BIRZEIT UNIVERSITY

Department of Electrical & Computer Engineering
ENCS5341 - Machine Learning and Data Science

Fashion-MNIST

Prepared by: Ahmad Abbas-1190245, Loor Sawalhi-1190075
Instructor: Yazan Abu Farha
Date: February 12, 2023

Abstract

Many applications, including driverless automobiles and online shopping, heavily rely on image identification and classification. In this project, classification of photos from the Fashion-MNIST dataset used three machine learning algorithms: Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbor (KNN). On the dataset, the algorithms were tested and trained. Various assessment measures, including accuracy and F1 score, were used to measure the algorithms' performance. The data was preprocessed and different features were used for the training process.With an average accuracy of 91.58 percent, the findings demonstrated that the SVM method with HOG and gray scale feature extraction had attained the best level of overall accuracy. The performance of the other algorithms was likewise impressive, with average accuracy ratings of 84.97-89.67%.

Contents

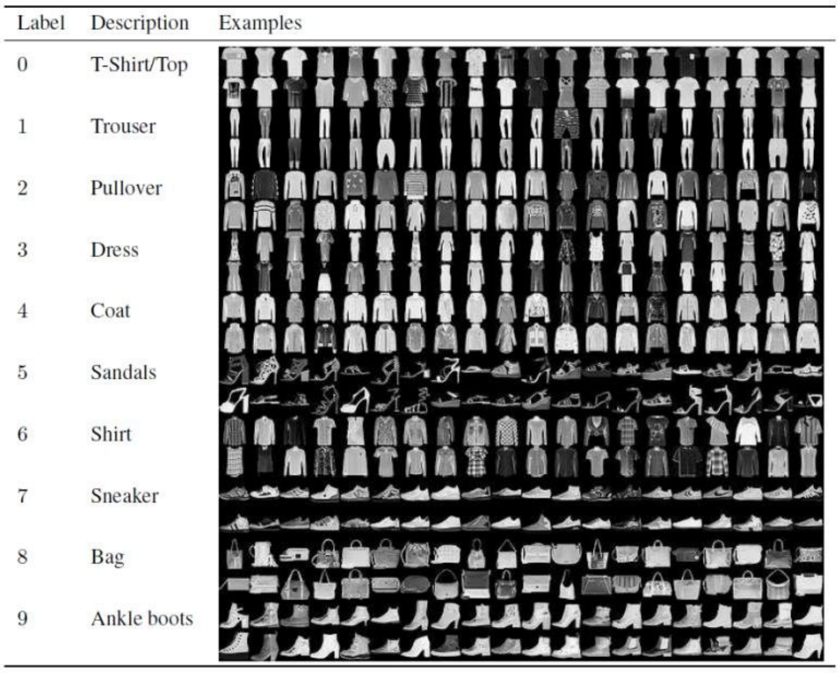
List of Figures

List of Tables

1 Introduction

Image classification is one of the most known applications in machine learning, especially in computer vision. For classifying images into right classes using classification models, feature extraction should be done [1]. In this project, The classification of the Fashion-MNIST dataset is demonstrated using three prominent classifiers, KNN (K-nearest neighbor), SVM (Support Vector Machine) and Random Forrest.

The Fashion-MNIST (F-MNIST) dataset consists of 60,000 samples in the training set and 10,000 samples in the testing set, featuring 10 categories of 28x28 grayscale images of fashion products. As shown in Figure 1.1, the class labels, names, and some sample images of F-MNIST are displayed [2].



Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure 1.1: Fashion-MNIST Dataset Images with Labels and Description

The k-Nearest Neighbors (kNN) algorithm is a straightforward and simple method for classification that has been demonstrated to be effective in various scenarios. To classify a data record t, the algorithm identifies the k nearest neighbors to t and forms a neighborhood for it. The majority vote among the data records in this neighborhood is used to determine the classification for t, with or without considering a distance-based weighting. The choice of the k value is critical for the success of the kNN algorithm, as the performance can vary greatly depending on it. There are various ways to select the k value, but one simple approach is to run the algorithm multiple times with different k values and choose the one that delivers the best performance [3].

