

Classification of Standard FASHION MNIST Dataset Using KNN and DT

Abstract—Due to its general simplicity, the MNIST handwritten digit recognition dataset is a poor choice for testing contemporary supervised classification ML techniques. It was originally designed to test basic ML algorithms. The MNIST fashion dataset, which comprises a comparable quantity of images, pixels, and classes, has lately taken the position of this dataset. This article uses Python code to classify image instances taken from the dataset and implements two classifiers: a kNN classifier and a decision tree classifier. In this article, we try different parameters for each machine learning algorithm to get the best performance, and for each classifier, we compare and evaluate the results. Next, the performances of the best kNN classifier and the best decision tree classifier is compared. Different metrics are used to evaluate the performance of the training model.

Keywords—*MNIST fashion dataset; K-Nearest Neighbor(KNN); Decision Tree(DT); Image Classification*

I. INTRODUCTION

The MNIST dataset, which consists of 10-class handwritten digits, was first presented by LeCun et al. in 1998 [1]. It was impossible to foresee the amazing development of deep learning algorithms and their performance at the time. Google Trends shows that the uncomplicated MNIST dataset has outperformed CIFAR-10 [2] and ImageNet [3] in terms of popularity despite the fact that deep learning can now do a variety of jobs. Its usage does not seem to be decreasing, despite requests for it in the deep learning sector and despite its simplicity.

Because of its vastness, MNIST has gained a lot of attention from deep learning researchers, who may use it to swiftly test and prototype their algorithms. All machine learning and deep learning frameworks, including Tensorflow and Scikit-Learn, provide useful utility functions and real-world applications that use MNIST straight out of the box.

Fashion-MNIST is built on the collection found on the Zalando website. Every fashion item on Zalando is accompanied by a series of expertly snapped images showcasing the item's numerous aspects, including front and back views, details, looks with a model, and looks while wearing an outfit. The original image is a 762 x 1000 JPEG file with the background color #fdfdfd in hexadecimal format. To efficiently serve multiple frontend components, the original image is resampled using a variety of resolutions, including large, medium, small, thumbnail, and tiny.

Fashion-MNIST is a dataset of fashion product photos designed to replace MNIST and offer a more difficult option for testing machine learning algorithms. Fashion-Photos MNIST, as shown in Fig. 1, is converted into a format that is the same as the MNIST dataset, making it instantly interoperable with any machine-learning program capable of using the original MNIST dataset.

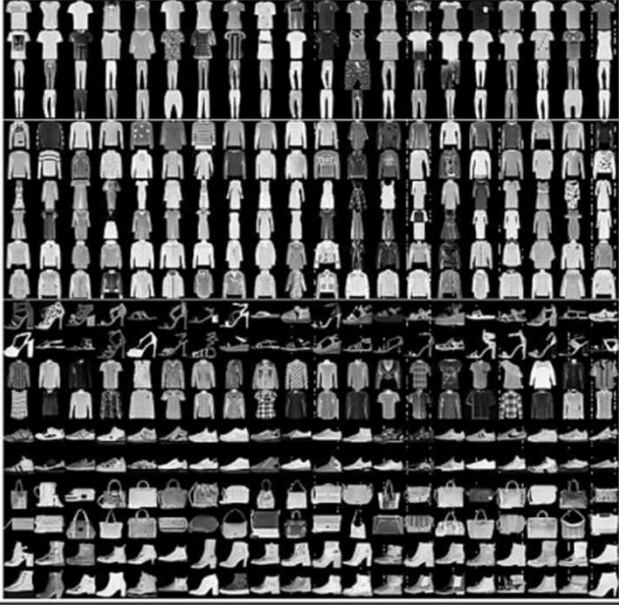
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig.1. Fashion-MNIST Dataset

II. FASHION-MNIST CLASSIFICATION BY KNN

One of the simplest machine learning methods for classification and regression is KNN (K-Nearest Neighbor). It is an algorithm for supervised learning. According to Fig. 2, if the majority of the K comparable samples in the feature space—that is, the samples that are closest to it—belong to a particular category, the sample itself must as well. In other words, the approach chooses to divide the sample's category [4] solely in accordance with the category of the adjacent sample or samples.

