

Comparative Analysis of Machine Learning and Deep Learning Models for Classification of the Fashion MNIST Dataset

Kirthan Prakash
College of Arts and Sciences
Texas A&M University
College Station, Texas
kirthan_prakash@tamu.edu

Saptarshi Alak Mondal
College of Arts and Sciences
Texas A&M University
College Station, Texas
saptarshi.mondal@tamu.edu

Kiran Babu Athina
College of Arts and Sciences
Texas A&M University
College Station, Texas
kiran3566@tamu.edu

Abstract—Fashion MNIST, with its set of 28x28 grayscale images of various fashion products, presents a more diverse, realistic and complex set of patterns and shapes to analyze, making it a better benchmark for machine learning algorithms as compared to the standard MNIST dataset. This paper offers an extensive comparison of several classification models benchmarked on the Fashion MNIST dataset. These models include Support Vector Machine Classifier(SVM), XGBoost, K-Nearest Neighbors (KNN), Logistic Regression, Decision Tree Classifier, Random Forest Classifier, Artificial Neural Networks (ANN), and Convolutional Neural Networks (CNN). In order to determine these models efficacy in image classification tasks, our work focuses on evaluating them using several metrics, including accuracy, precision, recall and F1-score.

Index Terms—Image Classification, Fashion MNIST, Artificial Neural Networks, Convolutional Neural Networks, Machine Learning Algorithms

Project codes are available publicly at GitHub project link

I. INTRODUCTION

The fashion industry has evolved as a result of tremendous changes in the fashion market over the past 30 years. The best approaches to boost profit are to better target sales and comprehend consumer preferences. The growth of online commerce has made it possible for consumers to purchase clothing more quickly and easily through websites. This has given rise to a huge amount of data and the need to efficiently process it to extract meaningful insights, which otherwise renders the raw data no longer needed and relevant.

One such basic problem in the garment industry is classifying clothing images into their respective garment categories. The paper focuses on solving this classification problem using various Machine learning and Deep learning methods. Later, comparing the efficiency and quality of each model in solving our Fashion MNIST classification problem.

It is necessary to first establish what exactly a classification problem is. In simple words, labeling a particular entity to a specific category such that it reflects some degree of similarity with other entities placed in that same category. In this case, based on the properties of several images of garments and their corresponding categories, we are trying to classify any new image of clothing article into its best-suited category.

II. CLASSIFICATION PROCESS

A. Introduction to Dataset

The Fashion MNIST dataset was created by Han Xiao, a researcher at Zalando Research, and was introduced in September 2017. Zalando is a European e-commerce company that focuses on fashion and lifestyle products, and the motivation behind creating Fashion MNIST was to have a dataset that reflects real-world challenges in the fashion domain. The Fashion MNIST is a large freely available dataset of images that is generally used to train and test machine learning models. The Fashion-MNIST dataset comprises 70,000 28x28 grayscale pictures of fashion items divided into ten groups. There are 10,000 samples in the test set and 60,000 samples in the training set. Each image is represented as a 28x28 array, and the pixel values range from 0 to 255.

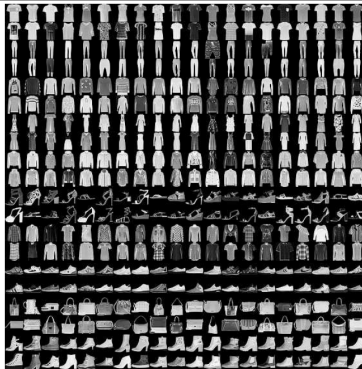
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Fig. 1. Fashion MNIST Dataset

The Fashion MNIST dataset provides a more challenging classification task than the simple MNIST dataset due to the complexity of distinguishing between different types of clothing. The original MNIST dataset is a set of handwritten digits that includes 70,000 28x28 grayscale images with 10 labels. Fashion-MNIST follows the same structure, but instead of using numbers, it uses images of fashion products.

The OpenML platform consists of a large collection of datasets for machine learning. It allows users to access datasets, share datasets, and even run experiments on datasets directly from the OpenML platform. The Fashion MNIST dataset is available on OpenML, and we have used the OpenML API to fetch and work with it.

B. Data Cleaning and Scaling

Data Cleaning: It is critical to train the machine learning model on correct and reliable data. Without adequate cleaning, the model may learn from faulty data, resulting in inaccurate predictions or classifications. For this project, we noticed that FashionMNIST was completely populated with no missing values.

Data Scaling: It is a transformation technique that changes the range of the feature values, making it easier for the training algorithm to converge faster. We scaled each pixel of the images by dividing 255, making each feature value between [0,1].

$$\chi = \alpha/255 \quad (1)$$

where χ is scaled pixel value and α is the old value.