Support vector machines (SVMs) are a type of non-parametric, supervised statistical learning technique for binary and multi-class classification problems. The method was introduced to find a hyperplane that separates a dataset into a specified number of classes according to the training examples. The objective of the training algorithm is to find the decision boundary that minimizes misclassifications during the training phase. The learning process involves finding a classifier with an optimal decision boundary to separate the training patterns in a high-dimensional space, and then using this classifier to separate simulation data under similar conditions [4].

The Random Forest (RF) classifier is a combination of tree-structured classifiers that has been advanced from Bagging by incorporating randomness. Unlike traditional decision trees that split each node based on the best split among all variables, RF splits each node based on the best split among a random subset of predictors. A new training dataset is generated from the original data, with replacement, and a tree is grown using this dataset and the random feature selection process. Trees grown in this manner are not pruned , resulting in exceptional accuracy . Additionally, RF is fast, robust against overfitting, and allows for the creation of as many trees as desired by the user[5].

In this project, the performance of the models (KNN, SVM, and Random Forest) was evaluated using two metrics: accuracy and F1-score. Accuracy reflects the average rate of correctly predicted labels among the labels that are either real or predicted [6], while The F1-Score evaluates a classification model's performance by combining Precision and Recall into a single metric using the harmonic mean. This metric is calculated from the confusion matrix. These metrics were used to assess the models' ability to classify the fashion products in the Fashion MNIST dataset accurately [7].

2 Technical Details

2.1 Algorithms and Hyper-parameter

Various machine learning algorithms requires fixed parameters before running, they are referred to as hyper-parameter, the selection of hyper-parameter is crucial such that many learning tasks depends on the selection of them and their performance is affected by their change. In order to obtain the best performance, machine learning practitioners can tune the hyper-parameter [8].

The actual dataset that we use to train the model called training dataset. The model makes observations and learns from this data. The evaluation of a model's fit on the training dataset during the tuning of hyper-parameter is done using a sample of data that is unbiased called validation dataset. Furthermore, the term "testing dataset" refers to the sample of data utilized to conduct an impartial evaluation of the final model's fit on the training data [9].

The algorithms used in this project include: K-nearest neighbor (KNN), Support Vector Machine (SVM) and Random Forest (RF), all of which are implemented in scikit-learn. The parameter ranges for KNN, SVM and Random Forest were obtained form a benchmark on this data set [2] and scikit-learn documentations [10]. In order to determine the best combination of parameters for a given machine learning model, we used to perform an exhaustive search over a specified hyper-parameter space and evaluate their performance using validation set to identify the hyper-parameter that result in the best model.

2.1.1 K-nearest neighbor (KNN)

There are two important hyper-parameter in KNN that need to be set: K and the distance metric. K is an integer that determines the number of nearest neighbors used to make the prediction. A larger value of K will result in a smoother decision boundary, while a smaller value of K will result in a more complex boundary. The choice of K will depend on the nature of the problem and the size of the training dataset. The distance metric is a mathematical function that measures the dissimilarity between two data points. Commonly used metrics include Minkowski distance, which is a generalization of Euclidean and Manhattan distance, and cosine similarity [11].The values for KNN Hyper parameters that will be used can be found in Table 1.

Table 1: Hyperparameters for KNN

Hyper-parameter	Type	Tested
K	integer	{1, 3}
Distance	nominal	{Manhattan, Euclidean }

2.1.2 Support Vector Machine (SVM)

Two significant hyper-parameters, C and kernel, can affect how well SVM performs. The regularization hyper-parameter C manages the compromise between obtaining a low training error and a low testing error. A margin will be greater for a lower value of C, whereas a margin will be narrower for a higher value of C, although at the risk of overfitting. The sort of transformation carried out on the data in order to project it into a higher-dimensional space is determined by the kernel hyper-parameter. The most frequently used kernels are sigmoid, radial basis function (RBF), polynomial, and linear [12].The values for SVM Hyper parameters that will be used can be found in Table 2.

