Visual Taxonomy - Attribute Prediction Challenge

Team : Overfitted Ogres: Ravindra Kapse , Rajarshi Bhattacharjee

1 Introduction

Our solution integrates various machine learning techniques in an attempt to enhance the product attribute classification accuracy and other metrics. We apply the YOLOv8 to detect and keep the relevant portions of the image. In handling missing values, we applied SWIN Transformer combined with Random Forest to impute them. We planned to use different models to address different aspects of attribute classification. We experimented with a fully connected ResNet to extract deep features; a multi-headed ResNet to process multiple attributes at one go(Multi-task learning); and a ResNet-in-XGBoost to classify slight variations in fine attributes accurately. For submissions we went with convolution-xgb, and with ResNet50s, and ensemble of them.

2 Data Preprocessing

2.1 Duplicate Removal

Duplicate Removal In our preprocessing workflow, as depicted in figure 1 (a), we processed a dataset of 70,214 images across 5 categories and discovered a high incidence of duplicates 1, especially prevalent in the Saree category. To address this issue, we employ a hashing technique to calculate and compare image hashes within each category, successfully identifying and removing duplicates by taking the mode within categories. This method reduced the data set to 45,146 unique images, thus eliminating the risk of data leakage between the training and validation data sets and preventing artificially inflated accuracy metrics.

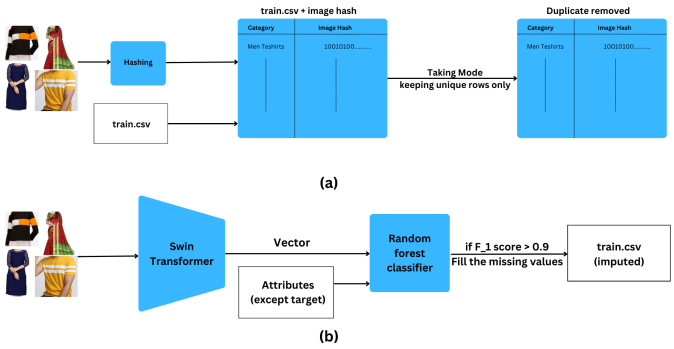

Figure 1: (a) workflow for duplicate removal(Hashing)., (b) workflow for imputing missing values

2.2 Background Removal

Further preprocessing involved background removal from images to enhance model focus on relevant features, as outlined in figure 2. We fine-tuned a YOLOv8n model, originally trained on the ImageNet dataset, for object detection to isolate clothing items from noisy backgrounds. To efficiently train this model, we selected representative samples for annotation by applying k-means clustering and visualizing with t-SNE as shown in the figure 3. This process helped cluster and select 200 images per category, totaling 200x5 = 1,000 images, which were then manually annotated. The annotated set was used to train the YOLOv8n model, ensuring that the cropped images used in training and testing focused solely on the relevant clothing items.

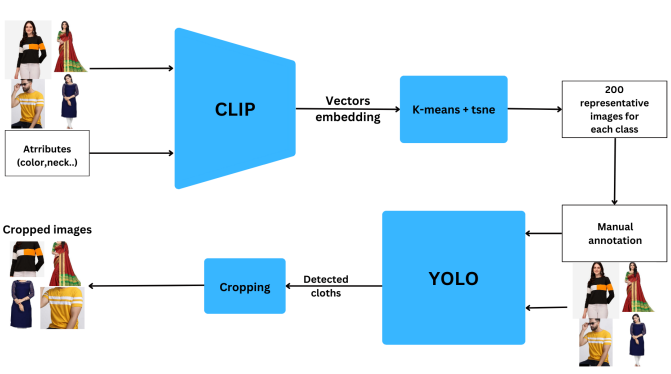

Figure 2: (a) workflow for cropping the images

2.3 Data Imputation

Additionally, as detailed in figure 1(b), our dataset contained 91,763 missing data points (attribute-wise) in train.csv, which we addressed using a combination of Swin Transformer and Random Forest to impute missing attributes. This hybrid approach utilized the Swin Transformer's capability to extract rich, contextual features from images, paired with low variance of Random Forest.

