Fashion Classification*

Shazia Nooruddin, Edgar Leon, Abhinav Sharma

281 Final Project

Fall 2023

Abstract

This project utilizes a subset of the Fashion Product Images dataset from Kaggle to explore a computer vision application that distinguishes between images of pants, shirts, shoes, and watches. Features we have investigated include histogram of oriented gradients (HOG), HSV value histogram feature, the Harris corner feature, SIFT with Bag of Visual Words (BOVW), BRISK, Local Binary Pattern, the RGB histogram feature, the ResNet 101 feature, and sum x-y.. After selecting our features, we implement Uniform Manifold Approximation and Projection (UMAP) to reduce dimensionality in order to achieve generalizability. We input our transformed feature vector in the classification models Linear SVC and K-Nearest Neighbors (K-NN). To optimize further, we consider efficiency by comparing RGB vs. grayscale images in our training set. In our final classifier, we transform our images to grayscale and implement HOG and sum x-y features in a Linear SVC model. We then optimize our classifiers using a hyperparameter search on the validation data, to achieve a final test accuracy of 99.2%.

1 Introduction

In our project, we take a subset of e-commerce data that has been supplied by Kaggle as Fashion Product Images data. Our subset contains 1322 images of shoes, 1249 images of watches, 1139 images of pants, and 1116 images of women's t-shirts, giving us a total of four classes. We begin preprocessing by resizing the images to 60x80. The horizontal and vertical resolutions are 96 dpi with a bit depth of 24. The figure below includes a sample of the labeled images: pants, t-shirt, watch, and shoe, respectively.

Figure 1: Example image from each category

The simple feature extraction techniques we explore include HOG, the Harris corner feature, SIFT, Local Binary Pattern, the RGB histogram feature, and the HSV value histogram feature.

*Please see https://github.com/W281/W281-FashionClassification/tree/main for the links to our notebooks and analyses.

The advanced feature extraction techniques we explore include BRISK, BOVW, VGG, and ResNet101. For classification, we utilize Logistic Regression to narrow down our feature set, and LinearSVC and K-Nearest Neighbors as our final classifiers. In order to speed up training and prediction, along with enhancing generalizability, we implement PCA and UMAP to reduce dimensionality without sacrificing important aspects of the images. We begin the rest of this report by examining the features explored, results from classification and dimensionality reduction for generalizability and efficiency, and accuracy results after hyperparameter tuning.

2 Feature Extraction

This section explores how we selected and extracted features from our images to be used in our classification models. Each feature was tested in a logistic regression model (due to its simplicity) after PCA dimensionality reduction transformation in order to determine if it would be helpful to be used in our final classifiers. We also discuss the theoretical reasoning behind why each feature was chosen. The table that depicts overall accuracies from testing each feature individually through logistic regression is below. All accuracies are from prediction on the validation set.

Feature	**Type**	**Validation Accuracy**	**F1 Score: Pants**	**F1 Score: T-Shirts**	**F1 Score: Watches**	**F1 Score: Shoes**
HOG	Simple	97.1%	97%	97%	95%	100%
HSV (Value) Histogram	Simple	45.3%	37%	36%	38%	72%
Harris Corner	Simple	55.7%	67%	41%	54%	59%
SIFT with BOVW	Complex	68.8%	86%	49%	49%	86%
BRISK	Complex	47%	20%	11%	53%	97%
RGB Histogram	Simple	51%	44%	44%	47%	67%
ResNet101	Complex	79%	75%	82%	89%	71%

Table 1: This table illustrates the accuracy and F1 scores for PCA transformed features through a preliminary logistic regression model.

2.1 Histogram of Oriented Gradients (HOG)

HOG is used to detect edge orientation magnitude and direction in a cell after dividing an image into cells, It is known to be robust in determining object structure and texture.

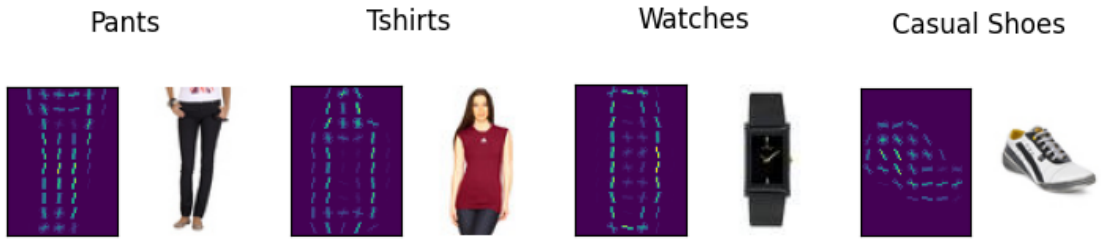

Figure 2: The following figure shows HOG features for each sample image.

Figure 2 illustrates how HOG represents each of the four classes. For pants, it clearly depicts four columns of vertically oriented gradients representing the edges of the pants, thereby preserving its shape. T-shirts and watches both depict rectangular shapes. They seem to be differentiated by the presence of the person wearing the t-shirt. We explore these class similarities more using other features. Shoes are represented by diagonal gradients within the center of the image, also preserving the shape of the shoe, which appears to be a right shoe posing at a diagonal angle. Because the shoe images in our dataset are in the same pose, this is an encouraging sign for using this feature. We test whether our hypothesis about HOG being a useful feature for our model by implementing it in a preliminary logistic regression model. As seen in table 1, results indicate an accuracy on the validation set of 97.1%, with shoes having an F1-Score of 100%, pants and t-shirts having a score of 97% each, and watches having a score of 95%.