Table 2: Hyper-parameters for SVM

Hyper-parameter	Type	Tested
C	continuous	{1, 10, 100}
Kernel	nominal	{sigmoid, linear, poly, rbf}

2.1.3 Random Forest (RF)

There are several hyper-parameters that can impact the performance of a Random Forest model, including n-estimators, criterion, max-depth, and max-features. n-estimators is the number of trees in the forest and is one of the most important hyper-parameters. A larger number of trees will result in a more robust model but will also take longer to train. The criterion hyper-parameter determines the quality of a split in a decision tree and can be either entropy or gini. Entropy is a measure of impurity in the data, while gini is a measure of the probability of declassifying a randomly chosen element. Max-depth is the maximum depth of each decision tree in the forest, and a smaller value will result in a shallower tree while a larger value will result in a deeper tree. Max-features is the number of features to consider when looking for the best split, and can be specified as either the square root of the number of features or the log base 2 of the number of features [13]. The values for Random forest hyper-parameters that will use can be found in Table 3.

Table 3: Hyper-parameters for Random Forest

Hyperparameter	Type	Tested
n-estimators	integer	{10, 50, 100}
criterion	nominal	{entropy, gini}
max-depth	integer	{10, 50, 100}
max-features	nominal	{sqrt, log2}

2.2 Evaluation Metrics

2.2.1 Accuracy

The accuracy metric provides an overall assessment of the model's ability to correctly predict the output on the complete dataset. Each individual in the dataset contributes equally to the accuracy score, as every unit holds the same weight [7].

2.2.2 F1-Score

The F1-score can be seen as a weighted average of Precision and Recall, with the best F1-score being 1 and the worst being 0. Precision refers to the number of correct positive results divided by the number of all positive results, while Recall refers to the number of correct positive results divided by the number of positive results that should have been returned. Precision and Recall hold equal importance in determining the F1-score and the harmonic mean helps find the optimal balance between these two metrics [7].

3 Experiments and Results

The presented work is implemented in Python using Jupyter notebook using libraries such as scikit-learn for classifiers and matplotlib for plotting.

3.1 Preparing Data and visualization

Preparing data for the Fashion MNIST dataset involves several steps, including importing the dataset into memory, splitting the data into training and testing sets, and preprocessing the data.

3.1.1 Reading data

We read the data using the library *utils/mnist_reader* which came with dataset repository on github [14]. Hopefully, the library reads the data and converts it into NumPy array with splitting it into training and testing data. And we shuffled the data using sklearn *shuffle* method.

3.1.2 Data Preprocessing

The MinMax Scaler was employed as a preprocessing technique to normalize the features in the dataset. The MinMax Scaler scales the features to a specific range, typically between 0 and 1. In order to ensure the scaler was trained on the correct data, the fit_transform method was used during the training phase. This method fits the scaler to the data and then performs the scaling. During the testing phase, the transform method was used to apply the scaling to the test data. This technique is useful in improving the performance of certain machine learning algorithms as it brings all the features to a similar scale, preventing any one feature from dominating the others.

3.1.3 Using Principal Component Analysis to visual data

PCA is a statistical method that helps in reducing the number of dimensions in a dataset while maintaining most of its information. It is widely used for dimension reduction, particularly for high-dimensional data [15]. We used it to plot the data as Figure 3.1:

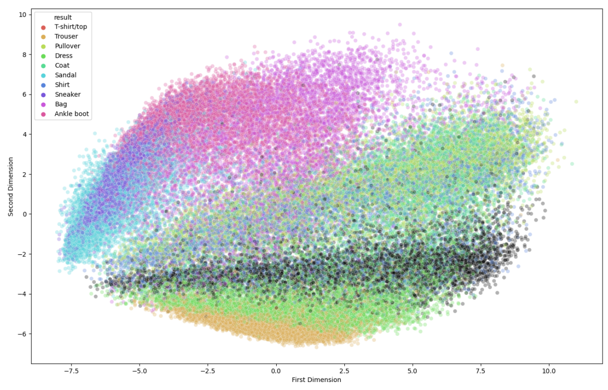


Figure 3.1: Fashion-MNIST Dataset Visualization

3.1.4 Split Data to training and validation sets

The train_test_split function from scikit-learn was being used here to split the training data into two parts: the training data and the validation data and the validation size is set to 1/3, meaning that 1/3 of the data will be used as validation data (20000 samples) and the remaining 2/3 will be used as training data (40000 samples). The random_state parameter was set to 42 to control the random number generator used. We must note that train_test_split splits data while preserving the balance.

