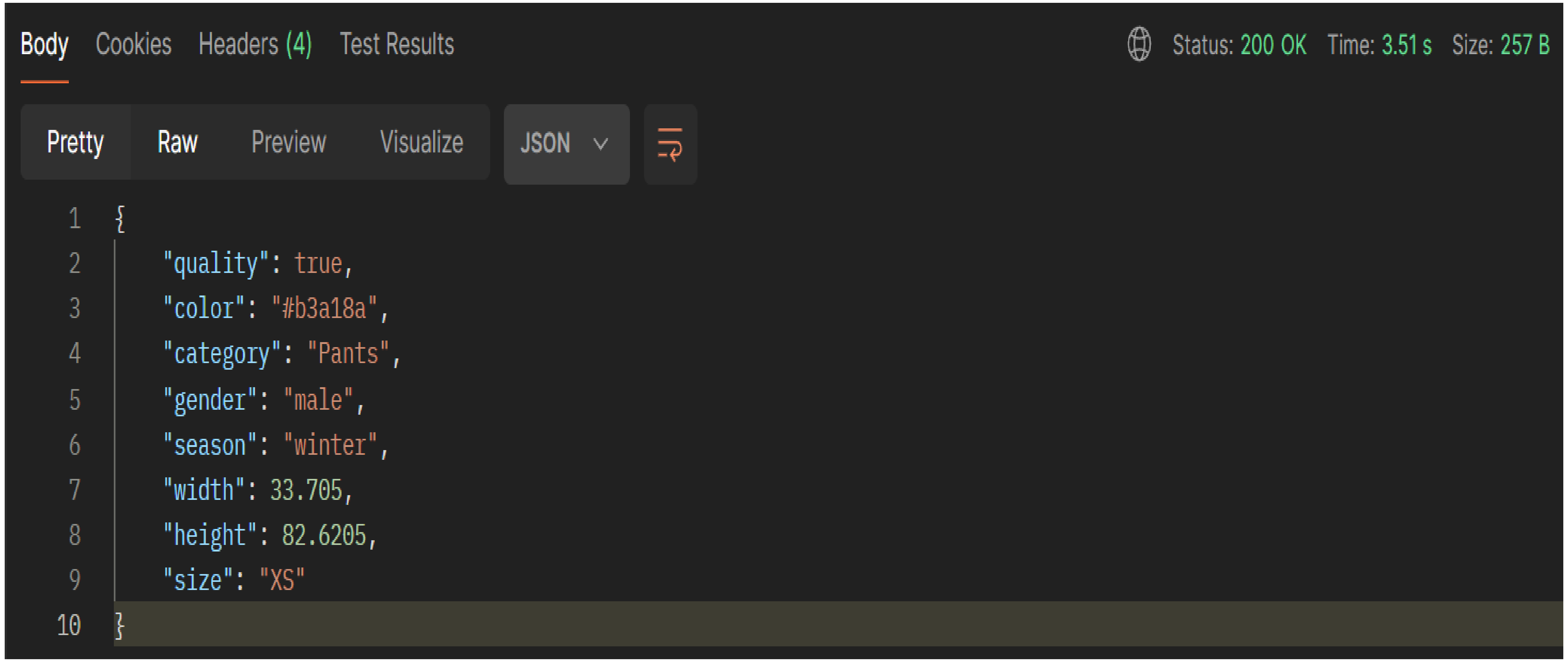
Phases Description

TA’AM
Team

Quality Assessment

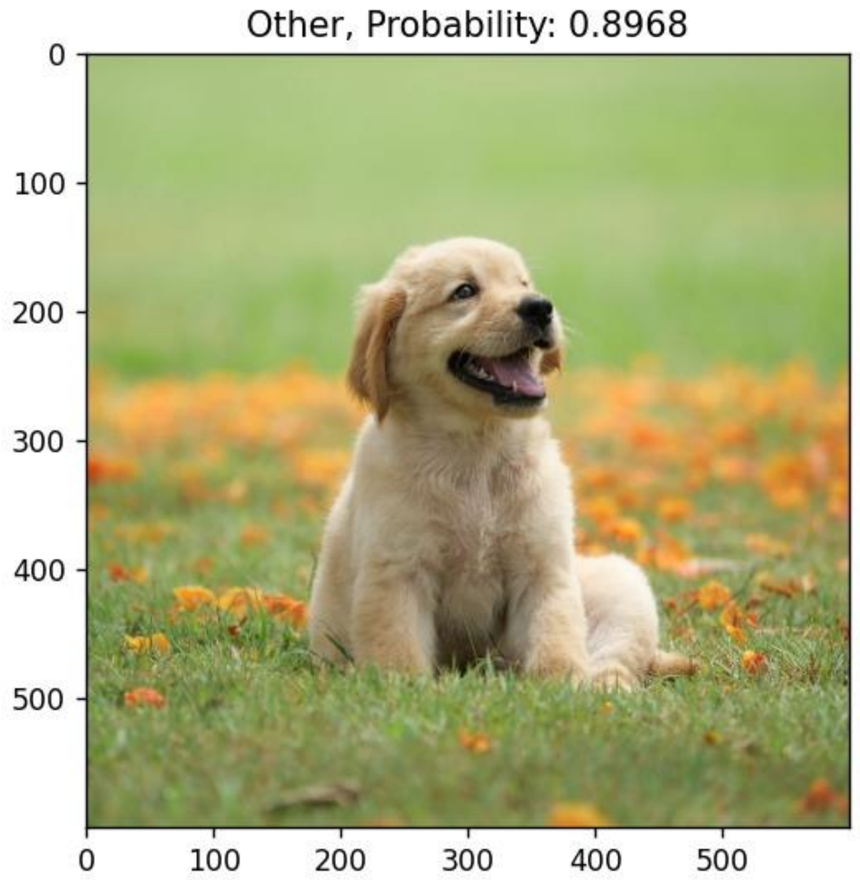
- Using the BRISQUE model, the image **quality score** was computed, and a specific threshold of **30%** was applied for image assessment.
- Brisque is a model that predicts how good an image looks without needing another image for comparison.
- Brisque extracts features like **contrast**, **brightness**, and **texture** from images, then uses **Support Vector Regression (SVR)**, to link these features to quality scores based on a dataset of manually rated images.