2.2 HSV (Value) Feature

HSV (Hue Saturation Value) is often used to represent color information when there is dense directional information over an image. Hue represents the type of color, saturation describes the intensity or vividness of a color, and value depicts the brightness of a color. We find that hue and saturation are not helpful features during exploration because clothing can come in a variety of colors and intensities. Instead, we explore value with the assumption that t-shirts and shoes may be brighter than pants and watches.

Figure 3: The image above illustrates the value representation of the watch class.

After testing our theory in a logistic regression model, we achieve a low accuracy of 45.3%, with shoes having an F1-score of 72% but the rest of the classes having scores of 38% or less. This can be explained by the pants, t-shirts, and watches images having similar brightnesses to each other, while shoes are much brighter than the rest of the classes due to the positioning of the shoe

in the center of the image with lots of whitespace around the shoe. Consequently, we do not move forward with this feature.

2.3 Corner Features

2.3.1 Harris Corner Feature

The Harris feature takes the differentials of a corner score with respect to the direction directly. By taking into account intensity in different directions from a pixel, this method is particularly useful for corner detection.

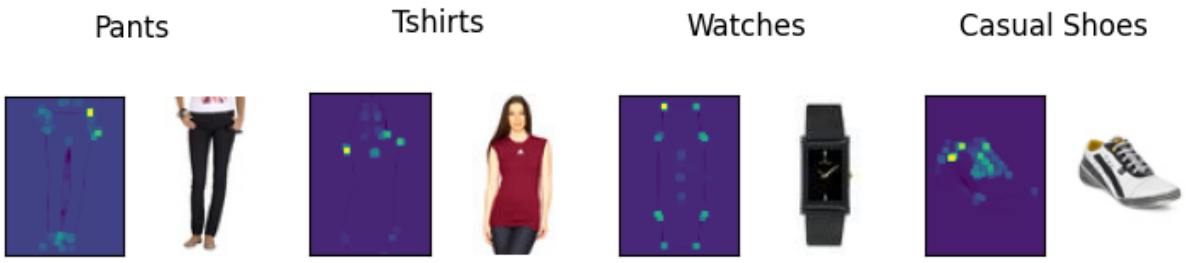

Figure 4: The figure above illustrates how the Harris feature detects corners in each class.

Figure 4 shows how the Harris feature identifies corners within each class. For pants, the crotch area is highlighted, along with each side of the waistband and the ankle ends. For t-shirts, the shoulder corners are found, but it appears to also identify more from the person in the image than desired, such as her hands and jaw. Each corner of the watch and band are clearly identified, but so are the watch dial, which may be problematic. With shoes, the feature identifies each lace corner along with the toe tip areas. In the logistic regression model, we get an accuracy of 55.7%, with F1-scores of 59% for shoes, 67% for pants, 41% for t-shirts, and 54% for watches. These results are not promising, possibly due to the nature of how rectangular three of the classes are, how the crotch area for pants is not clearly visible in all images, how humans are sometimes present in t-shirt images, and how watches have different dial patterns within them. We do not move forward with this feature.

2.3.2 SIFT with Bag of Visual Words (BOVW)

We also examine SIFT, which is another corner feature detector that finds key points robust to changes in scale, rotation, affine transformations, illumination, and projective transformation. SIFT is scale and rotationally invariant, while Harris is only rotationally invariant.

Figure 5: The figure above demonstrates the key corner points that SIFT identifies.

Like the Harris corner detector, the SIFT detector identifies the crotch of the pants and the ankles, but it does not point out the waistband corners. The sample image for the t-shirt in figure 5 does not include a person like in figure 4, but in both shoulders are key points. However, SIFT recognizes one corner end of a t-shirt, while Harris recognizes none. For the watch image in figure 5, mostly key points within the dial are identified, which appears less robust than Harris. For casual shoes in figure 5, the shoes do not have laces and many areas are identified as corners even though they are not. Running SIFT with logistic regression reveals a very low accuracy of 28.3%, with F1-scores of 0% for shoes, 23% for pants, 48% for t-shirts, and 51% for watches.

We attempt to remedy this feature by implementing it with BOVW in the more advanced K-NN classifier. BOVW represents images by histograms of their frequencies by extracting a feature from an image (SIFT for us), clustering the images through K-NN, assigning the SIFT features in an image to the nearest visual word cluster, and counting the frequency of each visual word in the histogram. Results include an accuracy of 68.8%, with t-shirts and watches having accuracies of 49%, and pants and shoes having accuracies of 86%. Although this is a huge improvement for pants and shoes compared to without using BOVW, the overall performance is still pretty low, leading us to conclude that this feature would not help our final classifier.

2.3.3 Binary Robust Invariant Scalable Keypoints (BRISK)

The BRISK algorithm constructs the feature descriptor of an image through the grayscale relationship of random point pairs in the neighborhood of the local image. It samples pixels over rings after Gaussian smoothing. Like SIFT, BRISK is scale and rotationally invariant. It is more efficient than Harris through its use of binary tests for feature description, and it is more adaptive in challenging environments that have variations in noise without any need for tuning.

Figure 6: This figure illustrates the result of key point detection using BRISK.