3.2 Baseline Model

The k-nearest neighbor (kNN) classifier performed well as a baseline model with an accuracy of 84.97% and a F1-score of 84.9% on the testing data. The best results were achieved when k was set to 3 and the Manhattan method was used. The results indicate that the kNN classifier performed well with the given hyper-parameter combinations.

3.3 Improvements

In an effort to improve performance, we conduct evaluations on two additional models: the Support Vector Machine (SVM) and Random Forest.

3.3.1 Support Vector Machine (SVM)

The SVM algorithm was used for its memory efficiency, and examines the performance of models on the dataset. With the use of the hyper-parameter combinations as previously discussed, SVM demonstrated the highest accuracy with a value of C = 10.0 and an Rbf kernel. The model achieved an accuracy of 89.64% and an F1-score of 89.6% on the testing data.

The improved performance of the Support Vector Machine (SVM) compared to the k-nearest neighbor (kNN) classifier could be due to the nature of the SVM algorithm, which is a linear classifier that finds the optimal boundary between classes [16], and the optimization of its hyper-parameters, such as the value of C and the choice of kernel. These factors allow SVM to work better in high-dimensional spaces and produce better results than kNN, which has fewer hyper-parameters to tune and is a non-parametric method that makes predictions based on the majority class of the nearest neighbors.

3.3.2 Random Forest

Random Forest is a popular machine learning algorithm used for classification and regression tasks. The model demonstrated the highest accuracy through the use of the hyper-parameter combinations as previously discussed, with n_estimators set to 100 and using the entropy criterion. The maximum depth was set to 100 and the splitter for maximum features was set to the square root. The model achieved an accuracy of 87.12% and an F1-score of 86.69% on the testing data.

Random Forest outperforms kNN due to its ensemble nature. Random Forest uses multiple decision trees to make predictions and reduces overfitting, making the model more robust to noise and improving generalization performance [17]. kNN is a simple method that can be negatively impacted by noisy or irrelevant features in the data. The ensemble nature of Random Forest helps to overcome these limitations, leading to improved performance compared to kNN. A full compression can be found on Table 4. Moreover, The confusion matrix of all models can be found on Appendix A.

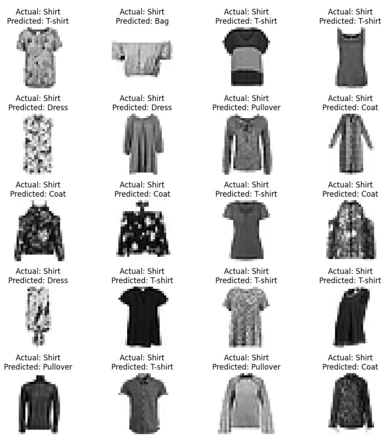
Table 4: Models accuracy and f1-score on Fasjion mnist

Model	Hyper-parameter	Accuracy	F1-score
KNN	K=1, distance = Manhattan	84.97%	84.9%
SVM	C=10.0, kernel=rbf	89.64%	89.6%

Random forest	n_estimators=100 criterion=entropy
	max_depth=100 max_features_splitter=sqrt
		87.12%	86.69%

3.4 Error and Evaluation Metrics Analysis

The SVM model was the best from previous analysis, but the shirt class had lowest accuracy. By selecting 20 random images, Investigation showed that shirt was often misclassified as T-shirt, pullover, or coat, indicating difficulty differentiating similar classes See Figure 3.2.



Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: Bag
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: Dress
Actual: Shirt Predicted: Dress
Actual: Shirt Predicted: Pullover
Actual: Shirt Predicted: Coat
Actual: Shirt Predicted: Coat
Actual: Shirt Predicted: Coat
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: Coat
Actual: Shirt Predicted: Dress
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: Pullover
Actual: Shirt Predicted: T-shirt
Actual: Shirt Predicted: Pullover
Actual: Shirt Predicted: Coat

Figure 3.2: missclassified shirts

Data augmentation helps reduce over-fitting and improve the differentiation between the shirt class and similar classes by increasing the diversity of the training set. This is achieved by transforming existing training images to create new, augmented images.[18]. Results showed that although the number of misclassifications for the shirt class was reduced by nearly 50 samples, the overall accuracy of the model decreased. This was due to the model being exposed to variations in shirts and t-shirts through the augmented data, which was not the desired outcome of the augmentation process. To address this, the Histogram of Oriented Gradient features were added to the data to improve the model.