Feature Extraction



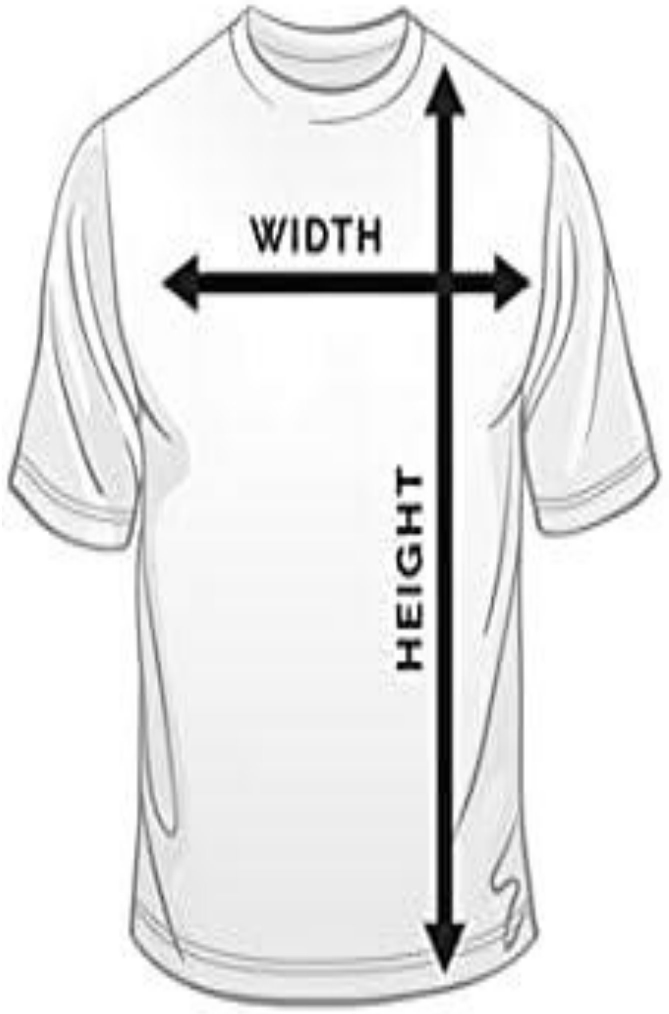
Category Classification

- This model validates whether the image **contains clothes or not**, based on that, it returns the classified category or "Other".
- In the case of "**Other**", the image is **rejected.**
- We utilized the **Pre-trained ResNet34** architecture on **Agrigorev's custom Kaggle dataset** for category classification.



Size Estimation

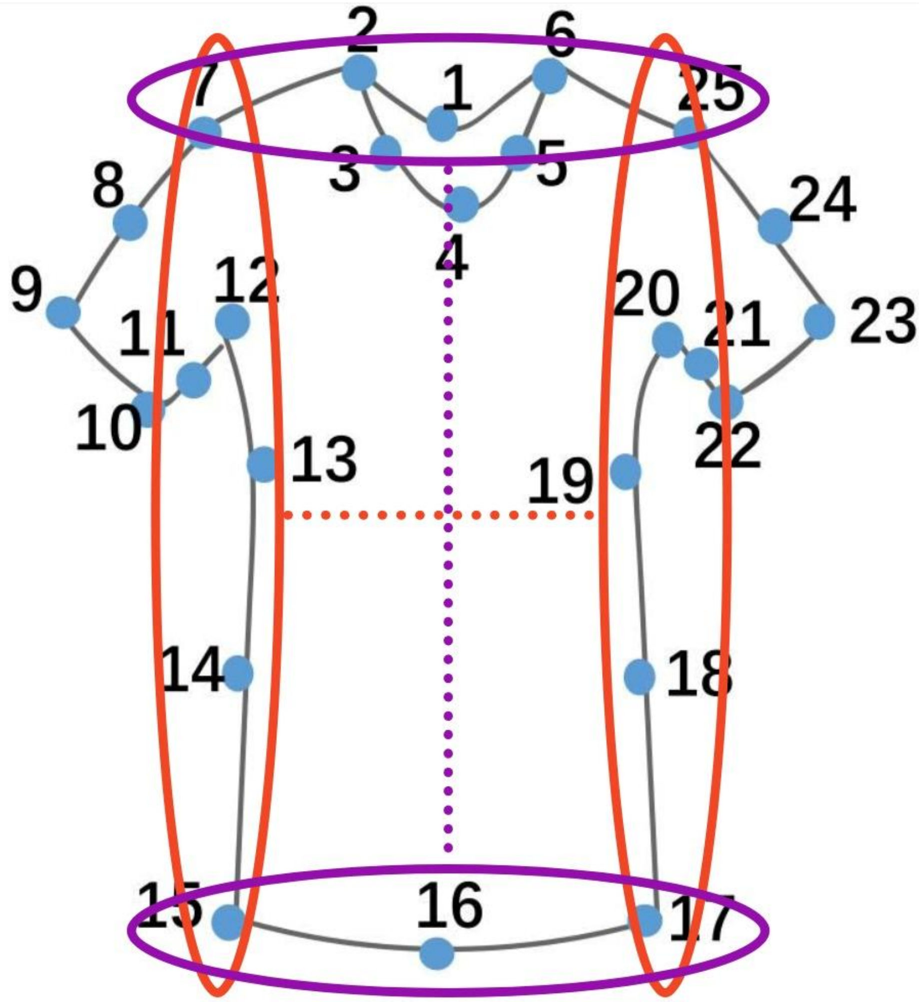
- This model is used to calculate the classified cloth size (width x height).
- Model Pipeline is:
  - **Key-points** detection model.
  - **Card detection** model.
  - The predicted key-points and the card measurements are inserted into an **equation** to calculate the estimated size of the cloth.