Fig. 2. PCA projection of Fashion MNIST onto 2D

C. Exploratory Data Analysis

The Fashion MNIST dataset consists of 28x28 grayscale images of 70,000 fashion commodities from 10 distinct classes. Upon examination, we notice that the dataset is well-prepared with each product at the center of the image. This eliminates the need for additional image preprocessing, such as alignment or rotation correction, and makes it straightforward to use machine learning algorithms on the dataset. Next, we examined the class distribution to make sure there was no large imbalance that could skew the model's training. With 7000 photos per class, we observed that the dataset is completely balanced.

Pixel intensity analysis validates the expected outcome for a grayscale image dataset, that the majority of the pixels are black, meaning the greatest frequency is 0 values. This

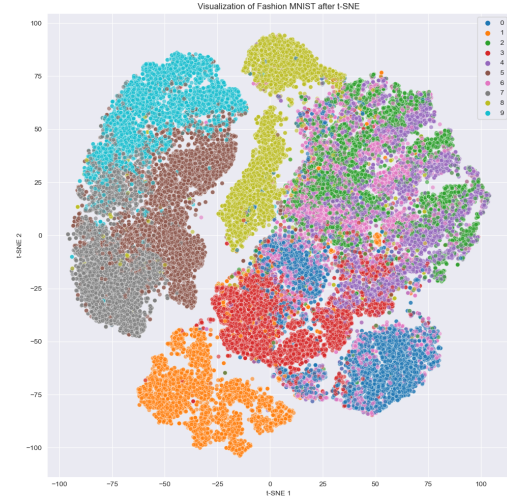


Fig. 3. t-SNE projection of Fashion MNIST onto 2D

insight raises the possibility of using dimensionality reduction techniques to reduce the number of features while preserving a substantial amount of information. We used Principal Component Analysis (PCA) and t-distributed stochastic neighbor embedding (t-SNE) to reduce the dimensions to 2D and plot the same.

By converting a dataset into a set of orthogonal (uncorrelated) variables known as principal components, that are arranged so that the starting few retain the majority of the variation in the original dataset, PCA is an unsupervised statistical technique that lowers a dataset's dimensionality. [3] By first generating a probability distribution in the high-dimensional space that shows similarities between data points and then mapping this to a lower-dimensional space in a way that maintains these similarities, t-SNE conducts dimensionality reduction. Fig. 2 and Fig. 3 reveal that while some classes form distinct clusters, others overlap, suggesting varying degrees of difficulty that a classification model might encounter. A thorough grasp of the Fashion MNIST dataset is created through these EDA processes, laying the groundwork for more intelligent and successful model creation and evaluation.

III. METHOD - MODEL CONSTRUCTION

"A classification is a set of objects that have similar properties. And these properties are selected to determine how similar or different the objects are." [2]

A. Classifier Summary

We have considered various classifiers for this projects. Starting with Machine Learning models like Support Vector Machines, Logistic Regression, K-Nearest Neighbors, Decision Tree Classifier, Random Forest Classifier to Deep Learning architectures in ANN and CNN and compared the performances of each.

1) *K-Nearest Neighbors (KNN)*: k-Nearest Neighbors (KNN) groups new data points according to the classification of their neighbours. The k nearest neighbours to the new data point are found using a distance measure, which is how the method works. Then, a majority vote among its neighbours establishes the new point's class. The hyper-parameters that we tuned for this model are:

- `n_neighbors`: Specifies the number of nearest neighbors to be used in the voting process.
- `weights`: Determines the weight to be given to each neighbor.
- `metric`: The distance metric that is used to determine how close or similar two data points are is defined by this parameter.

2) *Logistic Regression*: The sigmoid function is used in logistic regression to simulate the probability that an input falls into a particular class. Generally, maximum likelihood estimation is used to estimate the weights of the logistic regression model. This entails determining the set of parameters that make the observed outcomes most probable. The logistic function is defined as:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \quad (2)$$

The hyper-parameters that we tuned for this model are:

- `C`: Represents the inverse of regularization strength, tuned to avoid overfitting.
- `max_iter`: Specifies how many iterations are taken in total for the solvers to converge.