HOG improves accuracy in Fashion-MNIST. Combining HOG and pixel info benefits gradient and texture information. Adding features to dataset improves model performance. Accurate predictions and classifications result from incorporating more data info. Table 5 supports expectations.

Table 5: Comparison of Grey-scale and HOG+Grey-scale features

Model	Greyscale features	HOG + Greyscale
KNN	Accuracy=84.97%	Accuracy=86.3%
	F1-score=84.9%	F1-score=86.47%
SVM	Accuracy=89.67%	Accuracy=91.58%
	F1-score=89.6%	F1-score=91.61%
Random forest	Accuracy=87.12%	Accuracy=87.97%
	F1-score=86.9%	F1-score=88.13%

HOG is a feature extraction technique in computer vision used to represent image texture and shape. It divides an image into cells, calculates gradient orientations, and aggregates them into histograms as a feature descriptor. HOG is widely used in object detection, pedestrian detection, and image classification due to its ability to capture object information [19].

The use of accuracy as a metric for the task may not always be the most appropriate measure, as it only considers the number of correct predictions and does not take into account the severity of errors made by the model.A more appropriate metric may be F1-score, which takes into account both the precision and recall of the model and provides a more comprehensive evaluation of its performance.As we can see, the new features gave a better F1-score and was higher than accuracy due to the improvements on recall and precision because of the new features.

4 Conclusion

In conclusion, this study has investigated the performance of three machine learning algorithms, Random Forest, Support Vector Machine (SVM), and K-Nearest Neighbor (KNN), for the task of image classification using the Fashion-MNIST dataset. After successful implementation of the proposed classification machine learning algorithms, it has shown that the proposed system provides relatively good fashion object classification efficiency as compared to available literature works. The results showed that the SVM algorithm with HOG and gray scale feature extraction has achieved the highest overall accuracy, with an average accuracy of 91.58%. The other algorithms also exhibited strong performance, with average accuracy scores in the range of 84.97-89.67%.

The report's assessment of KNN, SVM, and RF contains drawbacks, including the possibility for overfitting, restrictions on each method, and reliance only on accuracy and F1-score as measures. The generalization of the model to fresh data might be impacted by overfitting. The methods also have their own drawbacks, such as KNN's high computing cost and SVM's challenges with extremely non-linear data. Relying just on accuracy may not provide a whole view of performance, especially in datasets with imbalances. Precision and recall may be balanced by using the F1-score. The study's findings offer insightful information, but these constraints must be taken into account.

Future improvement and innovation are possible in preprocessing and feature extraction for better image classification results. Alternative techniques and wider range of algorithms may improve fashion image classification. Further analysis of features' impact on classifier performance can enhance accuracy.

In conclusion, this study provides valuable insights into the performance of different machine learning algorithms for image classification on the Fashion-MNIST dataset. The results demonstrate that the combination of SVM with HOG and gray scale feature extraction offers the highest accuracy, but there is still room for improvement in the preprocessing and feature extraction stages. Further research in these areas holds the promise of even more advanced and accurate image classification models.

Acknowledgement

We would like to express our gratitude and appreciation To Mazen Amria for suggesting the combination of HOG and original image.

References

[1] K. Greeshma and K. Sreekumar, "Fashion-mnist classification based on hog feature descriptor using svm," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 5, pp. 960–962, 2019.

[2] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.

[3] G. Guo, H. Wang, D. Bell, Y. Bi, and K. Greer, "Knn model-based approach in classification," in *On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE: OTM Confederated International Conferences, CoopIS, DOA, and ODBASE 2003, Catania, Sicily, Italy, November 3-7, 2003. Proceedings.* Springer, 2003, pp. 986–996.

[4] M. A. Chandra and S. Bedi, "Survey on svm and their application in image classification," *International Journal of Information Technology*, vol. 13, pp. 1–11, 2021.

[5] V. Kullarni and P. Sinha, "Random forest classifier: a survey and future research directions," *Int. J. Adv. Comput*, vol. 36, no. 1, pp. 1144–1156, 2013.