Size	Width	Length
S	44 CM	64 CM
M	47 CM	67 CM
L	50 CM	68 CM
XL	54 CM	71 CM
XXL	56 CM	74 CM

Key-points Detection

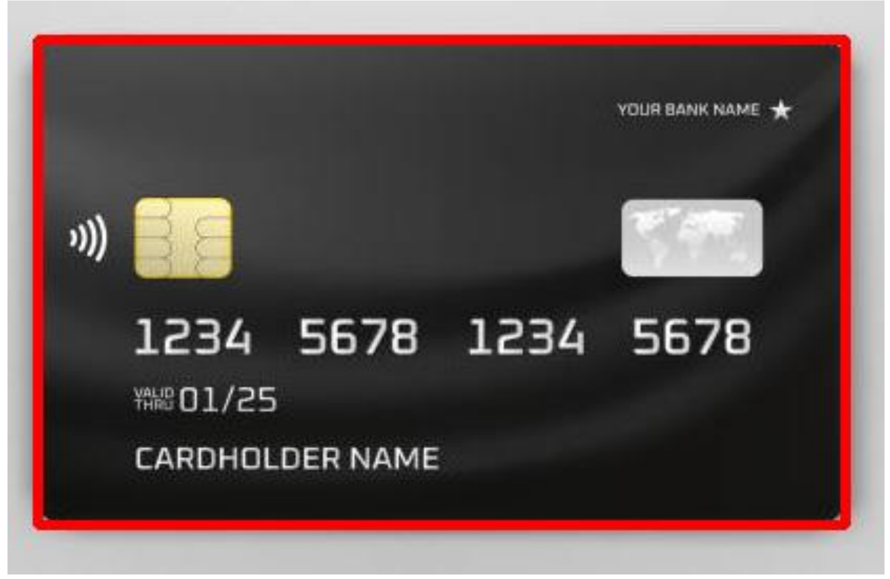
- 13 pre-trained ResNet50 models were employed to extract key-points for each category.
- The height and width of the product were determined by calculating the distances between specific key-points.



Short sleeve top

Card Detection

- This model **processes an image** containing a card, detects its boundaries, detects the largest contour, calculates the conversion factor, and returns it.
- This conversion factor allows for measurements in the image to be translated to real-world units (centimeters).
- Model Pipeline is :
  - Preprocessing
  - Contour Detection
  - Detect the largest contour
  - Get the width and height in pixels
  - Convert it from pixels to cm

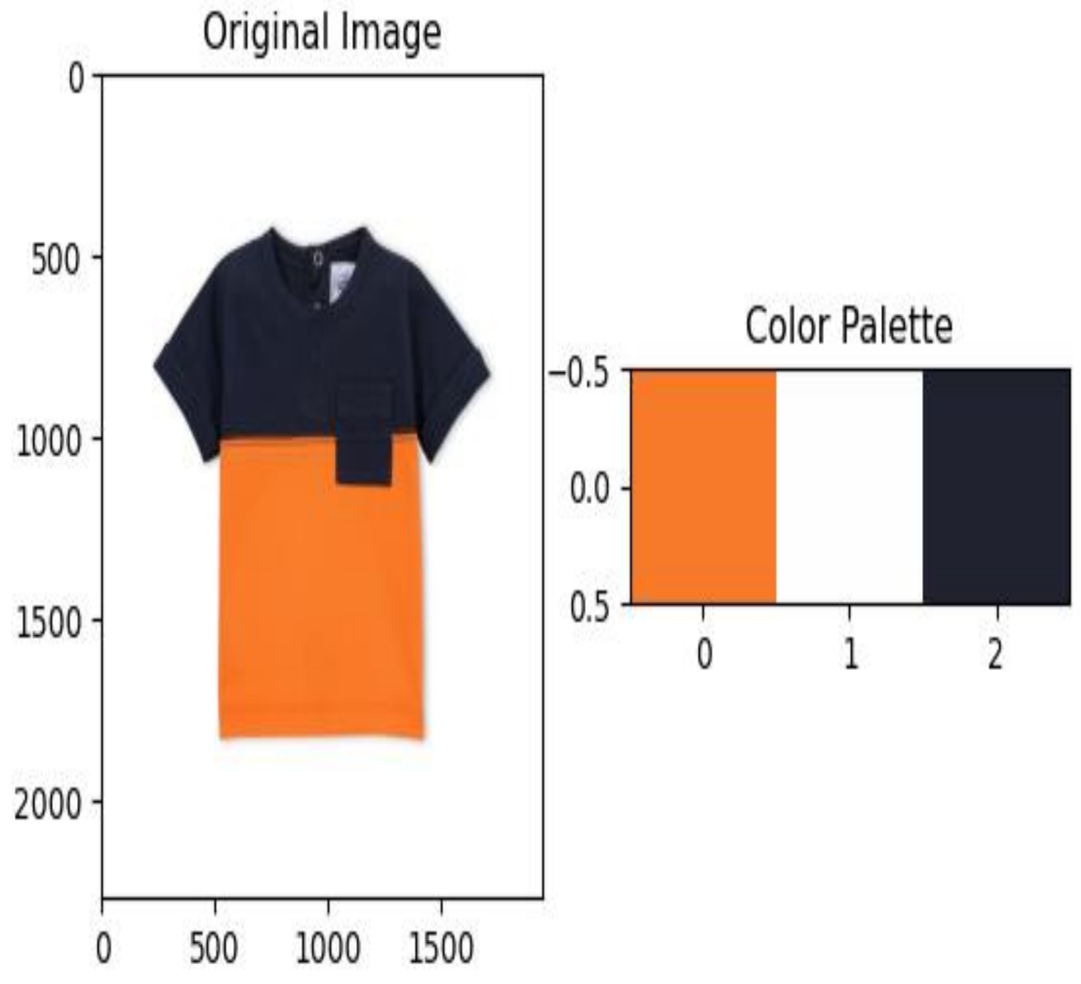


Estimated Size Equation

- An EN 13402 Standard for clothing size designation was used to map the measurements from centimeters to real-world measurements "S, M, L, XL".
- This is an example of the equation applied to get the waist of trousers:
  - Waist = (right - left) * 8.5 / reference width in pixels
  - Height = (down - top) * 5.0 / reference height in pixels

Color Detection

- The image was cropped using the detected key-points to **ignore background colors.**
- <u>K-Means</u> is used to segment the image into regions of similar colors and identify representative colors for each region, forming the color palette.
- The execution time to extract the most three dominant colors is approximately **0.1 seconds.**