We chose XGBoost for its efficiency in handling categories with implicit patterns, such as the frequent occurrence of short sleeves in polo neck men's t-shirts, a classification readily handled by tree-based models. We selectively imputed attributes where this combined method achieved over 0.9 accuracy, successfully correcting 39,748 data points. This targeted and precise imputation significantly improved the reliability and integrity of our training data, thus improving the robustness of our attribute classification models.

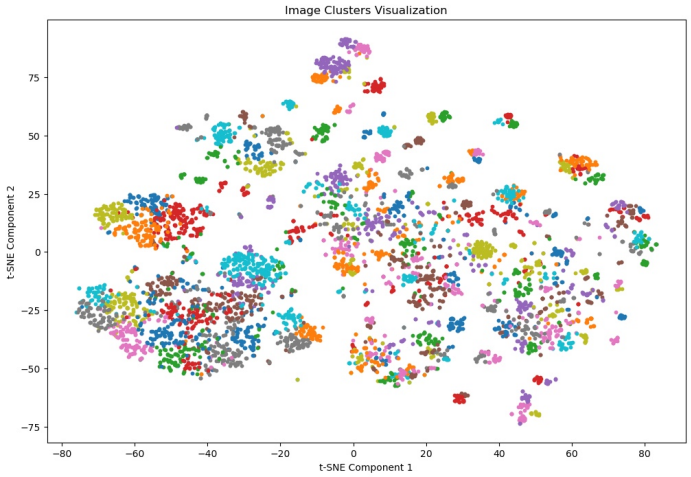

Figure 3: Representative clusters using t-SNE

3 Modeling

3.1 ResNet

In our project, we opted for the ResNet architecture, specifically the ResNet18, ResNet34, ResNet50, and ResNet101 variants, using transfer learning to leverage pre-trained models. This choice was driven by ResNet's robust performance in image classification tasks, attributed to its innovative use of deep residual learning. This approach helps mitigate the vanishing gradient problem, enabling the training of much deeper networks. Pre-trained weights (ImageNetV2) were used.

Architectural Enhancements:

- **Pre-trained ResNet:** Leveraged ImageNetV2 pre-trained ResNet for transfer learning.
- **Dropout Regularization:** Introduced a dropout layer (0.25 rate) to mitigate overfitting.
- **Customized Output Layer:** Replaced the default ResNet output layer with one tailored to the target classes.
- **Activation and Loss:** Used ReLU activation and cross-entropy loss for effective multi-class classification.

Figure 4: Example of duplicate images in training data

Table 1: Overview of Duplicates and Total Entries by Category

Category	Duplicates	Total
Men Tshirts	984	7267
Sarees	17149	18346
Kurtis	2763	6822
Women Tshirts	915	18774
Women Tops & Tunics	3256	19004

Multi-Head ResNet Architecture for Multi-Task Learning

For complex classification tasks that require the simultaneous determination of multiple attributes from a single image, we employed a multi-head ResNet architecture. This approach integrates the robust feature extraction capabilities of the ResNet50 model with the flexibility of multiple specialized classification heads.

Input Image: The network receives an input image, such as a Men's T-shirt, and processes it through the foundational layers of ResNet50. These layers, which extend up to the final convolutional stages, are shared across

all classification tasks. The shared feature representation is computed as:

$$F_{\text{shared}} = \text{ResNet50}_{\text{shared}}(I)$$

where I is the input image. This step extracts generalized features useful for predicting multiple attributes.

Attribute Heads: After the shared layers, the network divides into multiple branches (heads). Each head specializes in a specific attribute, such as color or pattern. The predictions for the i-th attribute are given by:

$$\hat{y}_i = h_i(F_{\text{shared}})$$

where h_i represents the layers of the i-th head.