[6] Y. Jiao and P. Du, "Performance measures in evaluating machine learning based bioinformatics predictors for classifications," *Quantitative Biology*, vol. 4, pp. 320–330, 2016.

[7] M. Grandini, E. Bagli, and G. Visani, "Metrics for multi-class classification: an overview," *arXiv preprint arXiv:2008.05756*, 2020.

[8] H. J. Weerts, A. C. Mueller, and J. Vanschoren, "Importance of tuning hyperparameters of machine learning algorithms," *arXiv preprint arXiv:2007.07588*, 2020.

[9] T. Shah, "About train, validation and test sets in machine learning," Jul 2020. [Online]. Available: https://towardsdatascience.com/train-validation-and-test-sets-72cb40cba9e7

[10] "scikit-learn." [Online]. Available: https://scikit-learn.org/stable/

[11] E. Alpaydin, *Introduction to Machine Learning.* MIT Press, 2010.

[12] T. Klatzer and T. Pock, "Continuous hyper-parameter learning for support vector machines," in *Computer Vision Winter Workshop (CVWW).* Citeseer, 2015, pp. 39–47.

[13] M. Schonlau and R. Y. Zou, “The random forest algorithm for statistical learning,” *The Stata Journal*, vol. 20, no. 1, pp. 3–29, 2020.

[14] H. Xiao, K. Rasul, and R. Vollgraf. (2017) Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[15] H. Abdi and L. J. Williams, “Principal component analysis,” *Wiley interdisciplinary reviews: computational statistics*, vol. 2, no. 4, pp. 433–459, 2010.

[16] T. Hastie, R. Tibshirani, and J. Friedman, *The elements of statistical learning: Data mining, inference, and prediction.* Springer, 2009.

[17] L. Breiman, “Random forests,” *Machine learning*, vol. 45, pp. 5–32, 2001.

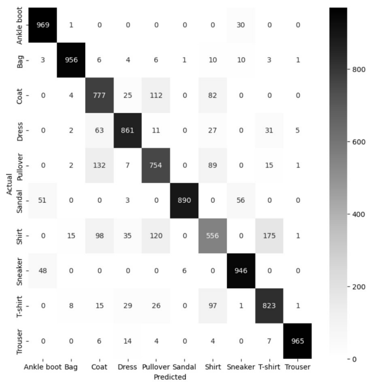
[18] C. Shorten, J. Campbell, and A. Kabán, “A survey on data augmentation in deep learning,” in *2019 19th International Conference on Information Fusion (FUSION)*. IEEE, 2019, pp. 1–8.

[19] N. Dalal and B. Triggs, “Histograms of oriented gradients for human detection,” in *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on.* IEEE, 2005, pp. 886–893.

Appendices

A Confusion Matrix of All models

KNN



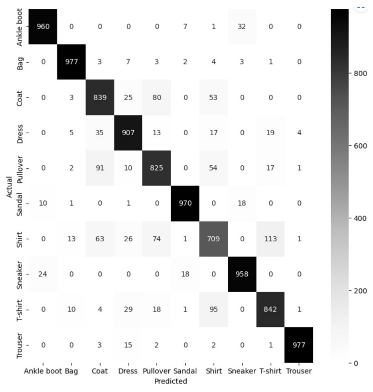
	Ankle boot	Bag	Coat	Dress	Pullover	Sandal	Shirt	Sneaker	T-shirt	Trouser
Ankle boot	969	1	0	0	0	0	0	30	0	0
Bag	3	956	6	4	6	1	10	10	3	1
Coat	0	4	777	25	112	0	82	0	0	0
Dress	0	2	63	861	11	0	27	0	31	5
Pullover	0	2	132	7	754	0	89	0	15	1
Sandal	51	0	0	3	0	890	0	56	0	0
Shirt	0	15	98	35	120	0	556	0	175	1
Sneaker	48	0	0	0	0	6	0	946	0	0
T-shirt	0	8	15	29	26	0	97	1	823	1
Trouser	0	0	6	14	4	0	4	0	7	965

```
              precision    recall  f1-score   support

  Ankle boot    0.90476   0.96900   0.93578      1000
         Bag    0.96761   0.95600   0.96177      1000
        Coat    0.70830   0.77700   0.74106      1000
       Dress    0.88037   0.86100   0.87058      1000
    Pullover    0.72991   0.75400   0.74176      1000
      Sandal    0.99220   0.89000   0.93832      1000
       Shirt    0.64277   0.55600   0.59625      1000
     Sneaker    0.90700   0.94600   0.92609      1000
     T-shirt    0.78083   0.82300   0.80136      1000
     Trouser    0.99076   0.96500   0.97771      1000

    accuracy                        0.84970     10000
   macro avg    0.85045   0.84970   0.84907     10000
weighted avg    0.85045   0.84970   0.84907     10000
```