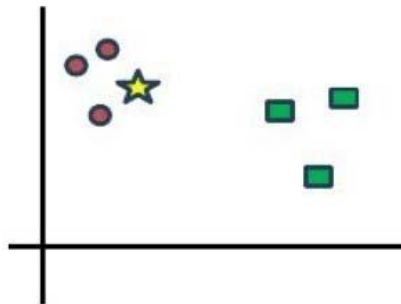


Fig.2. Working of KNN

A. The KNN is used to classify the following categories:

A data-statistics method serves as the foundation for KNN, as do many other machine learning algorithms. At the same time, KNN is instance-based learning, which is a type of lazy learning. As a result, it lacks a clear early training phase and instead begins categorization as soon as the data set is placed into memory [5]. You will vote for K, the closest point to the sample point, each time an unknown sample point is appraised. This is what K in KNN means, and K is typically an integer with a maximum value of 20.

The KNN classification algorithm's computation procedure is typically broken down into five

steps:

- 1) Determine how far the point that has to be categorised is from the point that belongs to the existing category.
- 2) Arrange in ascending order of distance.
- 3) Choose the K points that are closest to the categorized point out of all the K possible locations.
- 4) Count the instances of the category represented by the first K points.
- 5) The anticipated classification of the point to be classed was returned as the category with the highest frequency of the first K points.

B. The KNN is used to return to the following data:

By averaging the values of the closest K points, the value of the point to be predicted is determined. The Euclidean distance or another distance can be considered the "closest." The specific effect is similarly dependent on the data. In Fig. 3, a feature is represented by the x-axis, its value by the y-axis, and the known location is represented by the red point. Calculate the average of the closest three red points (yellow box), the first green point, and the green line to determine the location of the first point.

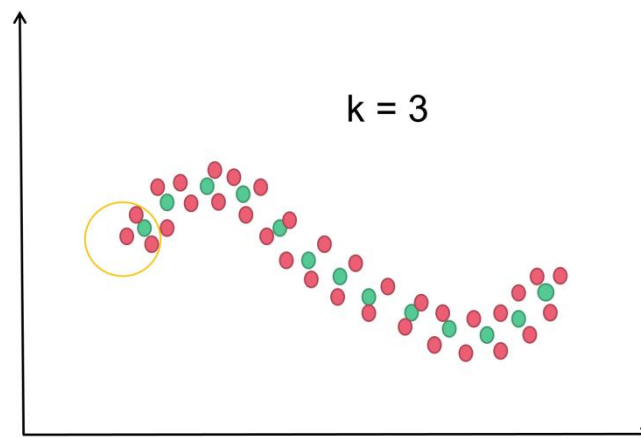


Fig.3. Selecting Nearest Data Points in KNN

Whether it is used for classification or regression, the KNN algorithm has numerous salient features:

- (1) The hyperparameter K of the method.
 - (2) Measuring distance The amount of similarity between the sample points is reflected in the distance between them in the feature space.
 - (3) Classified rules for making decisions, where the minority is required to follow the majority.
- Among them, there are a few issues that require consideration, including the selection of the K value and the distance measurement setting.

The final algorithm's prediction outcomes will be significantly influenced by the K value selection since it is the sole bit hyperparameter used in the KNN algorithm. The approximation error of "learning" will reduce if a smaller value of K is chosen, which is equal to forecasting using training examples in a smaller neighborhood. The only training examples that had a closer input instance would have an impact on the prediction outcomes. The drawback is that as "learning" progresses, estimation error will grow

and prediction outcomes will become more sensitive to instance sites in close proximity. The prediction is incorrect if the nearby instance point turns out to be noise [6]. In other words, a declining K value indicates a very complex and overfitting-prone overall model. If a high value of K is used, prediction using training examples in broad neighborhoods is equal. The benefit is that learning will result in less estimation error, while the drawback is that learning will result in more approximation error. At this point, the training example removed from the input instance will also contribute to the prediction, resulting in a simple overall model that is vulnerable to underfitting as the error K value of the prediction increases. Regardless of the input instance, it is simply anticipated to belong to the largest class of training examples under the extreme situation $K = N$, it can be assumed. Currently, the model is too basic and does not take into account a significant amount of important training data, which makes it undesirable [7].

