A Comparative Analysis of Machine Learning Models for Image Classification of Fashion Items Using the Fashion-MNIST Dataset

Pragati Naikare
Data Science
Texas A&M University
College Station, USA
pragatinaikare311@tamu.edu

Tejashri Kelhe
Data Science
Texas A&M University
College Station, USA
tkelhe@tamu.edu

Garima Badhan
Data Science
Texas A&M University
College Station, USA
gbadhan@tamu.edu

***Abstract*—In the realm of online fashion, advanced algorithms are essential for understanding clothing, aiding companies in tailoring sales strategies. This project centers on Fashion MNIST data classification, employing machine learning and deep learning techniques to enhance clothing recognition, search, and recommendation systems for online shopping. The complexity of categorizing clothing arises from intricate garment properties, demanding a robust solution. Convolutional Neural Networks (CNNs) emerge as a powerful tool to address this complexity. The study employs four machine learning classification models, utilizing the Fashion-MNIST dataset tailored for clothing classification. A comprehensive comparison is conducted to determine the most effective classification method. The project contributes to the comparison and selection of the best algorithm for image classification within the Fashion_MNIST database. The final CNN model, structured based on deep learning, achieves an impressive test accuracy of 91.03%.**

***Index Terms*—component, formatting, style, styling, insert**

I. Introduction

Image classification, a vital aspect of computer vision, entails assigning a class label to an image. It finds applications in object detection, scene recognition, and medical image analysis. The growing interest in machine learning for image classification involves training algorithms on labeled image datasets. The Fashion-MNIST dataset serves as a popular benchmark due to its accessibility and challenging nature for testing state-of-the-art algorithms. In the context of improved living standards and prevalent online shopping trends, efficient clothes classification is crucial for both clothing companies and consumers. This process enhances the online shopping experience by enabling quick and accurate product searches. Image classification, a fundamental challenge in computer vision, has practical applications in image and video indexing. Overcoming challenges like illumination variations, scale changes, and occlusions is critical for accurate predictions. This report explores machine-learning algorithms[6] for clothing image recognition. Initial attempts with KNN and random forest indicated room for improvement. Subsequently, Convolutional Neural Networks (CNN)[1][2][3] demonstrated superior results. CNNs, with three hidden layers featuring ReLU nonlinearity and a softmax final layer, exhibited clear advantages for clothing image classification. Through multiple algorithms and thorough analysis, this paper concludes that CNN offers distinct benefits over alternative methods in this domain.

II. Method:

There has been a significant amount of research on image classification using machine learning. Some of the most popular machine learning algorithms for image classification include KNN[6][8], CNNs and Random Forest[6][9]. KNN is a non-parametric algorithm that classifies new data points based on the majority class of their nearest neighbors in the training data. CNNs are a type of neural network that are well-suited for image classification tasks. CNNs are able to learn features from images that are relevant for classification. Random Forest is an ensemble learning algorithm that combines multiple decision trees to improve classification accuracy.
Our project is dedicated to identifying the most suitable model for the fashion MNIST dataset. Throughout our experiments, we assess the performance of various models, including K-Nearest Neighbour (KNN), Convolutional Neural Network (CNN), and Random Forest. Notably, KNN and Random Forest are supervised machine learning algorithms, while CNN represents a deep learning approach. The accuracy metrics for these models reveal that CNN consistently outperforms the others. To achieve the desired outcomes, we sequentially executed the following steps: loading data, performing data cleaning, conducting exploratory data analysis, visualizing data, splitting data, transforming data, reducing dimensionality[4][5], training the model, selecting the model, tuning hyperparameters, evaluating the model, and interpreting the model.

III. Experiments

Our goal is to accurately classify images into one of the 10 unique classes. To achieve this, we systematically conducted model training, selection, and evaluation on the datasets. Before initiating the complex process of model selection, we rigorously implemented a series of critical steps (denoted from A to F) to ensure the integrity and appropriateness of our datasets. These steps lay the groundwork for the subsequent stages of our experimentation, playing a crucial role in enhancing the reliability and performance of the models developed and evaluated for our image classification task.

