UNIVERSITY OF ILLINOIS URBANA-CHAMPAIGN

DEPARTMENT OF STATISTICS

Classification of Fashion Articles

Group Members:
Team Leader: Daniel Gu (wenyugu2)
Sharvi Tomar (stomar2)
Bruno Seo (sbseo2)

A project report submitted for the course of

STAT-542 Statistical Learning

Contents

Chapter 1: Introduction

1.1 Goal

The Fashion-MNIST dataset [XRV17] consists of $60,000$ training images and $10,000$ testing images where each image is a 28×28 grayscale image. In this project, our objective is to correctly identify Zalando's article images into one of the 10 categories.

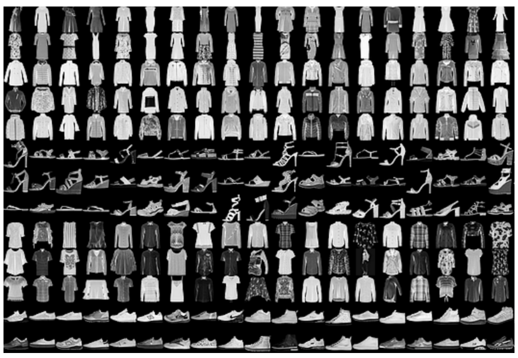

Figure 1.1: Fashion-MNIST

1.2 Approach

We have experimented with multiple models, including unsupervised learning techniques such as k-means clustering and DBSCAN and supervised learning methods such as SVM, random forests, logistic regression, and finally ensemble models. The first stage of the pipeline is some standard data preprocessing. We normalized all images into the range from 0 to 1 and then applied PCA to reduce the number of dimensions. The performance of each model is evaluated by accuracy on the testing set. Confusion matrix is also leveraged to establish more detailed insights into the model's capability of distinguishing each individual category. Cross validation is used to tune hyperparameters and our best model is able to achieve an accuracy of 91.39%. This model is a standard SVM with RBF kernel trained on HOG (Histogram of Oriented Gradients) features, a classic computer vision technique to extract features based on histograms of different gradient orientations. Since HOG computes each histogram in a local block of image, it is able to describe the structure or the shape of an object.

1.3 Conclusion

We explored unsupervised, supervised, and ensemble learning techniques and we found that a SVM based on HOG features(and reducing dimensions by applying PCA to explain 95% variation) gives the best performance (91.39% accuracy). In addition, although we expect that ensemble method should further improve model performance, the accuracy (91.27%) does not seem to improve. We think it might be due to the fact that in order for the ensemble model to further improve performance, all models in the ensemble must be diverse (i.e. they must make different mistakes and complement each other). However, in our case where we include SVM in the ensemble, it is already the best model in the ensemble and other models are making similar mistakes, e.g. they don't outperform SVM in either tricky cases such as classification between (T-shirt vs Shirt) or the easy cases. Therefore, adding extra models simply introduces more bias.

Chapter 2: Literature review

Image classification is a fundamental computer vision task. Handcrafted deep neural networks have achieved great results on image classification. However, designing such networks requires substantial human effort. Therefore, Neural Architecture Search (NAS) has been widely adopted. NAS methods have achieved astonishing results on the image classification task. Initially, NAS approaches used Reinforcement Learning and Evolutionary Algorithms. However, these approaches required substantial computational resources and time. Therefore, Differential Architecture Search (DARTS), a stochastic gradient descent approach of NAS was introduced. DARTS achieved competitive results at a significantly lower computational cost. However, DARTS makes numerous approximations to speedup leading to inferior performance. In [TKK21], the authors have utilized proven approaches such as the use of attention modules and fine-tuning to improve performance of the network. The attention mechanism allows neural networks to focus on a relevant region of the data. Attention module has 2 nodes(3 connections) and 3 operations. The attention module consists of the squeeze phase consisting of global average pooling operation, excitation phase consisting of two fully-connected layers, and then adaptively recalibrates channel-wise feature responses by explicitly modeling interdependencies between channels. A hyper-parameter r, reduction ratio, is maintained that allows to vary the capacity and computational cost of the attention module in the network. Contrary to DARTS, the authors have used a dual-stem approach at the inputs of the first cell. Rather than passing redundant information to both inputs of the first cell of the neural network, they propose applying 2 different transformations to the input image. Passing the first transformed image to the first cell as first input, and the second transformed image to the first cell as the second input. In this way, different information is passed to the first cell of the neural network. The model achieves state-of-the-art results on Fashion MNIST, COMP-CARS and MIO-TCD datasets while the results on other datasets were also competitive.