The cross-validation method is typically used in applications to choose the ideal K value. According to the analysis above, the general K value is quite low. The K value will be chosen from a limited set, and the one with the highest accuracy on the validation set will be chosen as the algorithm hyperparameter K.

In sample space, the distance between any two points reflects how similar they are to one another: a smaller distance denotes a higher similarity, whereas a larger distance denotes a lesser similarity. The Manhattan distance, the Euclidean distance, and the Minkowski distance are the three distances that are frequently employed in the KNN method.

The advantages and disadvantages of the KNN algorithm are as follows:
merits:

- 1) The theory is sound and the procedure is straightforward; it can be applied to both classification and regression.
- 2) It is applicable to nonlinear classification.
- 3) There is no apparent training procedure, but once the data set is placed into memory and the software begins to operate, there is no need to train or make a direct prediction, hence the training time complexity is 0.
- 4) The KNN technique is more appropriate than other approaches for the overlapping sample sets of class domains since it primarily relies on the nearby, constrained neighboring samples rather than the method to select the category.
- 5) Classes with large sample sizes can be automatically classified using this approach, however classes with small sample sizes are more likely to be misclassified.

shortcomings:

- 1) You should figure out how far apart each test point is from the practice area. Massive training sets result in large calculation amounts and significant time complexity, particularly when there are many features.
- 2) Having a high spatial complexity and requiring a lot of memory.
- 3) There is a sample imbalance issue (some categories have a lot of samples, while other categories have very few samples), and unusual categories have poor prediction accuracy.
- 4) The lazy learning approach, which does not learn, yields predictions more slowly than algorithms like

logistic regression.

In this dataset, using KNN as a classifier where N takes different values, the accuracy is shown in Fig. 4. It can be found that the value of N is steadily increasing as the value of N gets larger and larger.

```

N= 3000 mean accuracy= 0.83355
N= 3500 mean accuracy= 0.8366285714285714
N= 4000 mean accuracy= 0.8391
N= 4500 mean accuracy= 0.84252
N= 5000 mean accuracy= 0.84405
N= 5500 mean accuracy= 0.8440666666666666
N= 6000 mean accuracy= 0.8473
Wall time: 3min 6s

```

Fig.4. Accuracy for Different N

The above results are shown as line charts, as shown in Fig. 5:

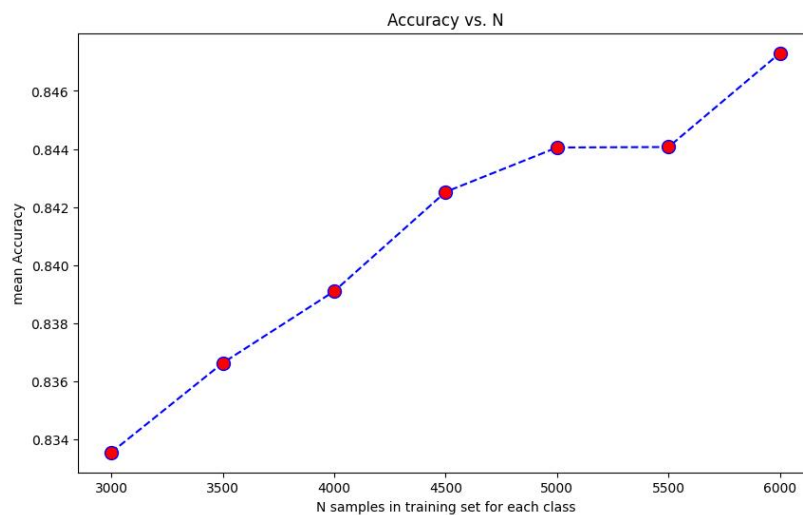


Fig.5. Accuracy vs.N

Next, look specifically at N = 6000, as shown in Fig. 6. An average accuracy of 0.8473 is obtained.

	precision	recall	f1-score	support
0	0.76	0.86	0.81	1000
1	0.99	0.95	0.97	1000
2	0.75	0.79	0.77	1000
3	0.91	0.88	0.89	1000
4	0.79	0.80	0.79	1000
5	0.99	0.78	0.87	1000
6	0.64	0.57	0.60	1000
7	0.85	0.94	0.89	1000
8	0.97	0.94	0.95	1000
9	0.87	0.96	0.91	1000
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