A. Load Data

We import the training and testing datasets for the Fashion MNIST dataset using the 'keras.datasets' module with the following code: 'from keras.datasets import fashion_mnist'. Subsequently, we examine the dataset's dimensions. The training dataset comprises 60,000 sample images, each containing 785 features. These features correspond to the pixel values of 28x28-pixel-sized images. Similarly, the test dataset consists of 10,000 images, each with 785 features. In summary, the Fashion MNIST dataset encompasses a total of 70,000 grayscale images, each sized at 28 pixels by 28 pixels.

B. Data Cleaning

Thorough data cleaning procedures were diligently applied to both datasets to mitigate potential issues. Our examination focused on two key aspects:
a. Identifying out-of-range pixels in the dataset.
b. Detecting the presence of any missing values within the dataset.
During the analysis, we specifically investigated whether the pixel values exceeded the range of 0 to 255. Our scrutiny revealed that all pixel values fell within this expected range, with none surpassing the upper limit of 255. Following this, we conducted a comprehensive check for missing values and confirmed the absence of any such instances in the dataset. This meticulous data cleaning process ensures the integrity and reliability of the datasets for subsequent analyses and model training.

C. Exploratory Data Analysis

Exploratory Data Analysis (EDA) was systematically conducted to glean insightful perspectives on the distribution, statistical properties, and visual patterns inherent in the dataset. This encompassed the following methodological steps:
a. Analyzing the class distribution of the data using both the training and testing set.
b. Examining key statistical measures of the data.
c. Employing data visualization techniques to enhance our understanding.
To analyze the class distribution, we generated plots Fig. 1 and Fig. 2 illustrating the relationship between 'Class Label' and 'Count.' The resulting figures vividly depicted that the classes were uniformly distributed across the datasets. This observation is pivotal, as it ensures a balanced representation of different classes, laying a solid foundation for subsequent analyses and model training.

We also delved into the statistical properties of the data, examining various measures that include count, mean, standard deviation, minimum value, maximum value, and quartiles

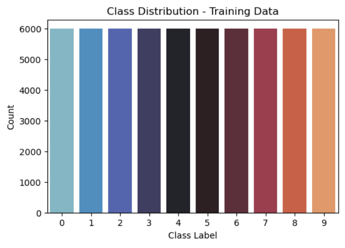


Fig. 1. Class Distribution - Training Data

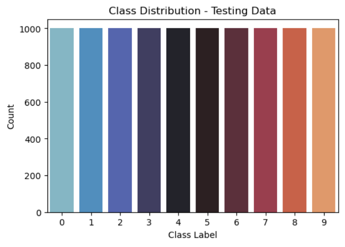


Fig. 2. Class Distribution - Testing Data

(25%, 50%, and 75%). This comprehensive analysis provided us with a nuanced understanding of the dataset's central tendencies, variability, and distribution characteristics. By exploring these statistical metrics, we gained deeper insights into the overall structure and variability of the data, facilitating more informed decision-making in subsequent stages of analysis and modeling. We utilized data visualization techniques to portray the distinctive class labels present in the dataset. The Fig. 3 illustrates the presence of 10 unique class labels, providing a clear and visual representation of the diversity within the data. This visualization not only aids in grasping the categorical composition of the dataset but also serves as a valuable reference for subsequent stages of analysis and model interpretation. By visually inspecting the class labels, we gain a more intuitive understanding of the dataset's categorical composition and diversity. Furthermore, as can be seen in Fig. 4 we employed t-distributed Stochastic Neighbor Embedding (t-SNE) as a powerful tool for visualizing our dataset. The t-SNE, operating as an unsupervised non-linear dimensionality reduction technique, proves valuable for exploring and visually representing high-dimensional data. Its non-linear nature allows for the differentiation of data points that would be indistinguishable using a linear approach. By leveraging t-SNE, we gain a more nuanced and intuitive visualization of the dataset's intricate relationships and structures, enabling a deeper understanding of the underlying patterns and clusters within the data. This approach enhances our ability to discern complex relationships that might not be readily apparent in the original high-dimensional space.