Shuning Shen [She18] suggested the use of Long Short-Term Memory Networks (LSTMs) in place of predominant CNNs for the task of image classification on the Fashion-MNIST dataset. LSTMs are a special type of Recurrent Neural Network with the advantage of being able to learn long-term dependencies better than RNN and providing a solution to the exploding and vanishing gradient problems. However, LSTM also suffers from long training time just like any RNNs and it requires sufficient memory bandwidth. The author conducted experiments on whether reducing the size of the training set could improve test performance. Shen reduced the original training set of 60,000 images to a random subset of 20,000 but concluded that it resulted in a performance reduction since the the model capacity would reduce and LSTM may not be able to fully learn the pattern pin the data. In addition, Shen tried to use network pruning to reduce the computational cost by removing connections between neuron if its weight is less than a specified threshold. Although the performance of a pruned model at threshold 0.05 slightly improved compared to the original model, the average time to update the weight is twice more than a normal LSTM. The proposed model with four hidden layers, with each layer comprising 64 hidden neurons has been able to achieve the an accuracy of 88.26%. The result is much worse than a traditional CNN since RNNs are not tailored to perform image classification tasks whereas CNNs are natural feature extractors due to their excellence in recognizing spatial patterns. Rather, RNNs are more suitable at tasks that require learning temporal patterns such as image captioning, speech recognition, etc.

Chapter 3: Data Processing

3.1 Dataset Source

The Fashion-MNIST data set is taken from the MNIST database (Modified National Institute of Standards and Technology). It is a large database of Zalando's article images that is intended to serve as a direct drop-in replacement for the original MNIST dataset for benchmarking machine learning algorithms.

3.2 Data Distribution

The class label distribution of both training and testing set is shown in Table 3.1. We can see that classes are uniformly distributed and thus we have a balanced dataset.

Training									
T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
6000	6000	6000	6000	6000	6000	6000	6000	6000	6000
Testing									
1000	1000	1000	1000	1000	1000	1000	1000	1000	1000

Table 3.1: Frequency Table

3.3 Dimensionality Reduction

3.3.1 Principle Component Analysis (PCA)

PCA is an unsupervised method and often used to perform dimensionality reduction that works by first centering the data, identifying the principal directions that explains the most variance and finally projecting the data to them. PCA is able to remove feature dependencies from multivariate data and also helps speed up computation for clustering tasks by using less features.

Figure 3.1 shows that 187 components is able to explain about 95% of the variance. We kept 95% of the variance before applying all classification models. It is also important to note that when applying PCA to testing data, we need to use the mean computed from training data since we do not know the mean for testing data beforehand.

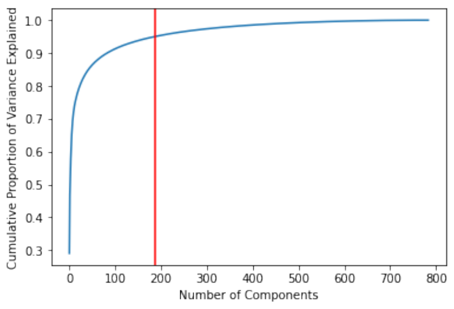

Figure 3.1: Cumulative Variation Explained

3.4 Clustering

3.4.1 k-Means Clustering

k-Means is a clustering technique that aims to partition n observations into k clusters in which each observation belongs to the cluster with the nearest mean (cluster centers or cluster centroid), serving as a prototype of the cluster. The algorithm requires to specify k beforehand. We chose $k = 10$. Figure 3.2 shows the projected data points onto 2D space using PCA and colored according to its assigned cluster.

Table 3.2 shows dominated class label for each cluster. Detailed class histograms for each cluster could be found in our code outputs.