Search By Image

- The MobileNet model extracts features from the query image, generating a **1 x 50176-dimensional vector** within a time frame of **20ms to 50ms.**
- Utilizing **cosine-similarity metrics**, the closest 5 products to the query image are calculated. Comparing the query image with **5000 images**, from the closest to the farthest, typically takes approximately **0.7s to 1s.**
- The **vector dimensionality is reduced to 5000** using an AvgPooling Layer followed by a Dense Layer, resulting in the model being **10 times faster.**
- Comparison is **limited to the same category** rather than all categories, making the model **13 times faster.**

Search By Image

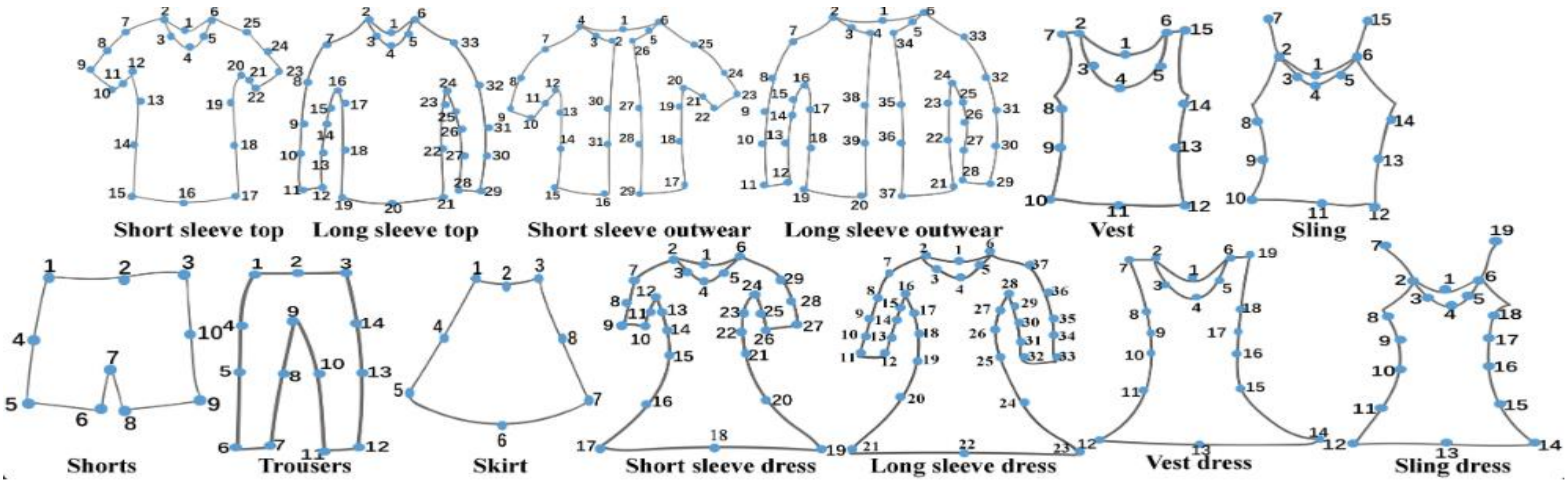
Overall, the model now takes around **30ms to 50ms** to execute.

Datasets

TA’AM
Team

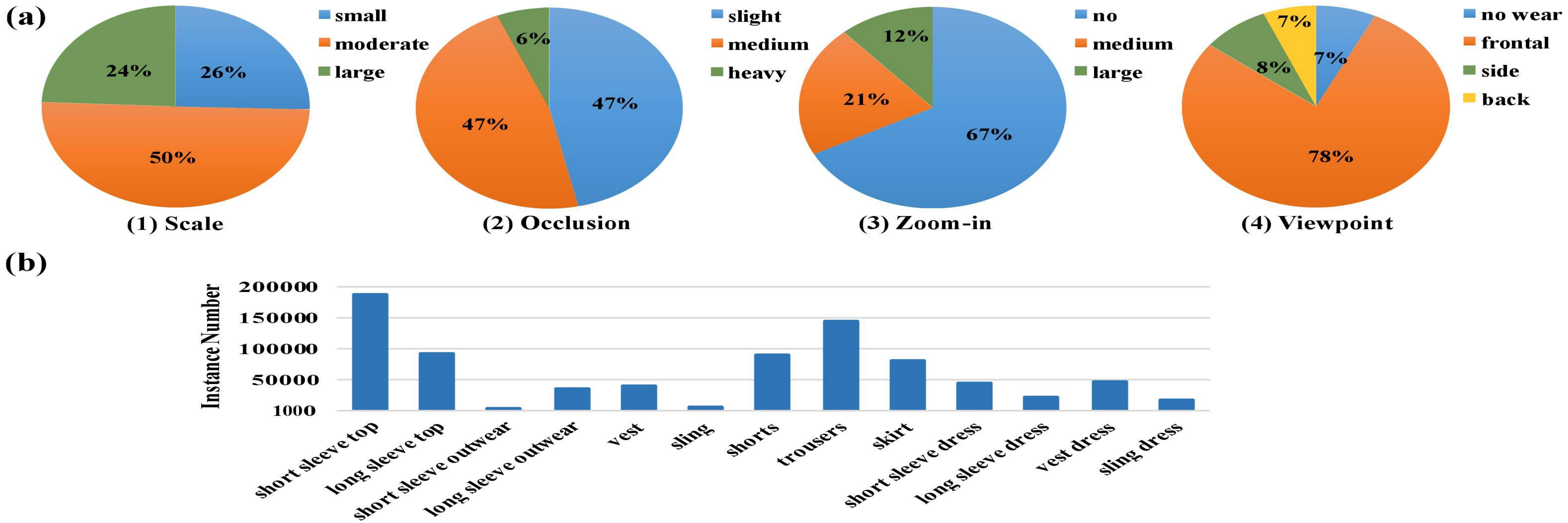
Deepfashion2 Dataset

DeepFashion2 is a comprehensive fashion dataset. It contains **286K** diverse images of **13 categories** from both commercial shopping stores and consumers.