D. Data Splitting

We executed a structured data partitioning process, resulting in distinct datasets named train_x, train_y, val_x, val_y, test_x, and test_y. The division was strategically carried out to create

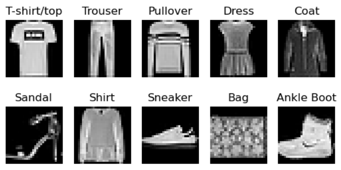


Fig. 3. All class labels

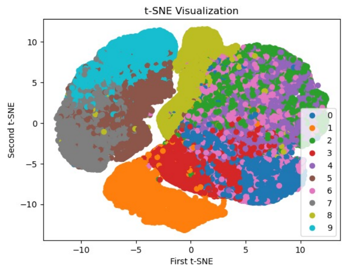


Fig. 4. t-SNE visualization

separate sets for training, validation, and testing. Specifically, we allocated 10% of the original train set as the validation set, leaving the remaining 90% as the revised training set. Within the train_x dataset, we excluded the feature column containing class labels, ensuring that it exclusively comprised feature data. Conversely, the train_y dataset exclusively retained the column containing class labels. This same principle was applied to create test_x and test_y datasets. This meticulous partitioning not only establishes distinct subsets for training, validation, and testing but also ensures that each dataset maintains the appropriate balance between feature data and corresponding class labels. Such a well-organized split is crucial for training and evaluating machine learning models effectively.

E. Data Transformation

We employed data normalization to standardize the data consistently across all datasets—training, validation, and test sets. The train_x, val_x, and test_x datasets were normalized by dividing each element by 255.0. This procedure transforms pixel values, originally ranging from 0 to 255, to a standardized scale between 0 and 1. Normalizing data in this way is common in machine learning to improve model convergence and performance. Bringing all values within the same scale facilitates a more consistent and effective learning process, assisting the model in better understanding underlying patterns in the data. This normalization is particularly valuable when features have different ranges or units, preventing certain features from dominating the learning process due to their larger magnitudes. Extending this normalization across all datasets establishes a uniform foundation for subsequent modeling steps, promoting better convergence and enhancing the model's ability to generalize to new, unseen data.

F. Dimensionality Reduction

We employed Principal Component Analysis (PCA) to conduct dimensionality reduction on our dataset [10]. The PCA model was fitted to the training set, and this transformation was consistently applied to all datasets, including the validation and test sets. To retain 90% of the information, we executed PCA on the train_x dataset. The subsequent transformation resulted in new datasets with reduced dimensions, namely (54000, 84) for the training set, (6000, 84) for the validation set, and (10000, 84) for the test set. By reducing the dimensionality while preserving a significant portion of the dataset's variance, we aim to streamline the computational complexity of the model and enhance its efficiency in capturing essential patterns within the data. This step contributes to creating a more computationally efficient and effective representation for subsequent modeling stages.