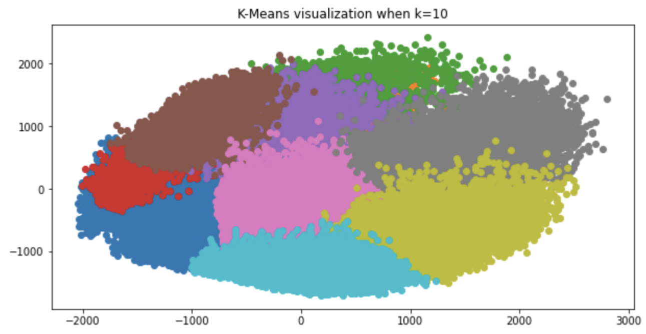

Figure 3.2: k-Means Cluster Visualization

Cluster Index	0	1	2	3	4	5	6	7	8	9
Label	Coat	Sandal	Trouser	Ankle boot	Shirt	Bag	Bag	Ankle boot	T-shirt	Sneaker

Table 3.2: K-means: Dominated Class Label in Each Cluster

3.4.2 Density-Based Spatial Clustering of Applications with Noise (DBSCAN)

DBSCAN is a data clustering algorithm that is based on the density instead of centroid such as k-means. It is of great advantage in situations where the data may contain noise because of the way clusters are formed. Unlike in K-means clustering algorithm where each observation becomes a part of some cluster even when it is scattered far in vector space and therefore, the clustering algorithm gets affected even for a slight change on data points, DBSCAN results are more reliable. While employing DBSCAN it is not required to specify the number of clusters beforehand and therefore, the algorithm enjoys great usability. It has two parameters: `Eps` (maximum radius of the neighborhood) and `MinPts` (minimum number of points in the ϵ-neighborhood of a point to be considered as a core point). The algorithm then merges points that can be reached by core points.

We observe that by using `Eps=8` and `MinPts=20`, DBSCAN is able to find 16 clusters and it gives more purity of label distribution in each cluster (i.e. most clusters only have one or two classes) but it has uneven distribution of samples between clusters: one large cluster that has dense points with several small clusters. This could be due to the fact that data cloud of some classes are probably very close to each other in space and DBSCAN then tries to treat them as one cluster since they are density-reachable to one another. Table 3.3 shows dominated class label for each cluster. Detailed class histograms for each cluster could be found in our code outputs.

Cluster Index	0	1	2	3	4	5	6	7
Label	Trouser	Coat	Ankle boot	T-shirt	Coat	Sneaker	Dress	Trouser
Cluster Index	8	9	10	11	12	13	14	15
Label	T-shirt	Shirt	Ankle boot	Pullover	Coat	Coat	Ankle boot	Dress

Table 3.3: DBSCAN: Dominated Class Label in Each Cluster

Chapter 4: Multi-class Classification

4.1 Histogram of Oriented Gradients(HOG)

HOG is a traditional computer vision technique to extract features based on different gradient orientations in a local region and it is able to describe objects of different shapes. Figure 4.1 illustrates the hog features of an image of shirt. The image on the right shows the dominant gradient direction in the histogram for every 4×4 cell. We can see that those directions give the general shape of the shirt. HOG takes in cell size as input and smaller cell would capture more detailed information since it is more local but in the meantime it would result in more feature variables. To balance the tradeoff, we chose a cell size of 4×4 which has 1296 features per sample. Then we further reduce the dimensionality by using PCA to keep 95% of the variance which ends up with only 354 features per sample image.

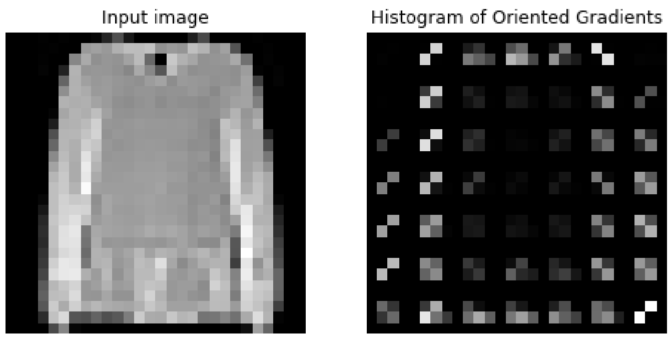

Figure 4.1: HOG features of an image of class Shirt

4.2 Support Vector Machine (SVM)

SVM is a supervised machine learning algorithm, that can be used for regression, classification and also to detect any outlier in the dataset. It is one of the most widely used classification model, where we plot each data point in an n-dimensional space, and create a hyper-plane to best separate the data points as per the labels.