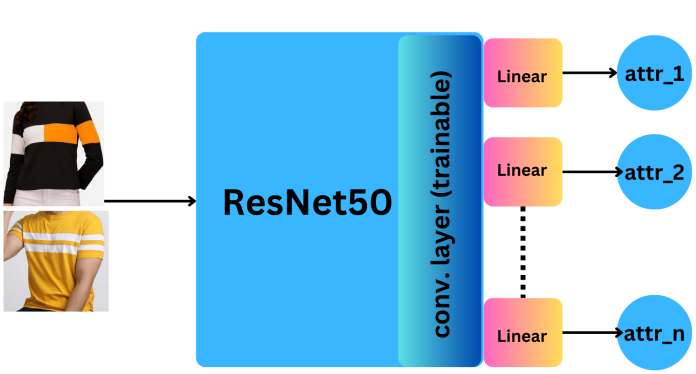

Figure 5: Multihead used for Men Tshirt

Training: Each attribute-specific head is trained using its corresponding ground truth label y_i. The loss for the i-th attribute is computed using a suitable loss function, such as cross-entropy:

$$\mathcal{L}_{\text{attr}i} = \text{CrossEntropy}(\hat{y}_i, y_i)$$

The shared backbone is trained using the total loss, which is a weighted sum of the individual attribute losses:

$$\mathcal{L}_{\text{total}} = \sum_{i=1}^{n} \alpha_i \mathcal{L}_{\text{attr}i}$$

Here, α_i is the weight for the i-th attribute loss, allowing control over each attribute's influence during training.

Inference: During inference, the shared backbone processes the input image to extract generalized features. Each head then outputs its predictions simultaneously:

$$\{\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_n\}$$

This parallel operation ensures comprehensive classification across all attributes.

Advantages of Multi-Head Architecture:

- **Efficiency:** Shared layers reduce redundancy and conserve computational resources compared to separate models for each attribute.
- **Simplicity:** A single model that addresses multiple tasks simplifies both training and deployment. Reducing training of 42 models to 5 models.
- **Improved Learning:** By leveraging shared features, the model enhances its ability to generalize across attributes while benefiting from cross-task learning.

This structured approach ensures that the multi-head ResNet model effectively meets the complex requirements of fashion. In our case we were able to build one and train it for this dataset. Training needs non-null values across all attributes for that category. This was the major hurdle since the dataset is moderately sparse, hence it gives good accuracy for only a subset of the categories.

3.2 ResNet with XGBoost

In our project, we developed a pipeline combining deep learning and machine learning techniques to improve image attribute classification accuracy. This approach integrates ResNet50 for feature extraction with classifiers like XGBoost for precise classification.

Feature Extraction with ResNet50: We use ResNet50, a deep convolutional neural network, to extract robust features from input images. The process includes:

1. **Data Preparation:** Images are resized to the ResNet50 input dimensions and normalized.
2. **Model Configuration:** The final layers of ResNet50 are adjusted to match classification tasks, with dropout added to prevent overfitting.

3. **Feature Extraction:** The modified ResNet50 outputs features representing high-level abstractions of the input images.

Classification with Machine Learning Models: Extracted features are classified using algorithms such as XGBoost, RandomForest, and SVM. Key steps include:

1. **Training:** Classifiers are trained on extracted features, optimized for accuracy and generalization.
2. **Integration:** Classifiers are combined in ensembles, using methods like majority voting to improve prediction accuracy.
3. **Evaluation:** Performance is measured using metrics like accuracy and confusion matrices.

Advantages of the Hybrid Approach:

- **Improved Accuracy:** Combines deep learning feature extraction with effective classifiers for better results.
- **Modularity:** Feature extraction and classification stages can be independently modified and optimized.
- **Scalability:** Suitable for handling large and complex datasets efficiently.

This method has shown great promise in classifying complex attributes from images, as demonstrated by our results in various categories such as Men's T-shirts, sarees, and kurtis, significantly improving over traditional image classification techniques.

3.3 Final Model

Our project utilized the ResNet50 architecture, guided by the model's superior feature extraction capabilities essential for the complex classifications required by our dataset. Detailed modifications and rationale are provided in Subsection 3.1 of this report.