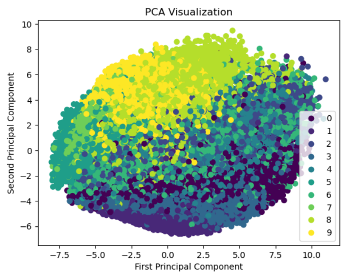


Fig. 5. PCA visualization

G. Model Training and Selection

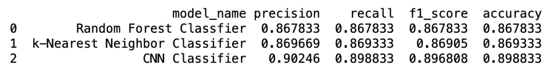
The previous preparatory steps laid a robust foundation for subsequent model selection by ensuring the cleanliness, integrity, and consistency of the datasets. In determining the optimal model for our specific case, we employed the following modelling approach:
a. The diverse models—KNN, CNN, and Random Forest—underwent training using the designated training set train_x. We used the default parameters for each model.
b. Subsequently, on the validation set, we assessed the performance of each model, carefully evaluating their accuracy, precision, recall, and other relevant metrics. This step allowed us to fine-tune hyperparameters and make informed decisions about model selection.
c. After performing model selection, taking the selected model, a further analysis was conducted on the test set to validate the models' generalization capabilities and overall effectiveness.
d. Overall, this comprehensive approach ensured a thorough evaluation of each model's performance across different datasets, contributing to the robustness of our findings.

Below are the specifics of model training and the corresponding results on the validation set for each of the three models: K-Nearest Neighbors (KNN), Random Forest, and Convolutional Neural Network (CNN).

1) KNN: The K-Nearest Neighbors (KNN) algorithm was employed for image classification. The features were extracted and the training involved comparing the similarity between the test image and labeled training images. The KNN algorithm was trained with its default parameters using the model provided by the scikit-learn library. For this application, the inputs to the model were derived from the refined training datasets obtained post Principal Component Analysis (PCA), namely train_pca and train_Y. In particular, the train_pca dataset encompasses all features excluding the class labels, while the train_Y dataset exclusively contains the class labels. Upon applying the KNN model to the validation dataset, the achieved accuracy was measured at 86.93%.

2) Random Forest: The random forest algorithm was applied with its default parameters using the model available in the scikit-learn library. The input datasets for this model were derived from the refined training data post-Principal Component Analysis (PCA), specifically named train_pca for features and train_Y for labels. The random forest, being an ensemble learning method, constructs multiple decision trees during training. Upon evaluation, the accuracy of this random forest model on the validation dataset was found to be 86.78%.

3) CNN: A Convolutional Neural Network (CNN), a deep learning approach for image classification, was employed. The network automatically learned hierarchical features from the image data during training. To train the CNN model, we adjusted the dimensions of the train, test, and validation sets to meet the required size. The implementation of CNN was facilitated through the features of the Keras library, serving as an interface for the TensorFlow library. In the CNN setup, default parameters were utilized, with a maximum of 10 epochs and a batch size of 128 specified. For training the model, the reshaped train sets, the designated batch size, a maximum of 10 epochs, and a verbosity level set to 1 were employed as inputs to the CNN model. The accuracy of this model on the validation dataset was recorded at 89.88%. These accuracy metrics provide insights into the model's learning performance over multiple epochs, showcasing its capability to progressively enhance its classification accuracy on the validation data.



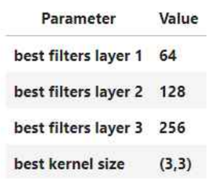
```
                     model_name precision    recall  f1_score  accuracy
0        Random Forest Classfier  0.867833  0.867833  0.867833  0.867833
1  k-Nearest Neighbor Classifier  0.869669  0.869333   0.86905  0.869333
2                 CNN Classifier   0.90246  0.898833  0.896808  0.898833
```

Fig. 6. Model Comparison

From Fig. 6., upon a thorough evaluation of each model's performance on the validation set, scrutinizing key metrics such as accuracy, precision, recall, and F1-score, the CNN model emerged as the preferred choice for further advancement. Notably, the CNN model exhibited superior accuracy and precision, surpassing the performance of both K-Nearest Neighbors (KNN) and Random Forest in the initial epoch. This decision to proceed with the CNN model is rooted in its promising early-stage performance, setting a strong foundation for subsequent refinement and optimization as the training progresses. The meticulous consideration of various evaluation metrics ensures a comprehensive and informed choice in selecting the most effective model for the image classification task at hand.