Figure 6 illustrates how BRISK, like the other corner detection features, is able to detect the crotch and a waistband point for pants, dial attributes for watches, and toe tips for shoes. For this particular example, BRISK gets lost in the words present on the shirt. However, results from the logistic regression model indicate an accuracy of 47%. While this seems low, broken down by class, the F1-scores are 97% for shoes, 20% for pants, 11% for t-shirts, and 53% for watches. Because of the high F1 score for shoes, we thought it would be advantageous to include this feature in our classification work, in order to differentiate shoes further from the other categories.

2.4 Local Binary Pattern (LBP)

LBP is used to distinguish between textures. It takes low computational power, is invariant to image rotation and scale, and resists fluctuations in image grayscale values. It works by providing thresholds to neighboring pixels based on the current pixel value and assigns a binary number at each pixel. Because each of our four classes have different textures, with pants usually

being denim, shirts being smooth cotton, watches having rigid wristbands, and shoes having sloping, we examine the usefulness of this feature in our feature vector.

Figure 7: The image above portrays how texture is represented in the four classes through LBP.

The textures in figure 7 are difficult to distinguish between classes. Consequently, we did not move forward with this feature.

2.5 RGB Histograms Feature

The next feature we explored was RGB histograms. RGB histograms illustrate the frequency of color intensities within the three color channels of an image, showing how much of each color is present. Because we assumed darker colors would be used for pants and a more variety of color would be present in t-shirts, we felt this feature was worth visiting.

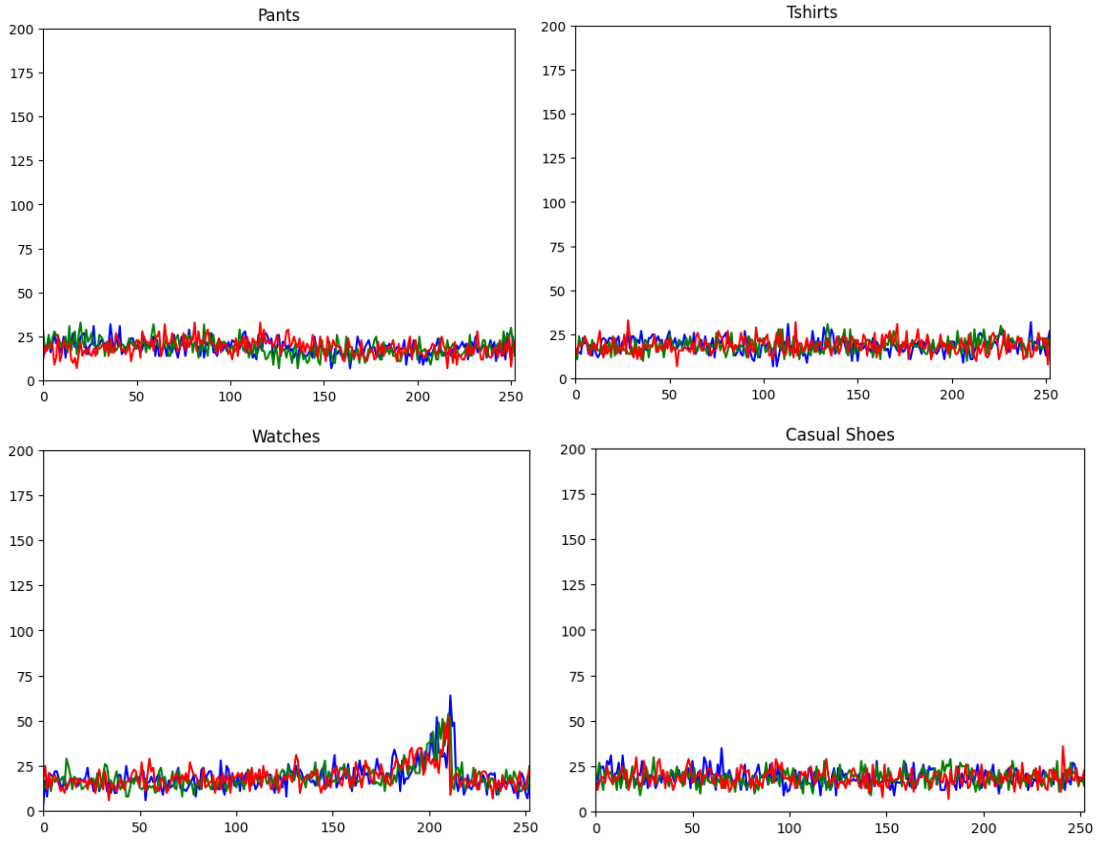

Figure 8: The plots above illustrate the mean RGB Histogram feature for each class.

Figure 8 reveals that many colors are present in all classes and are distributed pretty evenly except for watches. For watches, the peak frequency for RGB is around 200, which indicates a high presence of the color gray. This is likely an artifact of the background color present in the watch images. Nevertheless, we ran the logistic regression model and got an accuracy of 51% with F1-scores of 67% for shoes, 44% for pants and t-shirts, and 47% for watches. Because these scores are not favorable, we did not move forward with this feature.

2.6 Feature Embeddings from ResNet101

Residual Network 101 (ResNet101) is a pre-trained convolutional neural network that has 101 layers. It utilizes deep stacks of residual blocks and skip connections to learn from previous layers in order to extract complex features from images. Because ResNet101 is commonly applied in image classification, we found it appropriate to explore with our image dataset. In the logistic regression model, ResNet101 achieved an accuracy of 79% with F1-scores of 71% for shoes, 75% for pants, 82% for t-shirts, and 89% for watches. The expectation of ResNet101 would be to achieve a higher accuracy score so we examine the off-axis results.