3) *Support vector Machines (SVM)*: Finding a partition hyperplane in the sample space based on the train set and classifying various sample classes is the fundamental idea behind SVM classification. [5] The maximum margin hyperplane is constructed such that it is equidistant from both the support vectors and the sum of the distances is maximum. The SVM model generalises better as the margin of the model increases. SVM converts non-linear data into a higher-dimensional space where a hyperplane can be utilised for classification by using the kernel trick. SVM performs especially effectively in high-dimensional spaces and is ideally suited for problems with distinct class boundaries. The hyperparameters that we tuned for this model are:

- `C`: Regularization parameter. As C increases, the amount of regularization decreases.
- `kernel`: Type of kernel function used for the kernel trick.
- `gamma`: It is a parameter for non-linear kernels. It indicates the extent to which a single training example has an impact.
- `coef0`: Provides a degree of control over the model. Depending on the kernel, it adjusts the influence of higher-degree terms in the polynomial kernel or the sigmoid kernel.
- `degree`: Signifies the degree of the polynomial used in the 'poly' kernel.

4) *XGBoost*: Within the family of gradient boosting techniques comes the potent and well-liked machine learning algorithm known as Extreme Gradient Boosting. "XGboost is a typical Boosting method that integrates many CART regression trees." [6] The core ideas of XGBoost is add a regularization to the target function. [6] In classification, XGBoost builds an ensemble of decision trees in a stepwise manner, with each new tree trying to correct the flaws of the previous ones. The algorithm continuously improves the predictions by fitting a new model to the residual errors of the prior models at each iteration. One of its key features is the use of a regularized model formalization to control overfitting, making it robust and accurate. The hyperparameters that we tuned for this model are:

- `max_depth`: Indicates a tree's greatest depth. Although deeper trees can represent more intricate patterns in the data, they also have a chance to overfit data.
- `learning_rate`: Learning rate used in boosting. The contribution of each tree is reduced by this factor as a function of learning rate.
- `gamma`: Describes the minimal loss reduction needed to create a second division on a tree leaf node.
- `subsample`: The fraction of the data to be used for fitting each particular tree is specified by this parameter.

5) *Decision Tree Classifier*: In order to create a structure like a tree, Decision Tree divides the information into subsets on the basis of the most important attribute at each level. This procedure keeps on until a predetermined depth or the quantity of samples in a leaf node—or another stopping criterion—is satisfied. It uses criterion like entropy or gini index, to decide the best split. The hyper-parameters that we tuned for this model are:

- `Criterion`: Quantifies the quality of a split.
- `MinSamplesLeaf`: The bare minimum of samples needed at a leaf node.
- `MinSamplesSplit`: The least amount of samples needed to divide an internal node.

6) *Random Forest Classifier*: During training, a large number of decision trees are built using the Random Forest ensemble learning technique, which produces a label that represents the mode of the labels in the classification. We have tuned the same hyperparameters as used in our decision tree classifier.

7) *Artificial Neural Network*: An input layer, hidden layers for processing, and an output layer are the three layers that make up an Artificial Neural Network (ANN). The incoming data is transformed mathematically by the network's neurons, and during training, the weights—a term for the connections between neurons—are modified to provide accurate predictions.

Our ANN processes 28x28 grid input (image) with a flattening layer followed by Batch Normalization to normalize the input data, promoting stable and accelerated training. It utilizes two fully connected layers (128 and 64 neurons) with ReLU

activation, refining features. The output layer (10 neurons) employs Softmax for class probability distribution, aligning with the 10 classes.

8) *Convolution Neural Network*: Convolutional Neural Networks (CNNs) are multi-layered models adept at extracting intricate structures from input data. Trained through back-propagation, CNNs excel in discerning high-quality details and complex patterns, particularly in image analysis. It is particularly useful in extracting features from large scale images while having significantly lower number of parameters as compared to using an ANN. Our CNN architecture is shown below:

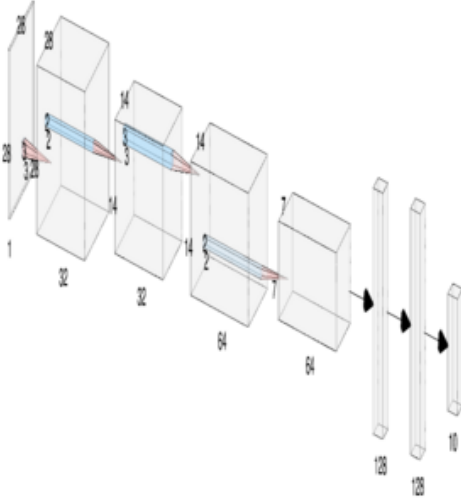


Fig. 4. Convolution Neural Network

The CNN architecture begins with a 28x28 grayscale image input. It employs a Conv2d layer with 32 filters, followed by ReLU activation and max-pooling. Subsequently, a similar process is applied with a Conv2d layer featuring 64 filters. After that, the hierarchical features are flattened and processed using ReLU activation in two Dense layers, each with 128 neurons. The final Dense layer, with softmax activation, produces class probabilities for the 10 output classes. This architecture systematically extracts and transforms features, enabling effective image classification.