H. Hyperparameter Tuning

In the course of model selection, where we opted to advance with the CNN model, we engaged in the crucial process of hyperparameter tuning. Hyperparameter tuning for a Convolutional Neural Network (CNN) involves systematically adjusting the configuration settings that are not learned during the training process, such as learning rate, batch size, and the number of layers or filters. Hyperparameter tuning is an iterative optimization task, aimed at enhancing the model's performance on the validation set. It involves experimenting with different combinations of hyperparameter values to identify the set that yields the best results. Following the meticulous tuning process, the hyperparameters that contributed to the optimal performance of the CNN model were determined and can be seen in Fig. 7. These refined parameters serve as the foundation for a more finely tuned and efficient model.

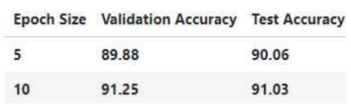


Parameter	Value
best filters layer 1	64
best filters layer 2	128
best filters layer 3	256
best kernel size	(3,3)

Fig. 7. Tuned hyperparameters for CNN

I. Model Evaluation

Following the refinement of hyperparameters, we subjected the CNN model to a comprehensive evaluation using the tuned configuration. A meticulous comparison of validation accuracy and test accuracy was conducted across various epochs to gauge the model's performance over time. This comparative analysis allows us to assess how well the CNN model generalizes to both the validation and test datasets, providing valuable insights into its robustness and predictive capabilities. The examination of accuracy trends across different epochs enables a nuanced understanding of the model's learning dynamics and its ability to consistently perform well on unseen data.



Epoch Size	Validation Accuracy	Test Accuracy
5	89.88	90.06
10	91.25	91.03

Fig. 8. Model Evaluation

J. Model Interpretability

To illuminate the decision-making process inherent in the CNN model, an essential interpretability technique known as Gradient-weighted Class Activation Mapping (Grad-CAM)[11] was adopted. Grad-CAM serves as a pivotal tool,

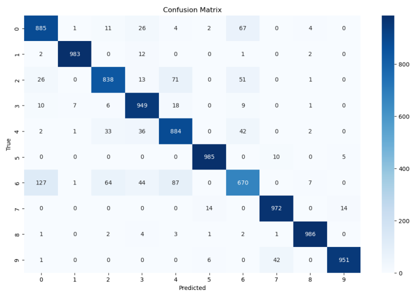


Fig. 9. Confusion Matrix for CNN

shedding light on the inner workings of the CNN by visually pinpointing regions within images that exert significant influence on the model's predictions. This elucidation proves instrumental in discerning the pivotal portions of input images that contribute most substantially to the CNN's classification decisions. By virtue of Grad-CAM's application, the interpretability of the CNN model is significantly enhanced, allowing for a deeper comprehension of the features and patterns instrumental in the model's decision-making process. This methodological approach not only fortifies the interpretability of the CNN but also affords researchers a nuanced understanding of the underlying mechanisms dictating the model's classifications, thus enriching the broader scope of CNN analysis and model transparency. These visualizations in Fig. 10 are instrumental in explaining the CNN's behavior, providing valuable insights into its inner workings.

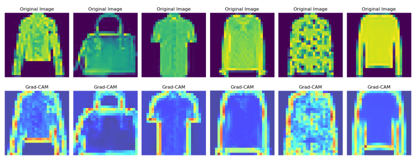
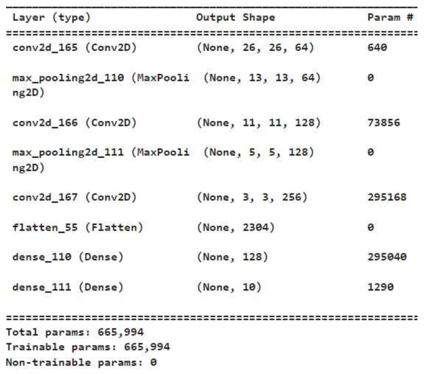