The hyperparameters finalized for our model are:

- **Learning Rate:** Set at 0.001, optimized for quick convergence without overshooting the loss landscape's minima.
- **Dropout Probability:** Fixed at 0.25 to mitigate overfitting, crucial for a deep network like ResNet50.
- **Batch Size:** Chosen as 64 to balance computational efficiency and effective gradient estimation.
- **Epochs:** The model was trained for 20 epochs for each attribute, sufficient for convergence without overfitting, confirmed by plateauing validation accuracy.
- **Unfreeze Layers:** The last two layers were unfrozen to allow the model to adapt more specifically to our dataset's features.

Hyperparameter Fine-Tuning: Hyperparameter tuning involved a mixed approach:

1. **Automated and Manual Adjustments:** A combination of grid search and manual tweaking was employed, especially for the learning rate and layer unfreezing adjustments, to optimize the model's response to training dynamics.
2. **Validation Performance:** The selection of hyperparameters was primarily driven by performance metrics on the validation set, ensuring the model's generalizability to unseen data.

4 Environment and Training Time

4.1 Computing Environment

The model training and evaluation were conducted on a robust computing setup tailored to handle the demands of deep learning workflows. The specifics of the hardware used are as follows:

- **GPU Specifications:** NVIDIA Corporation GA100 [A100 PCIe 80GB] (rev a1).
- **CPU Specifications:** The system was equipped with 32 CPUs.

4.2 Training Time

- **Duration:** The final model training was completed in approximately 1 hour, demonstrating the efficiency of our setup and training procedure.

5 Evaluation Metrics

5.1 Metric & Loss

F1-Score

In the evaluation of our model's performance, the primary metric used was the F1-score. This metric was chosen because it provides a balanced measure of the model's accuracy and robustness, especially in scenarios where class imbalances might affect the performance assessment. The F1-score is particularly valuable as it is the harmonic mean of precision and recall, thereby incorporating both false positives and false negatives into the performance metric.

The weighted F1-score, used to evaluate the performance of our model, can be represented mathematically as follows:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

where:

- **Precision** (P) is the ratio of true positive observations to the total predicted positives, calculated as $P = \frac{TP}{TP+FP}$.
- **Recall** (R) is the ratio of true positive observations to the actual positives, defined as $R = \frac{TP}{TP+FN}$.

The weighted F1-score aggregates the F1-score of each class with the weight reflecting the proportion of actual samples belonging to that class, thus:

$$F_1^{\text{weighted}} = \sum_{i=1}^{n} w_i \times F_1^{(i)}$$

where:

- w_i is the proportion of the number of samples in class i relative to the total number of samples.
- $F_1^{(i)}$ is the F1-score calculated for class i.
- **Class Imbalance Handling:** The weighted average method of calculating the F1-score takes into account the frequency of each class, providing a more accurate measure of model performance across unbalanced datasets.
- **Comprehensive Evaluation:** By combining precision and recall, the F1-score ensures that both the completeness of the model's predictions and their correctness are factored into the evaluation. This is crucial in applications where the cost of false positives and false negatives are significant.
- **Application Relevance:** In many practical applications, especially those involving categorization in diverse fields such as image recognition or customer segmentation, maintaining a balance between precision and recall is vital. The weighted F1-score provides a single metric to gauge this balance effectively.

Using the weighted F1-score allowed for a nuanced understanding of the model's effectiveness in predicting outcomes across multiple categories, making it a suitable choice given the complexity of the tasks our model was designed to perform.

Additionally, we employed the categorical cross-entropy loss function as the optimization criterion during training. This loss function is particularly suited for multi-class classification problems like ours.