SVM



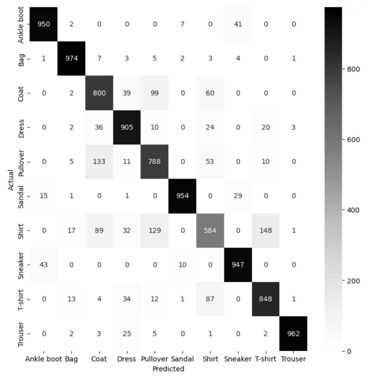
	Ankle boot	Bag	Coat	Dress	Pullover	Sandal	Shirt	Sneaker	T-shirt	Trouser
Ankle boot	960	0	0	0	0	7	1	32	0	0
Bag	0	977	3	7	3	2	4	3	1	0
Coat	0	3	839	25	80	0	53	0	0	0
Dress	0	5	35	907	13	0	17	0	19	4
Pullover	0	2	91	10	825	0	54	0	17	1
Sandal	10	1	0	1	0	970	0	18	0	0
Shirt	0	13	63	26	74	1	709	0	113	1
Sneaker	24	0	0	0	0	18	0	958	0	0
T-shirt	0	10	4	29	18	1	95	0	842	1
Trouser	0	0	3	15	2	0	2	0	1	977

```
              precision    recall  f1-score   support

  Ankle boot    0.96579   0.96000   0.96289      1000
         Bag    0.96637   0.97700   0.97166      1000
        Coat    0.80829   0.83900   0.82336      1000
       Dress    0.88922   0.90700   0.89802      1000
    Pullover    0.81281   0.82500   0.81886      1000
      Sandal    0.97097   0.97000   0.97049      1000
       Shirt    0.75829   0.70900   0.73282      1000
     Sneaker    0.94758   0.95800   0.95276      1000
     T-shirt    0.84794   0.84200   0.84496      1000
     Trouser    0.99289   0.97700   0.98488      1000

    accuracy                        0.89640     10000
   macro avg    0.89601   0.89640   0.89607     10000
weighted avg    0.89601   0.89640   0.89607     10000
```

Random Fores



	Ankle boot	Bag	Coat	Dress	Pullover	Sandal	Shirt	Sneaker	T-shirt	Trouser
Ankle boot	950	2	0	0	0	7	0	41	0	0
Bag	1	974	7	3	5	2	3	4	0	1
Coat	0	2	800	39	99	0	60	0	0	0
Dress	0	2	36	905	10	0	24	0	20	3
Pullover	0	5	133	11	788	0	53	0	10	0
Sandal	15	1	0	1	0	954	0	29	0	0
Shirt	0	17	89	32	129	0	584	0	148	1
Sneaker	43	0	0	0	0	10	0	947	0	0
T-shirt	0	13	4	34	12	1	87	0	848	1
Trouser	0	2	3	25	5	0	1	0	2	962

```
              precision    recall  f1-score   support

  Ankle boot    0.94153   0.95000   0.94574      1000
         Bag    0.95678   0.97400   0.96531      1000
        Coat    0.74627   0.80000   0.77220      1000
       Dress    0.86190   0.90500   0.88293      1000
    Pullover    0.75191   0.78800   0.76953      1000
      Sandal    0.97947   0.95400   0.96657      1000
       Shirt    0.71921   0.58400   0.64459      1000
     Sneaker    0.92752   0.94700   0.93716      1000
     T-shirt    0.82490   0.84800   0.83629      1000
     Trouser    0.99380   0.96200   0.97764      1000

    accuracy                        0.87120     10000
   macro avg    0.87033   0.87120   0.86980     10000
weighted avg    0.87033   0.87120   0.86980     10000
```