Figure 9: The figure above illustrates off-axis image classification to highlight common mismatches between t-shirts and watches.

The most confusion appears between light colored watches and light colored t-shirts with a human present, as shown in figure 9. Beyond the light color, another explanation could be the white space present above watches and human heads, which is not present in images of t-shirts without people in them. Because it has the second highest accuracy of all the features, explore this feature further in some classification models.

2.7 Sum X-Y

This is a feature we visited at the final stages of our investigation. The need arose to distinguish images by their profile pattern at the edges of their image. By first converting an image to black and white, and then adding the sum of its pixels across the X-axis the resulting graph will give us information about the left and right edges of the image. As seen in figure 10, for pants we expect the left edge to sum to a very high number, as all the pixels in the column are 255 (white), as we encounter the object, some pixels become zero (black) and therefore the sum drops. As we reach the other end of the image the sum again increases, as there is no object present, and all the pixels in that column are 255 (white). Each category in our dataset has an average distinct profile with regards to their location within the x-axis. We do the same for the y-axis and concatenate the x and y axis vectors. Training a LinearSVC classifier on the Sum XY vectors alone, we get an accuracy score of 90% and F1 of 85%. We add these vectors in the last section of our classifying model to increase our overall accuracy.

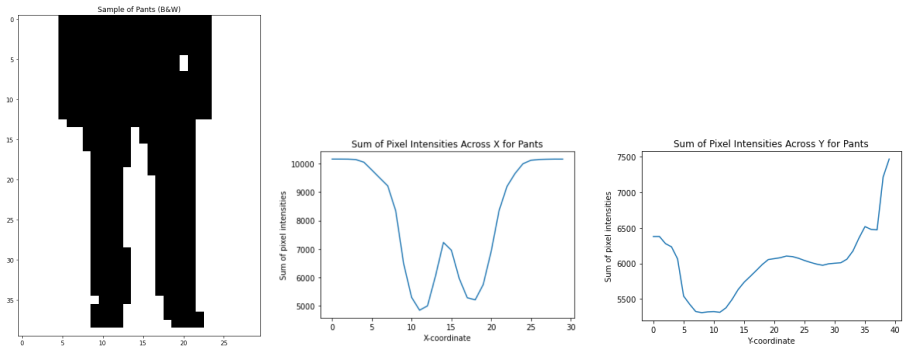

Figure 10: This figure illustrates the sum x-y feature applied to an example of pants.

2.8 Dimensionality Reduction

2.8.1 Principal Component Analysis (PCA)

The purpose of using PCA is to increase generalization in a machine learning application by reducing dimensionality of the data through a linear technique. Generalization is increased through reducing overfitting to the training data, reducing noise in the images, and decreasing the size of data enhances the efficiency during training and prediction. It works by examining the global covariance of the input data, calculating the eigenvalues, ranking the eigenvectors based on their values, and from then selecting the principal components that capture different variabilities within the data. For our first preliminary analysis, we implemented PCA. For our more advanced classifiers, we switched to using UMAP, which is described in the next section.

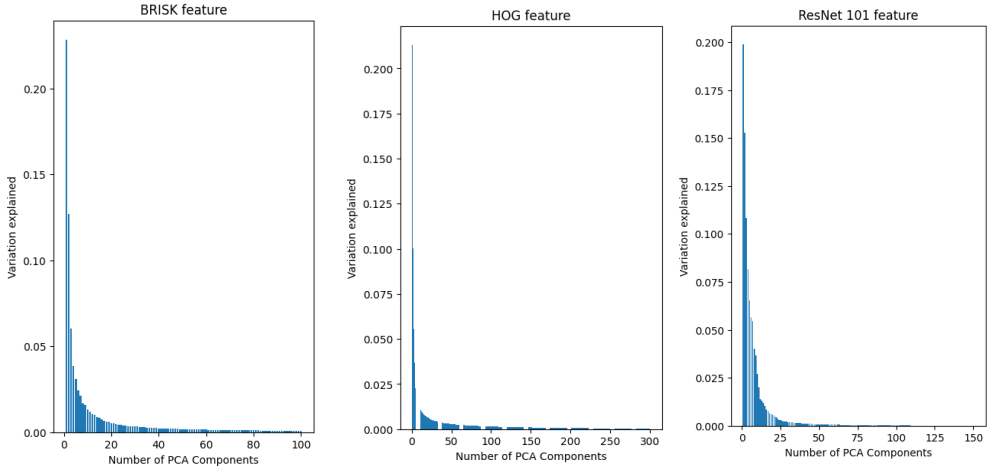

Figure 11: The following plots illustrate the amount of variation explained by each feature given dimensionality reduction by PCA.

2.8.2 Uniform Manifold Approximation and Projection (UMAP)

UMAP is a non-linear dimensionality reduction tool that preserves local and global image characteristics for high-dimensional data. It optimizes the representation of the image using fewer dimensions by minimizing a cost function that calculates the mismatch between pairwise similarities in high and low dimensional spaces. It uses parameters like number of neighbors and

distance to balance between preserving local and global characteristics of the image. We chose to explore UMAP due to its ability to handle more complex data in comparison to PCA despite the added computational cost.