4.2.1 SVM for Multi-class Classification Setting

SVM, by default, is a model for binary classification tasks. However, there are two main strategies to adapt it to support multiclass classification:

1. fitting a classifier for each class against the rest of the classes (one-vs-rest)

2. fitting $\frac{K(K-1)}{2}$ pairwise classifiers (one-vs-one) where K is the number of classes. The final prediction is made by either an argmax over class probabilities (if the model supports probability output) or majority voting (e.g. SVM, since it by default makes hard assignments to input samples).

We used Sklearn's implementation of multiclass SVM that uses the one-vs-one strategy.

4.2.2 Fine-tuning SVM

For finding the optimal hyperparameters, we resorted to Randomized-Search Cross-Validation with the following:

- **Training Data:** PCA applied on HOG features to keep 95% variance
- **Kernels:** 'poly', 'rbf', 'sigmoid'
- **C:** Uniform continuous values in the range (10, 100)
- **Best-parameters:** C: 64.88135039273247, `kernel`: 'rbf'

Table 4.1 shows accuracy and F-1 score of our fine-tuned SVM on the test data.

Classification Accuracy	91.39%
F-1 Score (with 'macro' average)	91.37

Table 4.1: Evaluation Metric on Test data

4.3 Random Forest

The random forest algorithm trains many decision trees on various sub-samples of the data (creating a forest) and uses them as classifier. It uses averaging to improve the predictive accuracy and control over-fitting.

4.3.1 Fine-Tuning Random Forest

For finding the optimal hyperparameters, we resorted to Randomized-Search Cross-Validation with the following:

- **Training Data:** PCA applied on HOG features to keep 95% variance
- **'max_depth':** [3, 5, 10]
- **'min_samples_split':** [2, 5, 10]
- **cv:** 5
- **Best-parameters:** `max_depth=10`, `min_samples_split=5`

Table 4.2 shows accuracy and F-1 score of our fine-tuned Random Forest model on the test data.

Classification Accuracy	82.92%
F-1 Score (with 'macro' average)	82.70

Table 4.2: Evaluation Metric on Test data

4.4 Logistic Regression

Logistic Regression is a popular binary classification algorithm. For employing it to a multi-class classification setting, there are the following two approaches:

1. One-vs-all: Transform to multiple binary classification problems
2. Multinomial Logistic Regression: Change of cost function from log-loss to cross-entropy loss and output result is not one probability value but rather a probability value for each class minus 1.

We trained Logistic Regression on the original data as well as the PCA-reduced data with 95% variation explained to obtain classification accuracy on test data 83.77% and 84.14% respectively.

4.5 XGBoost

A good model is the one with great generalization as well as prediction accuracy. Random Forest brings generalization by bootstraped aggregation while boosting brings greater prediction accuracy by sequentially building the trees on residuals of the prior trees. XGBoost is an efficient implementation of the gradient boosting algorithm designed for speedy computation in case of large and complex data.

We trained XGBoost on the original data as well as the PCA-reduced data with 95% variation explained to obtain classification accuracy on test data 85.82% and 82.88% respectively.

Chapter 5: Ensemble Model

Ensemble models can be built upon using multiple models and multiple features. While there are multiple ensemble techniques such as bagging, boosting, and stacking, we present two different ensemble models and explore the differences of these models: stacking classifier and voting classifier.