N=6000, mean accuracy= 0.8473
Wall time: 29.1 s

Fig.6. N = 600

Use Kfold to select the training set, as shown in Fig. 7. You can see the first one with the highest average accuracy.

```

TRAIN: [10000 10001 10002 ... 69997 69998 69999] TEST: [ 0 1 2 ... 9997 9998 9999]
mean accuracy= 0.8513
TRAIN: [ 0 1 2 ... 69997 69998 69999] TEST: [10000 10001 10002 ... 19997 19998 19999]
mean accuracy= 0.8457
TRAIN: [ 0 1 2 ... 69997 69998 69999] TEST: [20000 20001 20002 ... 29997 29998 29999]
mean accuracy= 0.8473
TRAIN: [ 0 1 2 ... 69997 69998 69999] TEST: [30000 30001 30002 ... 39997 39998 39999]
mean accuracy= 0.8477
TRAIN: [ 0 1 2 ... 69997 69998 69999] TEST: [40000 40001 40002 ... 49997 49998 49999]
mean accuracy= 0.8429
TRAIN: [ 0 1 2 ... 69997 69998 69999] TEST: [50000 50001 50002 ... 59997 59998 59999]
mean accuracy= 0.8472
TRAIN: [ 0 1 2 ... 59997 59998 59999] TEST: [60000 60001 60002 ... 69997 69998 69999]
mean accuracy= 0.8474
Wall time: 1min 52s

```

Fig.7. Use Kfold to Select the Training Set

Next, select the index of the first fold as a training set, as shown in Fig. 8.

	precision	recall	f1-score	support
0	0.77	0.89	0.83	1009
1	0.99	0.96	0.97	972
2	0.73	0.78	0.76	951
3	0.91	0.86	0.88	1012
4	0.77	0.80	0.79	979
5	1.00	0.80	0.89	1008
6	0.65	0.56	0.60	1022
7	0.86	0.96	0.91	1037
8	0.97	0.94	0.96	974
9	0.89	0.96	0.93	1036
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

Fig.8. Select the Index of the First Fold as Training Set

III. FASHION-MNIST CLASSIFICATION BY DECISION TREE

A technique for estimating the values of discrete functions is the decision tree algorithm. It is a typical classification method in which the data are processed first, comprehensible rules and decision trees are produced using an induction algorithm, and the trees are then used to analyze the fresh data. A decision tree is essentially a method of classifying data using a set of rules.

The 1960s saw the initial development of the decision tree method, which was applied up until the late 1970s. J. Ross Quinlan's ID3 algorithm seeks to minimize the depth of the tree. However, they disregarded the investigation of leaf counts. The C4.5 algorithm, which is enhanced based on the ID3 algorithm and appropriate for both classification issues and regression problems, significantly enhances the default treatment of predictor variables, pruning technology, and derivative rules [8].

When the data features are continuously distributed, CART and C4.5 enable the treatment primarily by treating continuous variables via binary segmentation, that is, by locating a certain split value: the eigenvalue is greater than the split value, or the left subtree, or the right subtree. The goal of this split value, which has distinct definitions for the C4.5 and CART algorithms, is to lessen the "degree of

confusion" within the divided subtree.

The "Occam's razor" principle, which states that a smaller decision tree is preferable than a larger one (be simple theory), is the foundation of Ross Quinlan's ID3 algorithm [9]. The feature with the highest information gain is consistently chosen in the ID3 algorithm in accordance with information gain theory. Without using the pruning procedure, nominal data sets can be divided using the ID3 method. Crop and merge neighboring leaf nodes that are unable to yield significant information improvements to solve the issue of excessive data matching (such as setting the information gain threshold). The problem of using information gain is that it frequently exhibits valuable features. In other words, the more distinct the values in the training set, the more probable it is that they will be interpreted as split attributes, which are occasionally meaningless. The C4.5 technique is employed since another ID3 cannot manage the continuous distribution of the data attributes. The continuous distribution of the data features is likewise supported by the CART method.