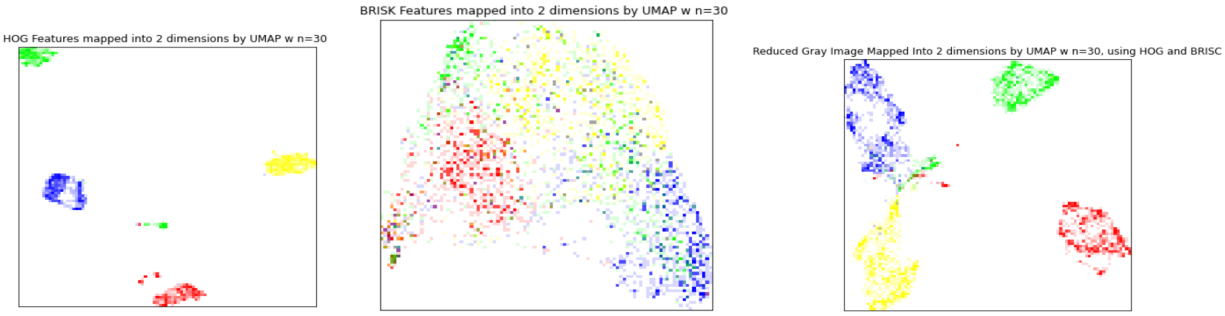

Figure 12: In the figure of UMAP plots above, the first plot depicts the HOG feature, the second plot depicts the BRISK feature, and the third plot shows HOG and BRISK together, all mapped into two dimensions.

As seen in figure 12, when HOG is mapped into two dimensions by UMAP, the four classes are clearly separated. In these examples, Pants are depicted as red, t-shirts are green, watches are blue, and shoes are yellow. For BRISK, there is more confusion, but there are still certain regions that appear split by class, for example a broad region appears to be overwhelmingly dominated by yellow (shoes), therefore we keep considering BRISK as a useful feature. The last plot shows the separation between the four classes with HOG and BRISK together, which is not as good as HOG alone, but is still very robust.

3 Classification

This section portrays multiple classification models trained on the fashion dataset. Features included in the models are HOG, BRISK, Average Intensities, and ResNet 101. Models are compared based on efficiency through time and accuracy following hyperparameter searches using GridSearchCV and cross validation. One path of exploration uses PCA for dimensionality reduction and the other uses UMAP.

3.1 Preliminary Results - Google Colab

3.1.1 PCA Transformed HOG and ResNet101 with Logistic Regression and Linear SVC

Logistic regression models were trained during feature selection using PCA. Logistic regression models are commonly used classifiers that model the probability of an instance belonging to a specific class. It defines a linear decision boundary to separate the feature space into regions corresponding to the four classes. We chose this model due to its simplicity, efficiency, and versatility. Since HOG was the best feature, and other features decreased accuracy, we experimented with the logistic regression model only including HOG.

Features	Model	Validation Accuracy	Best Hyperparameters	Training Time	Prediction Time
HOG	Logistic Regression	98%	C=100, Penalty=L1, Solver=Saga	4.62 s	0.016 s
HOG	Linear SVC	98%	C=10, Gamma=0.01, Kernel=RBF	0.26 s	0.23 s
HOG + ResNet101	Linear SVC	97.8%	C=10, Gamma=Auto, Kernel=RBF	0.21 s	0.11 s

Table 2: Results using PCA transformed HOG and ResNet101 with Logistic Regression or Linear SVC showing hyperparameter tuning and efficiency analysis.

Per table 2, after hyperparameter tuning, this model achieved a test accuracy of 98%. Training took 4.6182 seconds and prediction took 0.0159 seconds. The best hyperparameters were C=100, L1 penalty, and the Saga solver.

The second model we chose to explore using PCA transformed HOG was Linear SVC. Support Vector Machine classifiers work to identify a hyperplane that maximizes the margin between the four classes. It is particularly effective in high dimensional applications and when the data is not linearly separable. We chose this model due to its ability to classify without the assumption that the data is linearly separable. As seen in table 2, after hyperparameter tuning, this model also achieved an accuracy of 98%, but training time was much shorter at 0.2558 seconds and prediction time was longer at 0.2252 seconds. The best hyperparameters were C=10, a gamma of 0.01, and the RBF kernel.

Because ResNet101 was the second best performing feature, we attempted to include it in our classifier to experiment with accuracy and efficiency. As illustrated in table 2, after hyperparameter tuning, our Linear SVC model with HOG and ResNet 101 achieved an accuracy of 97.8%. Training took 0.2069 seconds and prediction took 0.1149 seconds. The best hyperparameters were C=10, gamma=auto, and kernel=RBF. Although efficiency increased, we felt it did not increase by much, and that accuracy outweighed efficiency here. Hence, we did not continue using ResNet101.

3.2 Advanced Results - Local Computing Environment

Grayscale	UMAP	UMAP (sec)	Features	Model	Validation Accuracy	Hyper-parameter search	Training Time (Classifier)	Total Training Time
No	No	x	Image Pixels	Linear SVC	98.4%	Y	818 s	818 s
Yes	No	x	Image Pixels	Linear SVC	98.1%	Y	70 s	70 s

Yes	Yes	16.8 s	Image Pixels	Linear SVC	93%	Y	0.3 s	16.763 s
Yes	Yes	16.8 s	Image Pixels	K-NN	96%	N	0.002 s	16.762 s
Yes	Yes	18.7 s	HOG, BRISK	Linear SVC	94%	Y	0.3 s	18.73 s
Yes	Yes	18.7 s	HOG, BRISK	K-NN	95%	N	0.002 s	18.7002 s
Yes	No	x	HOG	Linear SVC	98.2%	Y	132 s	132 s
Yes	No	x	HOG, Sum X-Y	Linear SVC	99%	Y	344 s	344 s

Table 3: This table depicts accuracy and training time results from exploring grayscale pixels, UMAP transformation, and HOG, BRISK, and Average Intensity features using models Linear SVC and K-NN.