IV. EXPERIMENTATION AND MODEL COMPARISON

A. KNN, Logistic Regression, Decision Tree and Random Forest using Grid Search CV:

One renowned technique for machine learning hyperparameter tuning is GridSearch. It entails building a grid of values for the hyperparameters and training the model for each possible combination of values.

All the four model were chosen after performing Grid Search over a parameter grid along with 3 fold cross validation. Then the best trained models were evaluated on test dataset.

	Test Accuracy	Recall Score	Precision Score	F1-Score
Decision Tree Classifier	0.7807	0.7807	0.7808	0.7805
Logistic Regression	0.8422	0.8422	0.8405	0.8410
Random Forest Classifier	0.8583	0.8583	0.8566	0.8565
XGBoost Classifier	0.8674	0.8674	0.8662	0.8665
K Nearest Neighbors	0.8692	0.8692	0.8714	0.8696
ANN (Artificial Neural Network)	0.8742	0.8742	0.8747	0.8735
Support Vector Classifier	0.8913	0.8913	0.8913	0.8912
CNN (Convolution Neural Network)	0.9128	0.9128	0.9124	0.9124

Fig. 5. Model Performance Comparision

B. SVM and XGBoost using Random Search CV:

Similar to Grid Search, Random Search helps find the optimal set of hyperparameter values for a model. GridSearch can be computationally expensive, especially with large datasets, a wide range of parameter values, or models that are slow to train. RandomSearch chooses random combinations to test from a predefined parameter space, as opposed to attempting every possible combination of parameters. RandomSearch is particularly useful when dealing with a large hyperparameter space or when the computational resources are limited.

Model	Hyperparameters Used
Logistic Regression	C: 0.1, max iter: 100
K Nearest Neighbors	metric: 'manhattan', n neighbors: 4, weights: 'distance'
Support Vector Classifier	C: 792.34; coef0: 4.46; degree: 3; gamma: 2.40; kernel: 'poly'
XGBoost Classifier	subsample: 0.7; max depth: 5; learning rate: 0.2; gamma: 0.5
Decision Tree Classifier	criterion: 'entropy'; min samples leaf: 2; min samples split: 5
Random Forest Classifier	criterion: 'gini'; min samples leaf: 1; min samples split: 5

Fig. 6. Model Hyperparameters Used

C. Artificial Neural Network (ANN) and Convolution Neural Network (CNN):

Both the Deep learning models ANN and CNN architecture (mentioned in Section III) underwent training on the complete dataset, utilizing batches comprising 64 instances. Employing a k-folds validation strategy with 3 folds, the models underwent training for 50 epochs during each fold iteration. At each epoch, the Validation Loss was computed, and the model checkpoint with the minimum Validation Loss was saved. If there was no improvement in the Validation Loss for 10 consecutive epochs, an early stopping mechanism was activated, then the last best model (checkpoint model) was utilized for testing. Next the trained models were evaluated on test dataset. Test accuracy, confusion matrix and other evaluation metrics were obtained.

V. CONCLUSION

We notice that even a simple CNN with only 2 convolution layers outperforms our best SVC model. While we can increase the performance by using a more complex and deep architecture, but this comes at a higher computational cost. On the other hand, the traditional machine learning models offer simplicity and interpretability, providing insights into the practical applicability of these models in real-world fashion item classification tasks.