C4.5 is an enhanced version of ID3 that gains from its advantages. C4.5 The method chooses attributes based on the information gain rate, overcoming the drawback of choosing attributes with numerous values for pruning during tree construction. It can also process incomplete data and finish the discrete processing of continuous attributes. The C4.5 method generates classification rules that are accurate and simple to grasp, but they perform poorly when building trees. Additionally, C4.5 is only appropriate for datasets that can be stored in memory because datasets must be scanned several times.

The CART method incorporates the post-pruning process and employs the Gini index as the splitting criterion (choose the features with the minimum Gini index). Although the ID3 and C4.5 algorithms can extract as much information as they can from the training sample set, the decision trees they build contain a huge number of branches and a complex structure. The decision tree algorithm CART, which chooses the test qualities in accordance with the GINI coefficient, appears in order to reduce the size of the decision tree and increase the effectiveness of producing decision trees.

To find the categorization rules hidden in the data, the decision tree algorithm builds a decision tree. The fundamental idea behind the decision tree algorithm is how to build a decision tree on a small size with high precision. It is possible to build a decision tree in two steps. Creating decision trees from the training sample set is the initial phase, which is called decision tree generation. The training sample data set is typically historical and integrated into data processing and analysis in accordance with actual requirements. The accuracy of the balance branch out will be impacted by the second step, the decision tree pruning. Decision tree pruning is generated in the previous stage of the decision tree inspection, correction, and repair process, primarily with a new sample data set (called test data set) during the process of preliminary rules.

A. Learning from decision trees

The objective is to create a decision tree model that can accurately categorize the instances using a specified training dataset. A set of classification rules are effectively inferred from the training dataset through decision tree learning. A decision tree with several nodes or none at all can accurately classify the training data. The conditional probability model should have a strong match not only with the training data but also with the unknown data, and while selecting a decision tree, we should select one

with fewer contradiction to the training data and good generalization capacity. Typically, the loss function is a maximum likelihood function that has been normalized. The main tactic is to make the loss function as the objective function as little as possible. Decision tree learning typically uses heuristics to address this optimization problem because selecting the best decision tree from all potential decision trees is an NP-complete problem, and the resulting decision tree is suboptimal (sub-optimal). The decision tree learning algorithm often involves the recursive selection of the best feature and the division of the training data into subsets based on this feature, producing the best classification algorithm for each subdataset. Feature selection, decision tree creation, and decision trees are all parts of the pruning process.

1) Pruning

The objective is to simplify the tree and improve its capacity for generalization. The next step is to change the parent or higher nodes to the new leaf node after removing two subdivided leaf nodes and returning them to the parent or even higher nodes [10].

2) Generation of the decision tree

The pruning of the decision tree reflects the global selection of the model, which is equivalent to the local selection of the model. While decision tree pruning takes into account global optima, decision tree generation just takes into account local optima.

3) Feature choice

If there are many features, they are initially eliminated during decision tree learning, leaving only the features with enough classification strength to handle the training data. (For instance, by omitting a name from the list of features.)

B. Typical algorithm

ID3, C4.5, CART, and other common algorithms are used with decision trees. The top 10 traditional algorithms in the field of data mining were chosen by ICDM (the IEEE International Conference on Data Mining), the world's premier academic organization and conference on data mining, in December 2006. The C4.5 method came in first. The ID3 algorithm, which forms the basis of the C4.5 algorithm, is a type of classification decision tree algorithm used in machine learning algorithms. The C4.5 algorithm generates classification rules that are accurate and simple to comprehend. The data set must be repeatedly scanned and sorted during the tree construction process, which makes the technique ineffective in real-world settings.

Data mining, risk analysis, operations research, and decision analysis are the fields where DT is most frequently applied. DT is also a well-liked tool in machine learning. DT appears as a structure like an upside-down tree. Each node selects a unique characteristic. Readers who are interested can view the specifics in [11].

After processing the MNIST dataset, the decision tree model was created with the `DecisionTreeClassifier` function of the `sklearn` library for training and prediction. There are many parameters in the `DecisionTreeClassifier` function. To get the optimal parameter setting, we need to set different decision tree parameters and find the best ones by training the resulting feedback after the prediction. After several experimental attempts, the best parameters of the decision tree were selected as follows: `splitter= "best"`, `min _samples_ split=2`, `min _samples _leaf=1`, and the rest of the parameters

can default. The decision tree classifier is constructed using the best parameters and trained on the training set. After training the model, the model is used to predict the test set and compare it with the real label values. The prediction results of the decision tree classifier had an accuracy of 0.7732.