3.2.1 RGB vs. Grayscale Pixels with Linear SVC

We then explored how images would perform in Linear SVC with RGB 60x80 image files. As seen in table 3, we found an accuracy of 98.4%. We then trained another Linear SVC model but with half the size (30x40) and single channel grayscale images and achieved an accuracy of 98.1%. Since a reduction in data from a vector size of 1x144000 to 1x1200 resulted in almost the same accuracy, we felt it was appropriate to move forward with reduced grayscale images to increase efficiency. Please note: for all the linerSVC models that were trained we ran a hyper-paramater search of sweeping C =1e-3 to 1e4, for a total of 7 values of C. We will explore more on hyper-parameter tuning when we pick our final model.

3.2.2 Linear SVC vs. K-NN using Grayscale Images through UMAP

Next, we implemented UMAP to reduce dimensionality on the grayscale images with the intent of increasing efficiency without compromising accuracy. Applying UMAP directly on the image vectors to get a final vector of 1x2 resulted in an accuracy of 93% in the Linear SVC model.

Since the accuracy for Linear SVC decreased, we decided to experiment with the K-NN classifier. Using Euclidean distance to measure similarity between images, K-NN relies on a voting mechanism from majority classes based on K nearest neighbors in order to classify in scenarios where decision boundaries are non-linear. We chose to explore this classifier due to the hypothesis that our images are able to be separated into clusters easily and due to its efficiency compared to Linear SVC. When applying the grayscale images to the K-NN classifier after UMAP transformation to get a 1x2 vector, the accuracy was greater than that of Linear SVC with a value of 96%, as seen in table 3. This led us to continue to compare the two models in our further investigations. However, this is not our final model as it performs worse than just inputting raw grayscale images into the Linear SVC model.

3.2.3 Linear SVC vs. K-NN using Grayscale Images, HOG, and BRISK through UMAP

Next, we tried implementing these models with two of our selected features from our prior exploration, HOG and BRISK. We chose HOG due to its high performance in the preliminary logistic regression model. We chose BRISK because of its impressive ability to identify shoes. As seen in table 3, the Linear SVC model accuracy increased slightly to 94%, while the K-NN model decreased slightly to 95%. Because these accuracies were still below 98%, we continued our investigation.

3.2.4 Linear SVC using Grayscale Images and HOG without UMAP

We suspected that BRISK may not have been helping the model. We also believed that UMAP may not have been helpful for accuracy, despite the improvement in efficiency, so we tested a model using Linear SVC with only HOG on grayscale images. Table 3 shows that this model achieved a slight higher accuracy as our baseline, 98.2% with a vector input of size 1x2928. The higher accuracy is due to the HOG feature vector allowing us to correctly label more t-shirts correctly. Whereas with only grayscale data as an input, t-shirt images of the following type were mis-labeled as a watch:

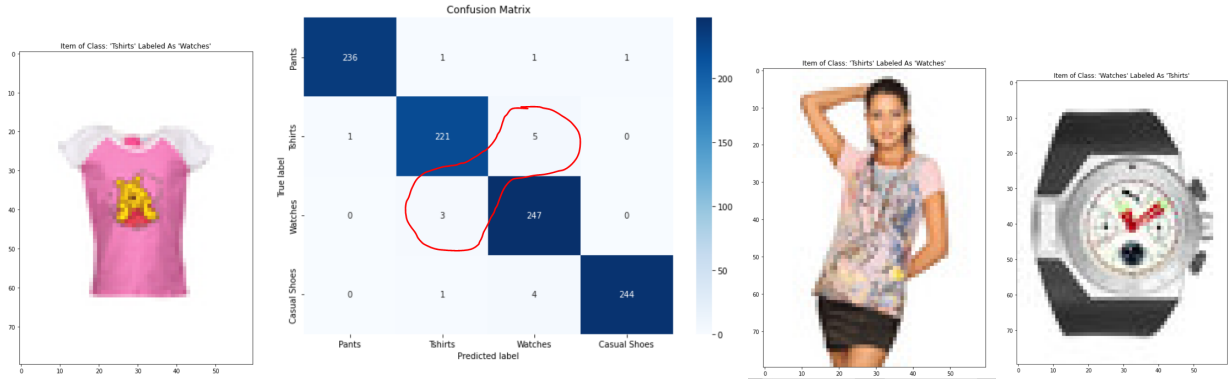

Figure 13: On the left, the image of a t-shirt is an example of the type that was mislabeled as a watch using grayscale data only. The second image is the confusion matrix after including the HOG feature. The last two images depict examples of t-shirts and watches getting mislabeled as each other even after implementing the HOG feature.

In order to improve the accuracy of our classifier we focused on the largest misclassified groups. From figure 13, these misclassified objects are largely composed of t-shirts labeled as watches and watches labeled as t-shirts. Therefore our input vector must include additional information that helps differentiate t-shirts from watches. We considered several options, including face detection, but the most efficient solution appeared to be to exploit the differences in the pixel pattern around the edges of each image. For t-shirts, all the five that were misclassified as watches had empty space at the top of the image, whereas watches did not; we decided to implement the Sum X-Y feature discussed earlier in this paper.