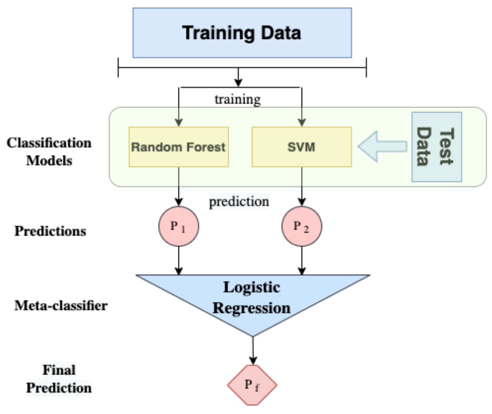

Figure 5.1: Stacking Classifier

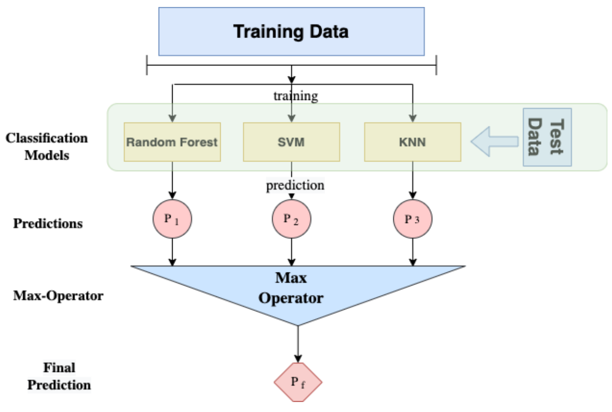

Figure 5.2: Voting Classifier

5.1 Stacking Classifier

Stacking is an ensemble learning technique to combine multiple classification models via a meta-classifier. The individual classification models are trained based on the complete training set; then, the meta-classifier is fitted based on the outputs, meta-features, of the individual classification models in the ensemble. The meta-classifier can either be trained on the predicted class labels or probabilities from the ensemble.

To be specific, as visualized in Figure 5.1, we fitted two different individual models in the first stage; Random forest and SVM. Then, we stacked the predicted probability of individual models. In the second stage, we created a new logistic regression classifier that uses the predicted probabilities from the previous stage. The implementation was done using the scikit-learn [PVG+11]. However, stacking classifier is known to overfitting the dataset due to information leakage. One potential solution is to split the training data into two folds and use the first fold to the train all weak learners in the ensemble. Then train the final meta classifier by using the predictions made from the trained weak learners on the second fold as inputs. Now we can ensure the information from training each model in the ensemble does not leak into the phase where we train the meta classifier.

5.2 Voting Classifier

A voting ensemble is an ensemble model that combines the predictions from multiple models to reach a final consensus. When used for classification, the predicted label from each model in the ensemble are aggregated into final prediction by using majority voting.

There are two approaches to the majority vote prediction for classification:

1. Hard Voting: Predict the class with the largest sum of votes from models
2. Soft Voting: Predict the class with the largest summed probability from models.

To be specific, as shown in Figure 5.2, we fitted three different individual models in the first stage: Random Forest, SVM, and KNN with model parameters (KNN: $k = 7$, L_1 distance; RF: `max_depth`: 50, `n_estimators`: 100; SVM: $C = 10$, kernel: poly). Then, we summed the predicted class probability of individual models. In the second stage, we simply outputted the class label that has the largest class probability according to the soft voting method. The implementation was done using the scikit-learn [PVG+11].

Since training SVM is the most computationally expensive, range from $O(n^2)$ to $O(n^3)$ depending on implementations, among all models in the ensemble, it is practically infeasible to perform fine-tuning using cross-validation with the available computational resource based on the observation that fine-tuning a SVM even when using the randomized CV strategy would require more than 5 hours. Fine-tuning an ensemble when a SVM is involved would yield too time consuming. Therefore, we directly used the best parameter setting for each model as we obtained before.

The voting classifier, as presented in Figure 5.2, is able to achieve a classification accuracy of 91.21% on the test data when trained on HOG features after applying PCA.

5.3 Model Results

The table below summarizes the accuracies obtained on test data using various models. SVM model trained on the data obtained by applying PCA on HOG features to explain 95% variation performs the best. This model gives the classification accuracy of $\approx$ 91.4% on the test data. Therefore, we report 8.6% as the misclassification rate of the proposed model.

Model	**Original data**	**PCA-reduced data**	**PCA on HOG features**
KNN (baseline)	86.13%	86.48%	83.32%
SVM	88.08%	**89.75%**	**91.39%**
Random Forest	87.79%	85.87%	85.59%
Logistic Regression	83.77%	84.14%	88.01%
XGBoost	85.82%	82.88%	88.12%
Stacking(Random Forest+SVM) Logistic Regression	89.33%	89.62%	91.27%
Voting(Random Forest, SVM, KNN)	**89.59%**	89.63%	91.21%

Table 5.1: Accuracies on Test Data for different Train Data settings

5.4 Result Analysis

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
Precision	0.83	0.59	0.79	0.92	0.81	1.00	0.75	0.88	0.99	0.95
Recall	0.84	0.99	0.77	0.78	0.73	0.87	0.51	0.97	0.91	0.96
F1 score	0.84	0.74	0.78	0.85	0.77	0.93	0.60	0.92	0.95	0.95