Deepfashion2 Dataset

- To solve the problem of **data imbalance**:
  - Each category has a **separate** model



Deepfashion2 Dataset

- Challenges:
  - Variability in the number of items per image:

    Solutions:

    - Ignore images with more than one item.
    - Crop images to ensure each one contains only one category.
  - Differences in the number of key points per category:

    Solutions:

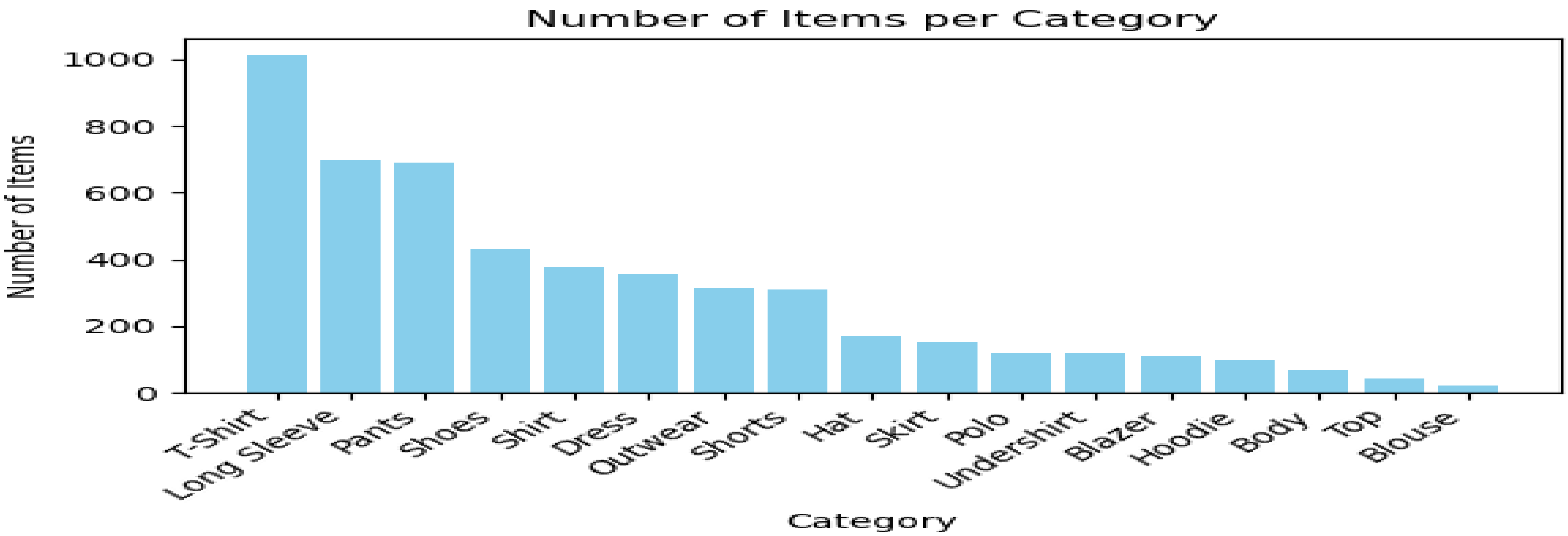
    - Train a separate model for each category.
    - Standardize the number of landmarks across all categories.
    - Choose some specific points.
    - Pad and Trim.

Agrigorev's Custom dataset

- This dataset is a 7GB "**5k** product image" of **20 different classes** taken by users.
- **80%** of the dataset used for training and **20%** for testing.

Agrigorev's Custom dataset

- To solve the problem of **data imbalance**:
  - **Data Augmentation** using FastAI library applied.
  - **Pre-trained** model was used.



Experiments and Results

TA'AM

Team

Category Classification

First Trial:

A custom CNN was used on the FashionMNIST dataset, achieving about 92.4% accuracy. However, it was considered inadequate due to the reliance on 28x28 grayscale images, which did not meet our criteria.

Second Trial:

We used Agrigorev's custom Kaggle dataset with a pre-trained ResNet34, achieving 88.5% accuracy after 50 epochs. This user-sourced dataset aligns well with our application's criteria, making it suitable for our purposes

Third Trial:

We used the DeepFashion2 dataset with a pre-trained ResNet50 architecture. However, it was not accepted because Agrigorev's dataset was more compatible with the types of images likely to be submitted, as much of DeepFashion2 included images of clothing being worn.

Category Classification

- Here's a summary table outlining the datasets and architectures employed for category classification:

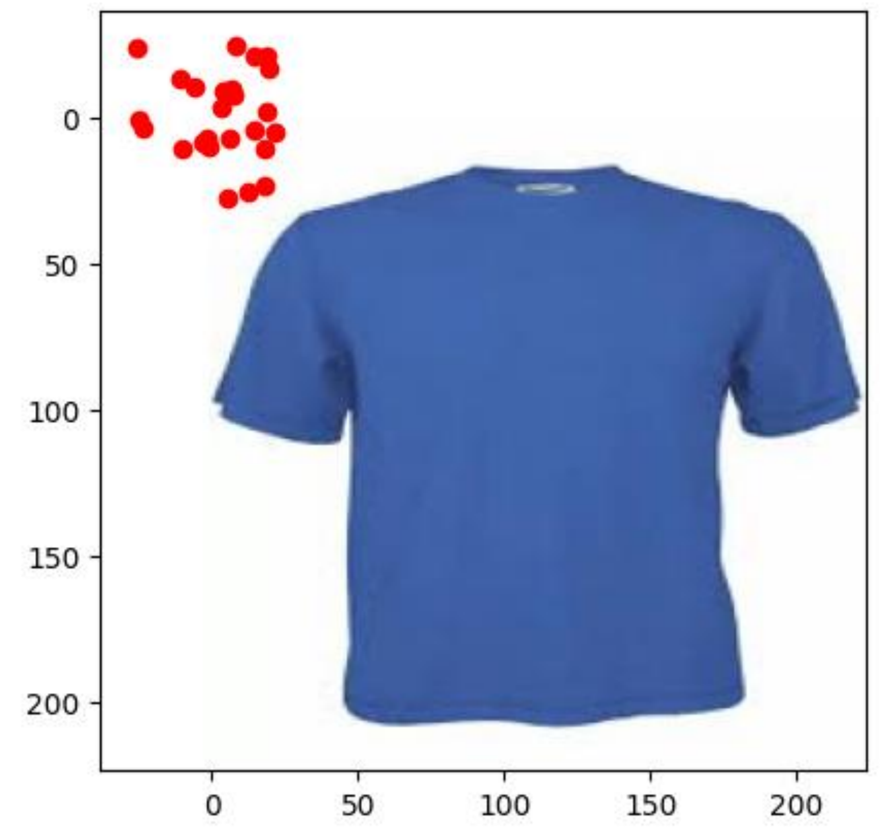
- **Higher accuracy doesn't mean it satisfies our criteria.**