5.2 Results

5.3 Error Analysis

5.3.1 Grad-CAM Analysis for Model Verification:

5.3.2 Purpose of Grad-CAM in Our Project:

In our project, Grad-CAM (Gradient-weighted Class Activation Mapping) was employed to visually verify where the convolutional neural network was focusing while making predictions. Specifically, we needed to ensure that

Table 2: F1 Scores by Category and Attribute

Category	Attribute	F1 Score	Mean F1 Score	Category	Attribute	F1 Score	Mean F1 Score
Women Top & Tunics	attr_1	0.79		Kurtis	attr_1	0.80	
	attr_2	0.78			attr_2	0.89	
	attr_3	0.90			attr_3	0.75	
	attr_4	0.84			attr_4	0.92	
	attr_5	0.98	0.87		attr_5	0.93	0.90
	attr_6	0.94			attr_6	0.92	
	attr_7	0.87			attr_7	0.93	
	attr_8	0.93			attr_8	0.95	
	attr_9	0.90			attr_9	0.99	
	attr_10	0.72					
Sarees	attr_1	0.72		Women Tshirts	attr_1	0.86	
	attr_2	0.75			attr_2	0.88	
	attr_3	0.76			attr_3	0.88	
	attr_4	0.61			attr_4	0.97	0.90
	attr_5	0.74	0.75		attr_5	0.76	
	attr_6	0.76			attr_6	0.95	
	attr_7	0.68			attr_7	0.98	
	attr_8	0.77			attr_8	0.93	
	attr_9	0.56					
	attr_10	0.94					
Men Tshirts	attr_1	0.76					
	attr_2	1.00					
	attr_3	0.98	0.93				
	attr_4	0.91					
	attr_5	0.99					

the model was concentrating on the clothing in the images, rather than the background, which can often contain noise and irrelevant objects.

5.3.3 Initial Observations

Initially, the application of Grad-CAM to the model's predictions on raw images revealed that the model was frequently misdirected by the background rather than focusing on the clothing itself. This was attributed to the high presence of background noise and extraneous objects that visually dominated the clothing, which is the actual subject of interest in our analysis.

5.3.4 Intervention with Image Cropping

To address this issue, we decided to employ an image preprocessing step where the clothing was isolated from the background. To achieve this, we utilized YOLO, an object detection system described in Subsection 2.2 of our report, to accurately crop out the clothing part from the images. This step significantly reduced the background noise and allowed the model to focus on the relevant features of the clothing.

5.3.5 Comparative Visualization with Grad-CAM

Below, we present a comparative visualization to demonstrate the effectiveness of our preprocessing step. The images on the left show the model's focus areas before cropping, and the images on the right illustrate the focus post-cropping, clearly showing enhanced model attention on the clothing.

Figure 6: Grad-CAM visualizations showing the model's focus before and after the cropping of images. Left: Before Cropping. Right: After Cropping

6 Conclusion

This project demonstrates a robust pipeline for image attribute classification, achieving significant advancements in model accuracy and efficiency. By integrating advanced preprocessing techniques, such as background and duplicate removal, and leveraging sophisticated modeling approaches like ResNet50 and multi-head architectures, the solution effectively addresses complex classification challenges. The use of Grad-CAM further ensured model reliability by verifying its focus on relevant image areas. Despite the notable improvements, issues like the presence of the 'default' class highlight the need for data refinement to further enhance model performance. Overall, the methods employed showcase a scalable and efficient framework for tackling similar tasks in the future.

7 Appendix

7.1 Analysis of the 'Default' Class in the Dataset

7.1.1 Implications of the 'Default' Class

In our dataset, the 'default' class represents a unique challenge. It does not carry any meaningful distinction in the context of our data. This class appears predominantly across various attributes, where it often coincides with other distinct classes. Upon close inspection, we found that the 'default' value is essentially a conglomerate of features from all classes, which significantly complicates the model's ability to correctly identify and classify the actual meaningful classes.

7.1.2 Case Studies

7.1.3 Category: Sarees

In the category of sarees, particularly for the attribute 'attr_4' (color), the 'default' class frequently mismatches with all other defined classes. This misalignment is depicted in Figure 7, where the 'default' class overlaps with multiple color categories, thereby confusing the model and diluting the predictive accuracy.

7.1.4 Category: Kurtis

Conversely, in the category of kurtis, for the attribute 'attr_1' (color), which contains a total of 13 distinct classes, our model demonstrates a robust capability to classify them effectively. This scenario is illustrated in Figure 7, where the presence of diverse and distinct classes enhances our model's classification performance.