Fig. 10. Model Interpretability



```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_165 (Conv2D)         (None, 26, 26, 64)        640

 max_pooling2d_110 (MaxPooli  (None, 13, 13, 64)       0
 ng2D)

 conv2d_166 (Conv2D)         (None, 11, 11, 128)       73856

 max_pooling2d_111 (MaxPooli  (None, 5, 5, 128)        0
 ng2D)

 conv2d_167 (Conv2D)         (None, 3, 3, 256)         295168

 flatten_55 (Flatten)        (None, 2304)              0

 dense_110 (Dense)           (None, 128)               295040

 dense_111 (Dense)           (None, 10)                1290

=================================================================
Total params: 665,994
Trainable params: 665,994
Non-trainable params: 0
```

Fig. 11. CNN Model Summary

IV. Conclusion:

The project underscores the importance of careful model selection, hyperparameter tuning, and iterative evaluation in the development of effective image classification systems. The comparison of different models, including KNN, Random Forest, and CNN, revealed that the CNN model outperformed the others in terms of accuracy on the validation set. The iterative evaluation of metrics such as precision, recall, and F1-score provided a comprehensive understanding of each model's strengths and weaknesses. The decision to proceed with the CNN model was based on its superior accuracy, especially in the early epochs. The success of the CNN model and the insights gained from this project provide a foundation for continued exploration and refinement in the realm of image classification. Future work can build upon these findings to further optimize model performance and explore advanced techniques for even more robust and accurate predictions.

References

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," Advances in neural information processing systems, vol. 25, pp. 1097-1105, 2012.

[2] E. Xhaferra, 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.

[3] 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.

[4] H. AlSaeed, N. Hewahi and R. Ksantini, "Dimension Reduction Techniques for Image Classification," 2022 International Conference on Innovation and Intelligence for Informatics, Computing, and Technologies (3ICT), Sakheer, Bahrain, 2022, pp. 358-365, doi: 10.1109/3ICT56508.2022.9990707.

[5] W. Khan, M. Turab, W. Ahmad, S. H. Ahmad, K. Kumar and B. Luo, "Data Dimension Reduction makes ML Algorithms efficient," 2022 International Conference on Emerging Technologies in Electronics, Computing and Communication (ICETECC), Jamshoro, Sindh, Pakistan, 2022, pp. 1-7, doi: 10.1109/ICETECC56662.2022.10069527.

[6] F. Ren, Y. Liu, J. Zhang, J. Qian, P. Gao and H. Li, "Research on garment image classification algorithm based on machine learning," 2021 3rd International Academic Exchange Conference on Science and Technology Innovation (IAECST), Guangzhou, China, 2021, pp. 264-268, doi: 10.1109/IAECST54258.2021.9695531.

[7] S. Bhatnagar, D. Ghosal and M. H. Kolekar, "Classification of fashion article images using convolutional neural networks," 2017 Fourth International Conference on Image Information Processing (ICIIP), Shimla, India, 2017, pp. 1-6, doi: 10.1109/ICIIP.2017.8313740.

[8] J. Guo and X. Wang, "Image Classification Based on SURF and KNN," 2019 IEEE/ACIS 18th International Conference on Computer and Information Science (ICIS), Beijing, China, 2019, pp. 356-359, doi: 10.1109/ICIS46139.2019.8940198.

[9] W. Man, Y. Ji and Z. Zhang, "Image classification based on improved random forest algorithm," 2018 IEEE 3rd International Conference on Cloud Computing and Big Data Analysis (ICCCBDA), Chengdu, China, 2018, pp. 346-350, doi: 10.1109/ICCCBDA.2018.8386540.

[10] A. Alkandari and S. J. Aljaber, "Principle Component Analysis algorithm (PCA) for image recognition," 2015 Second International Conference on Computing Technology and Information Management (ICCTIM), Johor, Malaysia, 2015, pp. 76-80, doi: 10.1109/ICCTIM.2015.7224596.

[11] R. R. Selvaraju, M. Cogswell, A. Das, R. Vedantam, D. Parikh and D. Batra, "Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization," 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 2017, pp. 618-626, doi: 10.1109/ICCV.2017.74.