Dataset	Architecture	Accuracy
FashionMNIST	Custom CNN architecture	e=10 -> 92.4%
Agrigorev's Custom Kaggle Dataset	**ResNet34**	**e=50 -> 88.5%** e=100 -> 82.3%
DeepFashion2	ResNet50	e=40 -> 90.5%

Key-points Detection

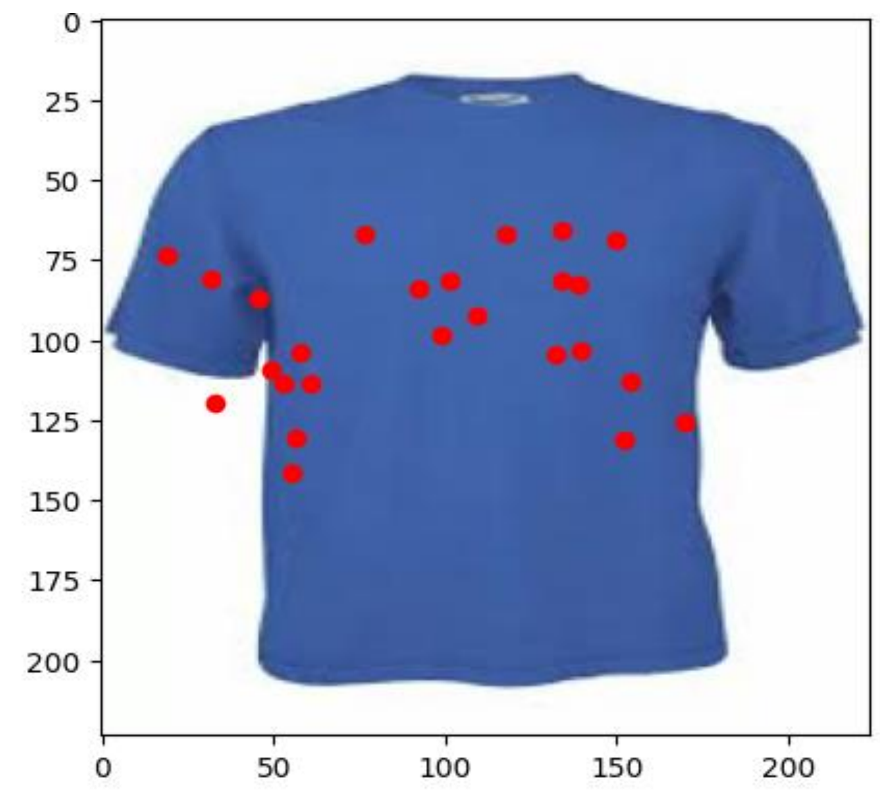
First Trial:

A basic CNN model was used for key-point detection, but it **performed poorly**, **with a loss of 7000 after 5 epochs**, indicating unsatisfactory results. The model focused on single-category images.



Second Trial:

MobileNetV2 was used for key-point detection, achieving **a loss of 2000 over 10 epochs**. This model focused on single-category images.



Key-points Detection

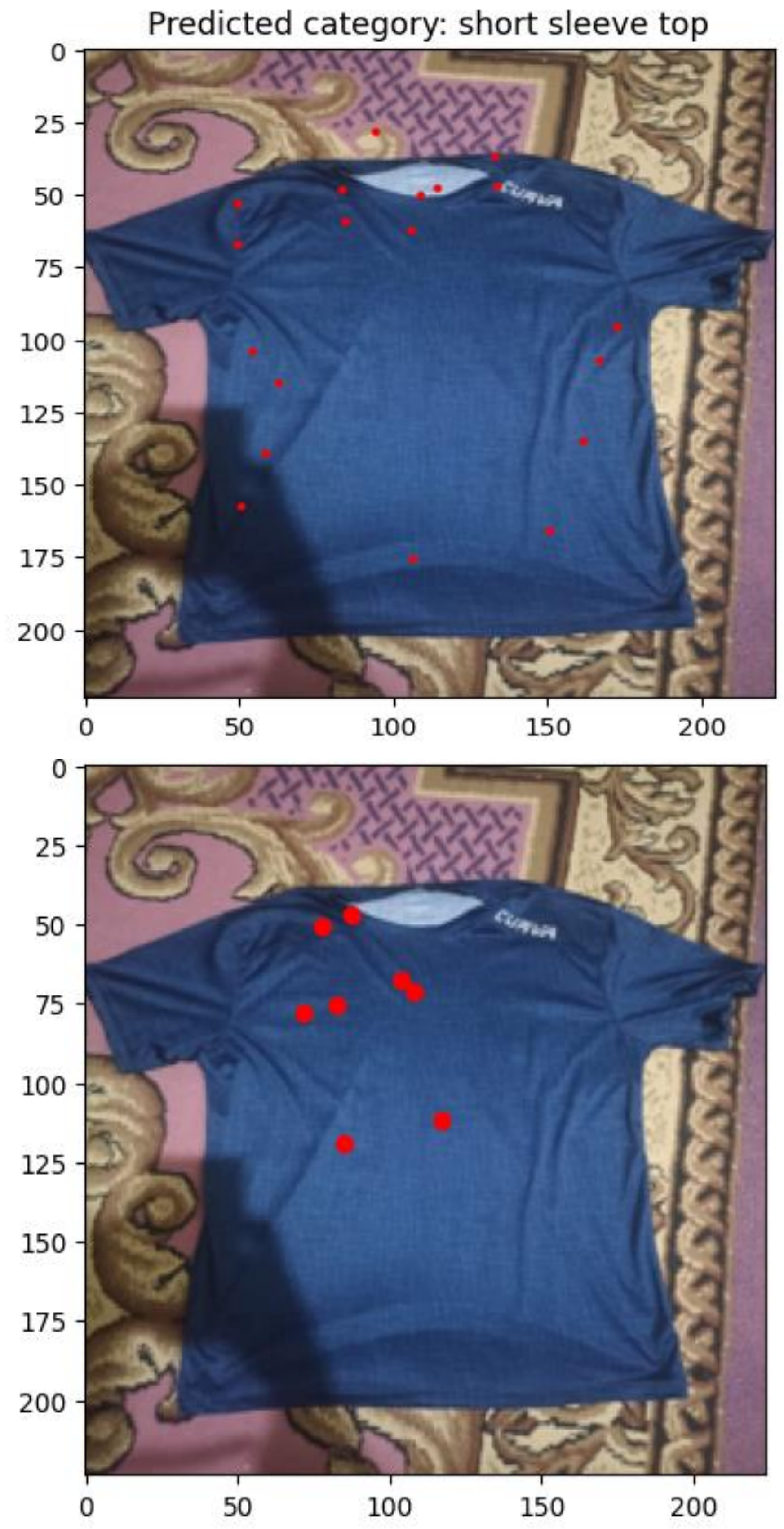
Third Trial:

A pre-trained ResNet50 model was used to extract key points and classify categories simultaneously. The best results were a loss of 0.085964 and a category accuracy of 88.3%, achieved after 40 epochs with a learning rate of 0.0001..

Fourth Trial:

ResNet50 was used for specific key-point detection with the following results:

- Learning rate of 0.001 and 5 epochs: validation loss was 794.32.
- Learning rate of 0.001 and 15 epochs: validation loss decreased to 598.56.
- Learning rate of 0.003019 and 10 epochs: validation loss increased to 912.74.

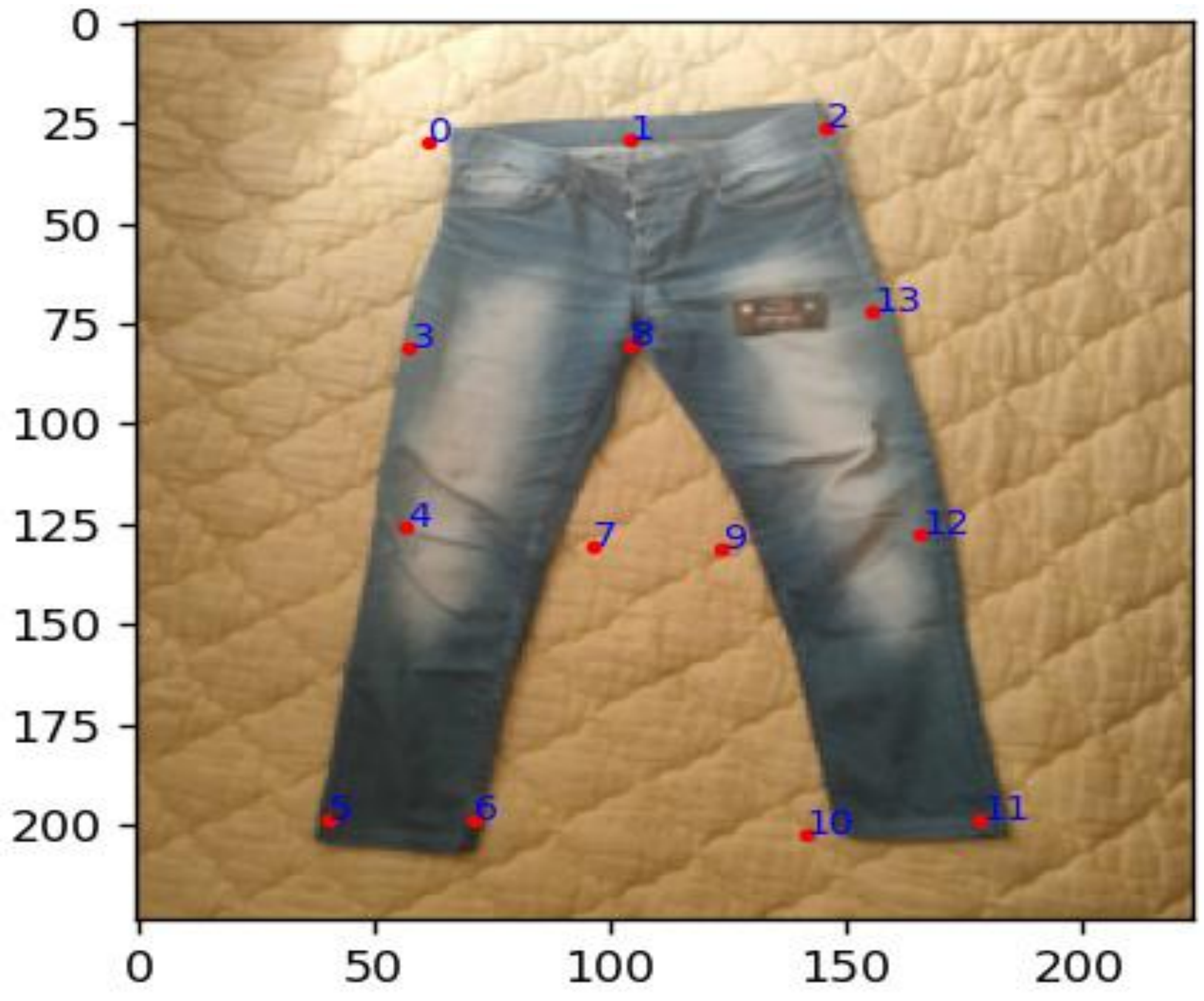


Key-points Detection

Fifth Trial:

13 pre-trained ResNet50 models were employed to extract key-points for

each category.



Key-points Detection

Target	Architecture	Validation MSE Loss
Single Category	Basic CNN	e=5 -> 7000+
Single Category	MobileNetV2	e=10 -> 2000+
Single Category	ResNet50	e=15 -> 598+
Single Category	**13 pre-trained ResNet50**	-
All Categories	Multi-task using pretrained ResNet50	e=40 -> 0.0859

Color Detection

	ColorThief	Mini-Batch KMeans
Published In	2017	2018
Execution Time	3 colors: 0.34 sec 10 colors: 0.37 sec 100 colors: 0.4 sec	3 colors: 0.1 sec 10 colors: 0.3 sec 100 colors: 0.4sec
Accuracy	Accurate	More Accurate

Search by Image

- VGG16 Results weren't satisfying compared to ResNet50 and MobileNet.
- Given that the results of ResNet50 and MobileNet were similar, MobileNet was selected due to its **faster processing speed**.