3.2.5 Linear SVC using Grayscale Images, HOG, and Sum X-Y without UMAP

After adding the Sum X-Y feature to try and optimize accuracy, we were successfully able to achieve an accuracy of 99.1% on the validation data without hyperparameter tuning with a final vector size of 1x2998.

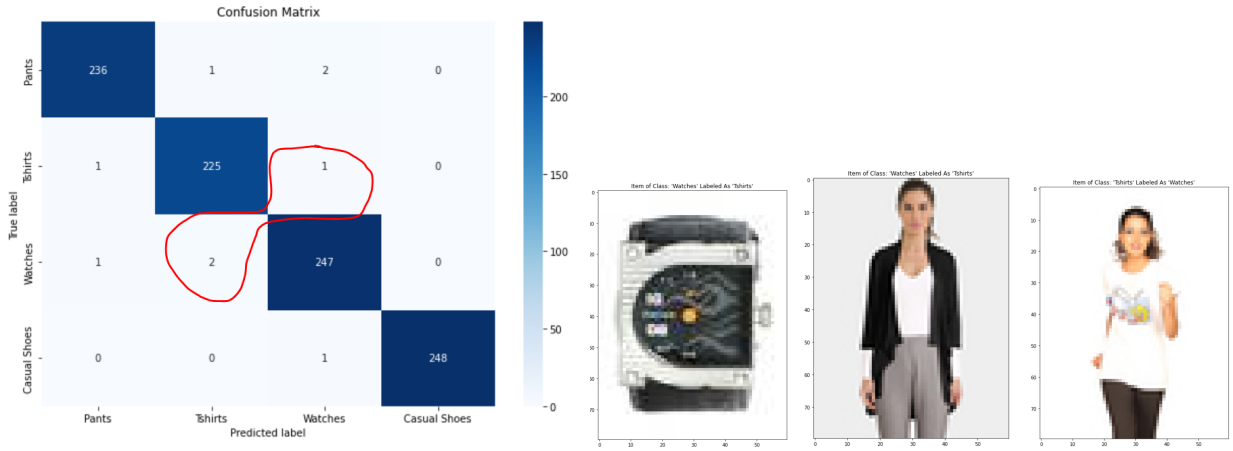

Figure 13: The figure above depicts the confusion matrix on the validation set before hyperparameter tuning for our final model. T-shirts and watches had the most mismatch.

Figure 13 shows that two watches were labeled as t-shirts and one t-shirt was labeled as a watch. The reason for our classifier labeling the watch as a t-shirt is unknown to us. The second image is a mislabeled image in our validation data set therefore we have to dismiss it. The t-shirt that is labeled as a watch we believe is due to the white t-shirt the model is wearing which changes the Sum X-Y profile for this particular image. If we would have implemented face detection functionality it is very likely that the watch and t-shirt would have been clearly distinguished in these cases. We leave that for future exploration.

3.2.6 GridSearch for Hyperparameter Tuning on Validation Set

Next, we implement GridSearchCV using cross validation to determine the best hyperparameters for the Linear SVC model to avoid overfitting on the training data.

Trial	**Model**	**Accuracy**	**Time (sec)**
0	{'C': 1e-05}	0.990674	6.724075078964233
1	{'C': 0.0001}	0.990674	15.451654434204102
2	{'C': 0.001}	0.990674	24.91872477531433
3	{'C': 0.01}	0.990674	34.6297998428344
4	{'C': 0.1}	0.990674	42.88117718696594
5	{'C': 1}	0.990674	50.30094599723816
6	{'C': 10}	0.990674	57.47549319267273
7	{'C': 100}	0.990674	65.27541947364807
8	{'C': 1000}	0.990674	74.29915046691895

Table 4: The table above illustrates the results from GridSearch on the validation set.

The hyperparameter C in a LinearSVC model controls the trade-off between the margin size and the misclassification error. A larger C means that the model will try to find a smaller margin, but it may also overfit the training data and perform poorly on new data. Due to the accuracy not being impacted by the value of C, we choose the smallest C (1e-05) due to the reduced processing time (6.72 seconds) and being most likely to perform better on the test data.

Our final classifier model is as follows:

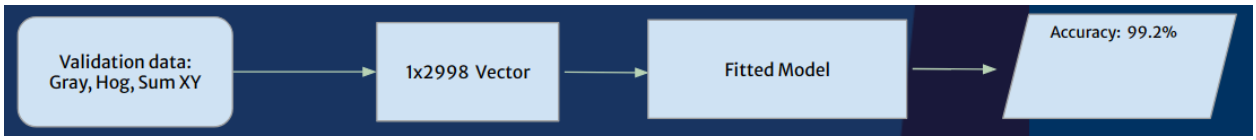

Figure 14: This image shows the flowchart for our final model.

After determining the optimal hyperparameters, our model achieved a final test accuracy of 99.2%. The confusion matrix in figure 15 reveals the same issues as the validation data, with watches being labeled as t-shirts and vice versa. However, one more issue is also seen: a watch being labeled as shoes. However as we inspect the examples, we see that the shoes image is actually a mislabeled wallet, therefore we can dismiss it. Regarding the t-shirt labeled as a watch and vice versa, a face detection algorithm would have helped us in differentiating those two images; however it would have been a computational trade off.