The presence of a 'default' class, which essentially acts as a placeholder or noise within our dataset, undermines the model's ability to discern between truly distinct categories. This class muddles the attribute-specific distinctions critical for accurate classification. Our findings advocate for the removal of the 'default' class from the training process to enhance data quality and model accuracy.

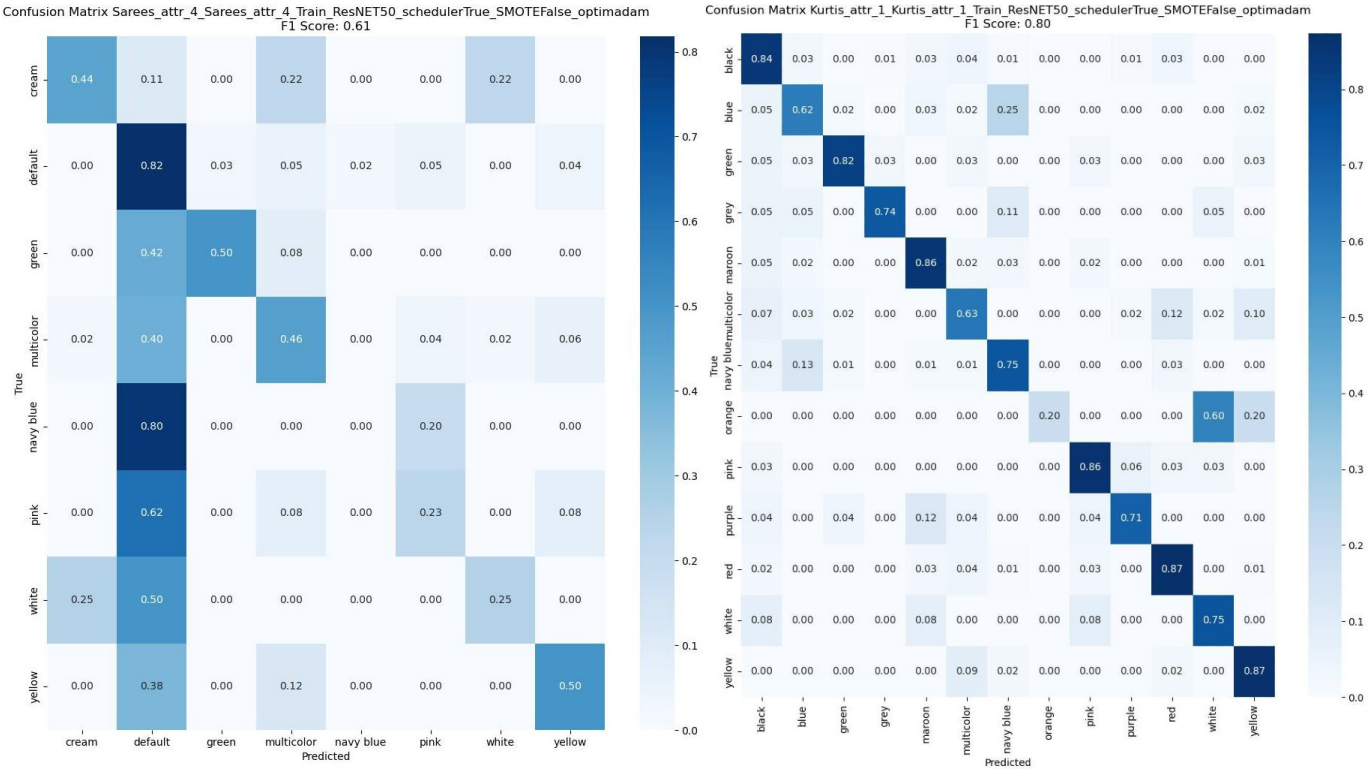

Figure 7: Comparative visualization of class specificity in model classification tasks. The left image shows the class overlap in sarees, while the right image demonstrates effective classification in kurtis.