Summary

Category classification

Built torch-based models to predict the **main category** of the product.
(Apparel/Accessories/Footwear/Personal Care/Free Items/Sporting Goods/Home)

Approach

Steps:

1. Data preparation: splitting into train and test data
2. Defining image transforms
3. Creating Datasets using ImageFolder
4. Creating Dataloaders
5. Defining model architecture
6. Creating training loop, model training
7. Evaluation on test data

Experimentation was done with 3 model architectures:

1. Transfer learning using a pretrained resnet50 model

- Added custom classification layers to pretrained feature extractor from resnet50 model
- Achieved a test accuracy of 0.9317

2. Transfer learning using a pretrained resnet18 model

- Added custom classification layers to pretrained feature extractor from resnet18 model
- Achieved a test accuracy of 0.9217

3. Using custom model architecture

- Used a custom model architecture with Conv2D, MaxPool2D and Linear layers
- Achieved a test accuracy of 0.9314

The code for the above can be seen at notebooks/category_classification.ipynb

Finally, the custom model was chosen for inference as part of the API.

Future improvements

- Current models predict the main category of the product, future models can be built to predict more specific features, like the **subcategory** or **article type** of the product.

Color Extraction

Approach

1. RGB value extraction

- Used **class activation maps** to get activations of each pixel in the image
- The class activation map was obtained for the predicted category class, with respect to the category classification model
- Applied a threshold to generate a **segmentation mask,** for highly activated regions of the image
- Applied the segmentation mask on the image to obtain only the necessary pixels
- Applied **K-Means Clustering** to obtain the most dominant colors in the filtered pixels
- The rgb value of the biggest cluster represents the final rgb value extracted from the image
- This is then mapped to the color category, as explained below

2. Mapping RGB value to Color Category

Preliminary steps:

- Hard coded each color category to be mapped to a specific rgb value
- The mapped rgb value was defined using https://web.njit.edu/~walsh/rgb.html
- A **KNNClassifier** was trained on the mapped colors, with **Euclidean distance**

Inference steps:

- Given the rgb value extracted from an image, the **nearest neighbor** was found using this KNNClassifier
- The color category of the nearest neighbor is returned as the final predicted color category for the image.

Performance

- The above approach obtained an accuracy of 0.20, on 20% of the entire dataset (a random subset)

The code for the above can be found in notebooks/color_extraction.ipynb

Current Issues

- The **Euclidean distance** metric is not indicative of the visual similarity between colors
- As a result, the predicted color category is different from the color of the predicted rgb value, hence the poor performance.

Future Improvements

- The below can be tried to improve the performance of the color category mapping
    - Using a **meaningful distance metric** that represents the visual similarity between rgb values.
    - Using a **different color space** instead of rgb, that can be mapped more meaningfully.