The decision tree can read the data set and extract the rules included in some columns of data. It applies to numerical and nominal kinds (discrete data, the results of variables are only valued in the limited goal set). Using decision tree models to solve categorization issues has various benefits. The decision tree calculation is simple, straightforward, and effective. While rules are typically simple to perceive and comprehend, decision trees can handle data with unrelated properties and quickly create rules that are intelligible. Additionally, decision tree models have certain drawbacks, including issues with handling missing data, overfitting, and neglecting correlations between dataset features.

IV. CONCLUSION

From the final results, the KNN classifier works better than the best decision tree classifier. However, their effect is not the best of all the algorithms, using the convolutional neural network Algorithms can easily improve the accuracy to more than 90% [12] [13].

REFERENCE

- [1] Y.LeCun, L.Bottou, Y.Bengio, and P.Haffner.Gradient-based learning applied to document recognition.Proceedings of the IEEE , 86(11):2278–2324, 1998.
- [2] A.Krizhevsky and G.Hinton.Learning multiple layers of features from tiny images.2009.
- [3] J.Deng, W.Dong, R.Socher, L.-J.Li, K.Li, and L.Fei-Fei.Imagenet: A large-scale hierarchical im age database.In Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on , pages 248–255. IEEE, 2009.
- [4] C.Sheppard, Tree-based Machine Learning Algorithms: Decision Trees, Random Forests, and Boosting.CreateSpace Independent Publishing Platform, 2017
- [5] A.D.Bhat, H.R.Acharya and S.H.R., "A Novel Solution to the Curse of Dimensionality in Using KNNs for Image Classification," 2019 2nd International Conference on Intelligent Autonomous Systems (ICoIAS), 2019, pp.32-36, doi: 10.1109/ICoIAS.2019.00012.
- [6] S.Cheamanunkul and Y.Freund, "Improved kNN Rule for Small Training Sets," 2014 13th International Conference on Machine Learning and Applications, 2014, pp.201-206, doi: 10.1109/ICMLA.2014.37.
- [7] T.Makkar, Y.Kumar, A.K.Dubey, Á.Rocha and A.Goyal, "Analogizing time complexity of KNN and CNN in recognizing handwritten digits," 2017 Fourth International Conference on Image Information Processing (ICIIP), 2017, pp.1-6, doi: 10.1109/ICIIP.2017.8313707.
- [8] A.Comeau and C.McDonald, "Analyzing Decision Trees to Understand MNIST Misclassification," 2019 IEEE MIT Undergraduate Research Technology Conference (URTC), 2019, pp.1-4, doi: 10.1109/URTC49097.2019.9660504.
- [9] K.Lavanya, S.Bajaj, P.Tank and S.Jain, "Handwritten digit recognition using hoeffding tree, decision tree and random forests — A comparative approach," 2017 International Conference on Computational Intelligence in Data Science (ICCIDS), 2017, pp.1-6, doi: 10.1109/ICCIDS.2017.8272641.
- [10] M.Bari, A.Ambaw and M.Doroslovacki, "Comparison of Machine Learning Algorithms for Raw Handwritten Digits Recognition," 2018 52nd Asilomar Conference on Signals, Systems, and Computers, 2018, pp.1512-1516, doi: 10.1109/ACSSC.2018.8645138.
- [11] Liu, Xu, Y., Duan, G., Qiu, C., & Tan, J.(2021).Accurate on-line support vector regression incorporated with compensated prior knowledge.Neural Computing & Applications, 33(15), 9005–9023. <https://doi.org/10.1007/s00521-020-05664-2>
- [12] M.Kayed, A.Anter and H.Mohamed, "Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture," 2020 International Conference on Innovative Trends in Communication and Computer Engineering (ITCE), 2020, pp.238-243, doi: 10.1109/ITCE48509.2020.9047776.
- [13] E.Xhaferri, E.Cina and L.Toti, "Classification of Standard FASHION MNIST Dataset Using Deep Learning Based CNN Algorithms," 2022 International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT), 2022, pp.494-498, doi: 10.1109/ISMSIT56059.2022.9932737.