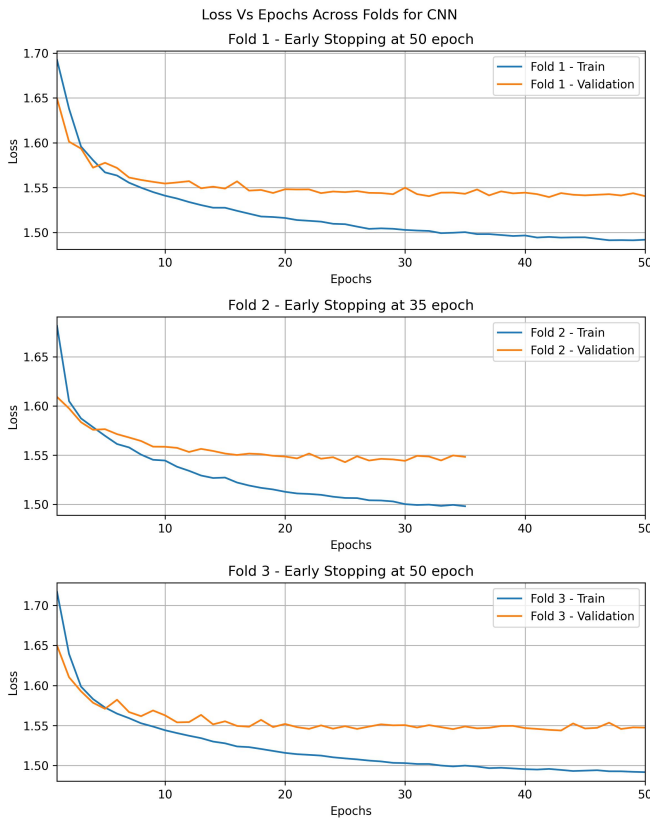


Fig. 7. Convolution Neural Network Loss plot

Using Support Vector Classification (SVC) over neural networks for classifying the Fashion MNIST dataset presents several advantages. Firstly, with an accuracy of 89.13%, SVC performs significantly better than ANN (87.59%) and is only slightly behind CNN (91.3%). This demonstrates its competitive effectiveness in handling image classification tasks. Additionally, SVC models are generally easier to implement and require less computational resources compared to neural networks, particularly CNNs, which are known for their computational intensity. This makes SVC a more accessible and cost-effective option for many applications. Furthermore, SVC requires less hyperparameter tuning, making it a more robust choice for scenarios where a simpler model is sufficient or preferred. Its mathematical foundation in margin maximization offers a clear, interpretable mechanism for classification, which can be advantageous in applications where understanding the decision-making process is important. Overall, while it may slightly lag behind in accuracy compared to CNNs, SVC's balance of efficiency, simplicity, and robustness makes it a strong contender for the Fashion MNIST dataset classification.

Using machine learning to automate several procedures inside a fashion industry can result in significant cost reductions. Different ML models can vary in their ability to accurately classify fashion products from images. Based on our analysis, the company can choose the type of model based on their use case. When using models internally in

the company for inventory categorization and management, to automatically categorize new clothing items into appropriate categories, high accuracy would be required. While deploying a model externally to enhance customer interaction on their website, may call for an efficient ML model. Models that require less computational power for training and inference are more economical, especially when operating at scale.

LITERATURE REVIEW

SVM and XGBoost Classifiers are powerful machine learning algorithms which perform well in image classification tasks. According to experiments from [6], XGBoost functions best when machine learning and image categorization technologies are combined. "Support vector machine (SVM) is a kind of generalized linear classifier which can be used to classify data or images." [5]. We find that these models perform with a high accuracy on the Fashion MNIST dataset as well.

As deep learning techniques have advanced, CNN image recognition has become overly used in fashion-related fields such as autonomous clothing labelling, garments retrieval, and categorization. From [1], "we find that the details of the training algorithm are dependent on a variety of factors; data features, optimizers, convolutional layers, step size, and network parameters." From [4], "LeNet-5 gives a higher performance (an accuracy over 98% was obtained) as compared to other existing models." If our primary goal is accuracy, then we can implement the LeNet-5 architecture.

REFERENCES

- [1] 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), Ankara, Turkey, 2022, pp. 494-498, doi: 10.1109/ISMSIT56059.2022.9932737.
- [2] J. D. Savchuk and A. Doroshenko, "Investigation of machine learning classification methods effectiveness," 2021 IEEE 16th International Conference on Computer Sciences and Information Technologies (CSIT), LVIV, Ukraine, 2021, pp. 33-37, doi: 10.1109/CSIT52700.2021.9648582.
- [3] Y. Shuai, Y. Zheng and H. Huang, "Hybrid Software Obsolescence Evaluation Model Based on PCA-SVM-GridSearchCV," 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 2018, pp. 449-453, doi: 10.1109/ICSESS.2018.8663753.
- [4] 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), Aswan, Egypt, 2020, pp. 238-243, doi: 10.1109/ITCE48509.2020.9047776.
- [5] M. Yang, S. Cui, Y. Zhang, J. Zhang and X. Li, "Data and Image Classification of Haematococcus pluvialis Based on SVM Algorithm," 2021 China Automation Congress (CAC), Beijing, China, 2021, pp. 522-525, doi: 10.1109/CAC53003.2021.9727433.
- [6] H. Chen, R. Du, Z. Liu and H. Xu, "Android Malware Classification using XGBoost based on Images Patterns," 2018 IEEE 4th Information Technology and Mechatronics Engineering Conference (ITOEC), Chongqing, China, 2018, pp. 1358-1362, doi: 10.1109/ITOEC.2018.8740537.