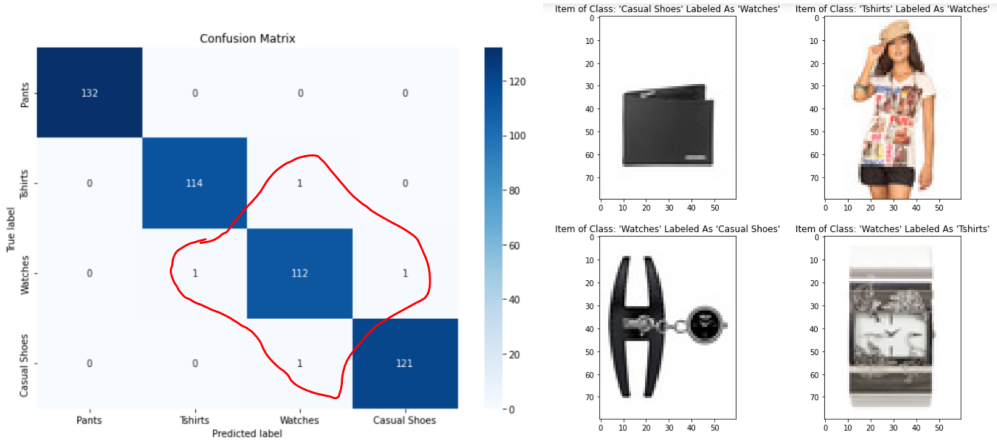

Figure 15: Above is the confusion matrix for our final model after selecting the best hyperparameters. To the right are the mislabeled watches / t-shirts.

Grayscale	UMAP	Features	Model	Validation Accuracy	Test Accuracy	Best Hyperparameters	Training Time	Prediction Time
Yes	No	HOG, Sum X-Y	Linear SVC	99.1%	99.2%	C=1xe-5	6.73 s	0.025 s

Table 5: This table depicts information on our best classifier, including the hyperparameters, efficiency timing, and accuracies.

RGB	UMAP	Features	Model	Validation Accuracy	Best Hyperparameters	Training Time	Prediction Time
Yes	No	Image Pixels	Linear SVC	98%	C=1xe-5	121s	0.82 s

Table 6: This table depicts information on our baseline classifier, including the hyperparameters and efficiency timing, for comparison with our final model.

	Improvement
Accuracy	1%
Training Time	94%
Prediction Time	97%

Table 7: This table depicts our final metrics when comparing our final model with the baseline model.

From table 7 our biggest improvement is in reducing computational complexity: Training Time and Prediction Time.

3.2.6 Efficiency vs. Accuracy Tradeoff

Throughout the classification process, we found that input vector size through dimensionality reduction significantly improved efficiency, but often reduced accuracy. For example, without any dimensionality reduction, we found our models would take hundreds of seconds to train, as seen in table 3. With PCA, in which we reduced our data to 300 components, training time significantly decreased to around 4 seconds with logistic regression, as seen in table 2. Compared with logistic regression, when switching to linear SVC, training time decreased by a factor of about 20 while maintaining accuracy. We also found that adding features that did not perform as well as HOG often decreased accuracy due to the increase in dimensions without adding any useful information. Even though UMAP takes a greater computational cost than PCA, we found that UMAP outperformed PCA in terms of effectiveness due to dimensionality being reduced to 2 instead of 300. But accuracy suffered as a result.

The model with the best accuracy and efficiency was the final linear SVC model that included HOG and Sum X-Y at 99.2 % with a training time of 6.73 seconds and prediction time of 0.025 seconds. Even when compared to the training time of a K-NN model of 0.002 s, overall the final linear SVC is better because once we take into consideration the mapping time required by UMAP (16.76 s) it reduces the efficiency of the overall K-NN classifying system (16.762 s). In summary, we felt that the linear SVC model was the best overall because its training time was better than our baseline, but it was able to achieve almost perfect accuracy and very quick prediction.

4 Generalizability

Our dataset was split into train, validation, and test groups before training with splits of 70%, 10%, and 20%, respectively. During training we implemented hyperparameter searches using GridSearchCV to avoid overfitting to the training set. We focused on tuning accuracy for different experiments by testing on the validation set to avoid overfitting on the training data. Given the high accuracy results on the validation sets, as seen on our final model, we believe we achieved generalizability for images of t-shirts, pants, watches, and shoes, given that the new data is formatted appropriately (white backgrounds, centered). Our team does admit that further investigation is needed to further differentiate t-shirts from watches, specially t-shirt images that include models.

Our training process would be improved by doing further analyses on optimal PCA components, and further exploring the optimal number of neighbors for UMAP and final number of output dimensions. Ideally, when considering performance we would want UMAP to be used in a final model, because it improves performance drastically, so that would be our next goal to strive for. Our training process would also be improved by implementing a face detection algorithm, since we have evidence from this study that it would help separate t-shirts from the other categories in certain specific conditions. Unfortunately, we believed BRISK would help do the same for shoes, but it did not provide fruitful results.

5 Conclusion

In conclusion, we trained three different types of classifiers and achieved an almost perfect result with quick speed in training and testing. We increased efficiency by transforming our images to grayscale, reducing the size of our dataset and the size of each image, and implementing more complex models. We increased accuracy by implementing HOG and Sum X-Y features to enhance our feature set through a complex model and a hyperparameter search. Limitations include that our final model has high dimensionality, and that our model does not distinguish certain objects with white foregrounds from their white background, which may affect generalizability.