	VGG16	ResNet50	MobileNet
Feature Extraction time per image	1s to 2s	20ms to 50ms	**20ms to 50ms**
Vector Dimension	25088	100352	**50176**
Cosine-similarity with 5000 images	0.4s to 0.7s	1.5s to 1.8s	**0.7s to 1s**

Mobile application

TA'AM

Team

Mobile Application

Functional Features

The mobile application has been developed with key functional features, including caching for improved performance, state management for efficient data handling, and RESTful APIs for data retrieval. Firebase supports real-time updates, while pagination enhances the handling of large datasets. Secure authentication mechanisms are also in place.

Mobile Application

Non-Functional Features

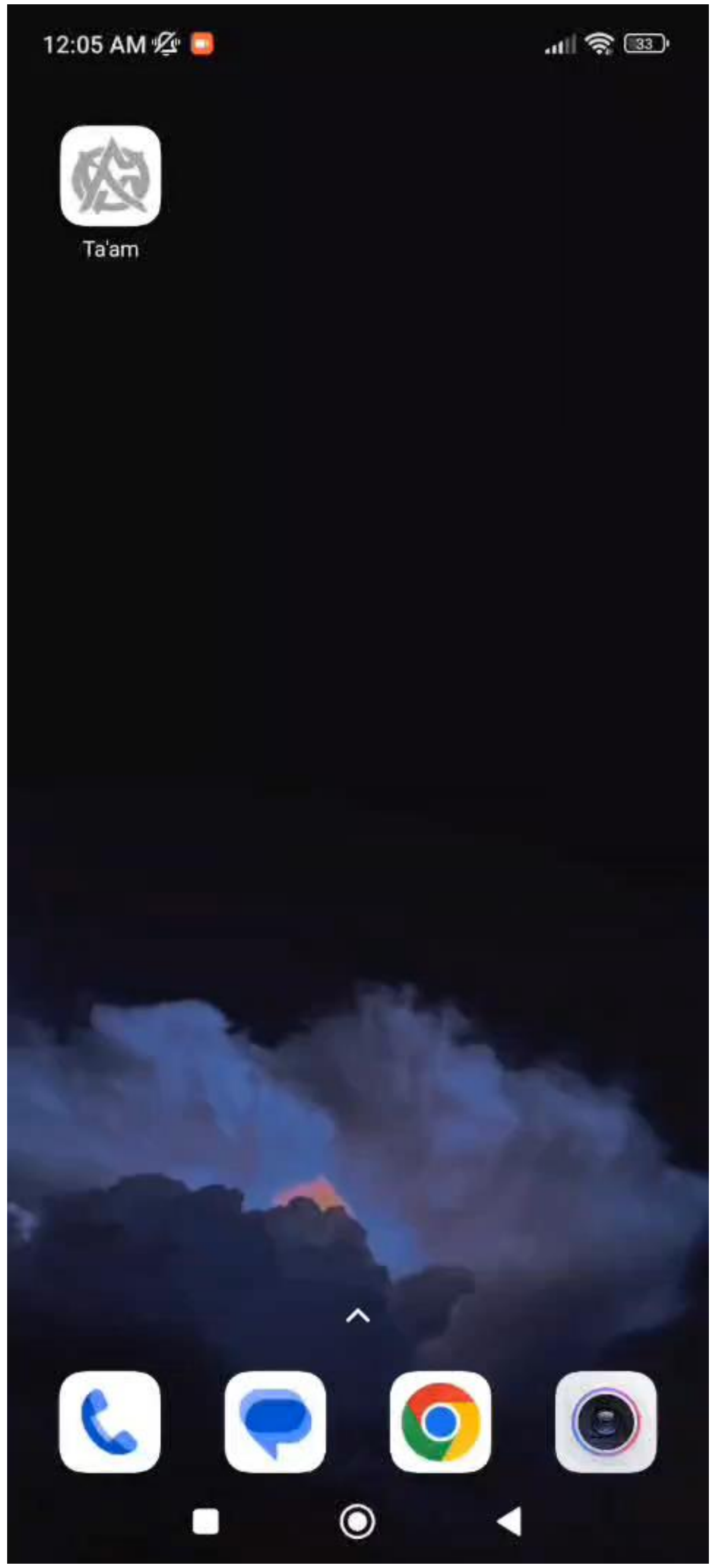
The app's non-functional features include portability, adaptability, usability, and maintainability. These ensure the app runs seamlessly across platforms, adapts to evolving needs, provides an intuitive user experience, and allows for easy updates. This focus guarantees long-term viability and user satisfaction.

Demo

TA’AM

Team

Demo



Conclusion and Future Work

TA'AM

Team

Conclusion

This project uses advanced models to enhance clothing image processing and search. Deep learning identifies details like category, size, and color, while BRISQUE checks image quality. Key point analysis improves size estimation and user experience. ResNet34 is trained on Agrigorev's Custom dataset for classification, and ResNet50 on DeepFashion2 for key-point estimation, enabling effective handling of diverse clothing items.

Future Work

- Improve model accuracy with fine-tuning and larger datasets.
- Adding intelligent recommendation system.
- Integrate depth sensors for better garment sizing, considering accessibility.
- Include tools for environmental impact awareness to support sustainability.

References

TA'AM
Team

References

1. A. Paulauskaite-Taraseviciene, et al., "An intelligent solution for automatic garment measurement using image recognition technologies," Applied Sciences, vol. 12, no. 9, pp. 4470, 2022.
2. B. Lao and K. A. Jagadeesh, "Convolutional Neural Networks for Fashion Classification and Object Detection," 2015.
3. S. S. Islam, E. K. Dey, M. N. A. Tawhid, and B. M. M. Hossain, "A CNN Based Approach for Garments Texture Design Classification," Adv. technol. innov., vol. 2, no. 4, pp. 119–125, May 2017
4. T. Sennikova, "Clothes Classification with the DeepFashion Dataset and Fastai," towardsdatascience, Available: https://towardsdatascience.com/clothes-classification-with-the-deepfashion-dataset-and-fast-ai-1e174cbf0cdc, Accessed: 19/11/2023.
5. M.Ibrahim Mohamed, “CalorieMe Image-based Calorie Estimator System”, 2023