Table 5.2: Classification report when using KNN

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
Precision	0.78	1.00	0.75	0.87	0.76	0.94	0.71	0.92	0.92	0.93
Recall	0.87	0.96	0.79	0.89	0.80	0.92	0.53	0.93	0.96	0.94
F1 score	0.82	0.98	0.77	0.88	0.78	0.93	0.61	0.93	0.94	0.94

Table 5.3: Classification report when using Random Forest

	T-shirt	Trouser	Pullover	Dress	Coat	Sandal	Shirt	Sneaker	Bag	Ankle boot
Precision	0.88	0.99	0.88	0.91	0.87	0.98	0.73	0.95	0.99	0.98
Recall	0.86	0.97	0.85	0.92	0.88	0.97	0.76	0.97	0.98	0.97
F1 Score	**0.87**	0.98	0.87	0.92	0.88	0.97	**0.74**	0.96	0.98	0.97

Table 5.4: Classification report when using SVM

While we expected the ensemble models to outperform every single model, our empirical results show that a single SVM trained on the data obtained by applying PCA on HOG features outperforms the all of the ensemble models. This result is explainable in that the single models' (Random forest, logistic regression, and KNN) performance is worse than SVM even before being combined as an ensemble model. That is, it is highly plausible that they have played as noise rather than a meaningful signal.

To be specific, Table 5.2, Table 5.3, and Table 5.4 show why voting ensemble model fails to capture the meaningful signal. For every label, we can observe that a single SVM outperforms both KNN and Random Forest in F1-score. For example, in the most tricky classification case (T-shirt vs Shirt), we can observe that SVM dominates in F-1 score compared to other models. This is also true for other cases as we can see from the table. Considering voting classifier chooses the most voted class among input models, voting classifier is susceptible to noises. The same explanation could apply to the stacking classifier where we expect the ensemble model to outperform each base classifier when each model is diverse enough (i.e. they must make different mistakes). However, with the presence of one superior model, the performance of the entire ensemble could not be further improved. The final meta-classifier would be heavily biased towards the most superior model.

Since the SVM model trained on the data obtained by applying PCA on HOG features achieved the highest accuracy, we report it as our proposed model. Here, we further analyze the results of the proposed model.

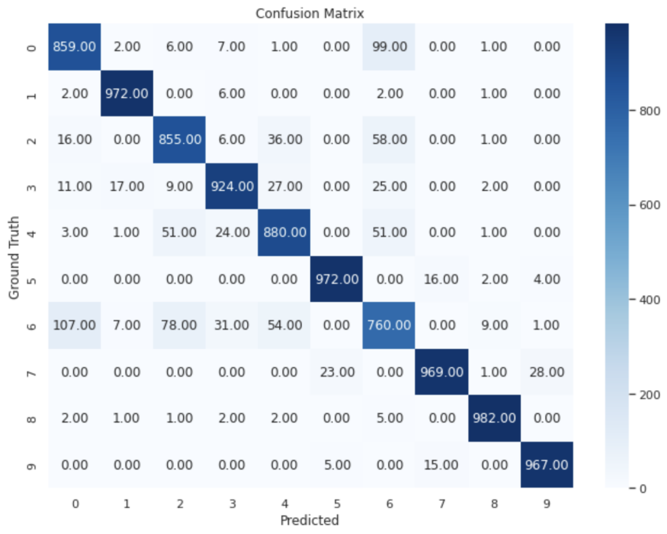

Figure 5.3: Confusion matrix of the SVM model where features are reduced using PCA on HOG features.

The confusion matrix and the classification report are visualized in Figure 5.3 and Table 5.3 respectively. They show that the proposed model predicts (8) Bag (F1 score: 98 %) and (1) Trouser (F1 score: 98 %) the most successfully while (6) Shirt (F1 score: 74 %) the least successfully. This result is understandable by the confusion matrix in that the most misclassified label of Shirt is T-Shirt. The average of overall F1 score is 91 %.

Bibliography

[PVG^{+}11] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[She18] S Shen. Image Classification of Fashion-MNIST Dataset Using Long Short-Term Memory Networks. *International Journal on Recent and Innovation Trends in Computing and Communication*, 6(4), 2018.

[TKK21] M. Tanveer, M. Karim Khan, and C. Kyung. Fine-Tuning DARTS for Image Classification. *International Conference on Pattern Recognition (ICPR)*, 6(4):4789–4796, 2021.

[XRV17] Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-mnist: A Novel Image Dataset for Benchmarking Machine Learning Algorithms, 2017.