Image Classification on Fashion Mnist Dataset

Eldar Eyvazov
eldar.eyvazov@studenti.unipd.it

1. Introduction

The objective of this project is to evaluate the performance of various machine learning algorithms on the Fashion MNIST dataset. Subsequently, neural network models will be applied to the same dataset. The results of these approaches will be compared to determine the most effective method for fashion image classification. The objective of the resulting model is to achieve robust recognition of images, demonstrating flexibility to accurately classify previously unseen images. To accomplish this, we will conduct a detailed analysis of the characteristics of the errors to discern whether they are systematic and informative or merely ambiguous and random.

1.1. Results

Ultimately, we will demonstrate that a Convolutional Neural Network (CNN) can achieve robust image recognition by effectively extracting features from images. These features are then processed by a dense Neural Network, which enables accurate classification. This approach highlights the CNN's capability to handle and recognize even previously unseen images.

1.2. Dataset

The Fashion MNIST dataset is a large collection of 70,000 grayscale images, each representing one of 10 different categories of fashion items such as T-shirts, trousers, bags, and shoes. Each image in the dataset has a resolution of 28x28 pixels. The dataset is divided into 60,000 training images and 10,000 test images.

Before feeding the images into the machine learning models, the pixel values are rescaled to be in the range of 0 to 1. This rescaling is done by dividing each pixel value by 255, which is the maximum pixel value. Rescaling the pixel values is an important preprocessing step as it normalizes the data, making the learning process more efficient and stable. Normalized data helps in faster convergence during the training phase and can lead to improved performance of the neural network models by ensuring that the input features have a consistent scale. The dataset is divided into three parts: training, validation, and test sets. Initially, 60,000 images are used for training, and 10,000 images are reserved for testing. The training set is further split, with 75 percent used for training and 25 percent for validation. This ensures that the model's performance can be reliably evaluated on unseen data.

2. Methodology

During the project, many different methods have been used and compared to see how they respond and perform in image classification. The methods evaluated, in order, are:

1. Logistic Regression
2. Perceptron
3. Decision Tree and Random Forest
4. Support Vector Machine (SVM)
5. Dense Neural Network
6. Convolutional Neural Networks (CNNs)

All of these models have been trained by iteratively tuning their parameters on the validation set to identify the optimal configuration.

Towards the end of the project, more sophisticated models were employed to determine if they provided any performance improvements. Specifically, a Dense Neural Network with a "funnel" architecture was used, starting with 784 input neurons and gradually reducing to 10 output neurons. Additionally, Convolutional Neural Networks (CNNs) were utilized to assess whether this renowned model for image classification could surpass our best-performing model.

3. Models

3.1. Logistic Regression

Logistic Regression is a statistical model that in its basic form uses a logistic function to model a binary dependent variable. In the context of this project, it was employed for image classification tasks. The model was trained to distinguish between different categories of fashion items in the Fashion MNIST dataset.

To enhance its performance, the Logistic Regression model's hyperparameters were fine-tuned using the validation set. This iterative tuning process aimed to find the optimal configuration that maximized the classification accuracy. The best configuration was found with $C = 0.1$, achieving a training accuracy of 86.78% and a validation accuracy of 85.69%. The performance of Logistic Regression was then compared with other models to evaluate its efficacy in the context of image classification. The test accuracy of the logistic regression model is 84.21%. The small gap between the test accuracy and the previously reported training and validation accuracies suggests that overfitting is minimal. Such a small difference indicates that the model generalizes well to unseen data, and the performance drop is not significant enough to conclude that there is substantial overfitting.

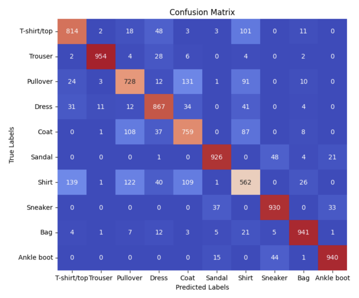

Figure 1. Confusion matrix for the logistic regression model.

Figure 1 shows the confusion matrix for the logistic regression model. The model demonstrates high accuracy in categories like "Sneaker," "Sandal," "Ankle Boat", and "Trouser," with many correct predictions. However, there is noticeable confusion between "Shirt," "T-shirt/Top," and "Pullover," "Coat" indicating difficulty in distinguishing these similar items. Overall, the model performs well but struggles with certain categories that have overlapping features. As we know, the main diagonal of the confusion matrix represents correct classifications. In Figure 1, these values are high across most categories, indicating that the logistic regression model performs well overall. However, the off-diagonal values show misclassifications, highlighting areas for improvement.

3.2. Perceptron

The perceptron model is a simple linear classifier used for binary classification tasks. In this project, it was applied to a multi-class classification problem. The best configuration was training the model with early stopping enabled and without fitting the intercept, achieving a training accuracy of 84.43% and a validation accuracy of 83.21%. The test accuracy of the perceptron model is 82.65%. The percentage difference between the training accuracy (84.43%) and the test accuracy (82.65%) is approximately 2.1%. This small gap indicates that the data is quite linearly separable and suggests that there is no significant overfitting.

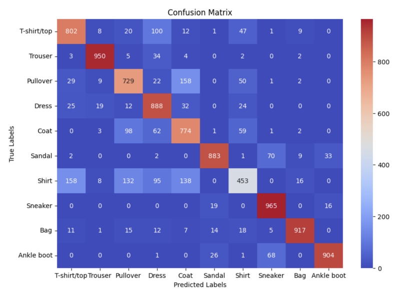

Figure 2. Confusion matrix of Perceptron.

Figure 2 shows the confusion matrix for the Perceptron model. The model demonstrates high accuracy in categories like "Sneaker," "Trouser," "Sandal," "Dress" and "Bag." However, there is significant confusion between "Shirt," "T-shirt/top," and "Pullover." Overall, the Perceptron model performs well but struggles with certain categories that have overlapping features.

3.3. Decision Trees

Decision Trees were utilized to perform image classification tasks. And again various hyperparameters were tuned to identify the optimal configuration, focusing on 'maximum depth' and 'mininimum samples split'. After evaluating multiple configurations, the best performance was achieved with 'maximum depth' set to 10 and 'mininimum samples split' set to 3. This configuration resulted in a training accuracy of 85.14% and a validation accuracy of 80.74% and test accuracy of 79.73%. The slight decrease in accuracy from the training set to the validation set and further to the test set suggests that the model may be overfitting to some extent. The model performs better on the training data compared to unseen data, as indicated by the higher training accuracy. However, the performance difference is not very large, indicating that while overfitting is present, it is not extreme. This suggests that the model's ability to generalize to new data is relatively good.

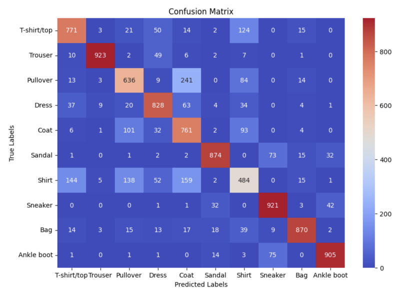

Figure 3. Confusion matrix of Decision Tree

Figure 3 shows the confusion matrix for the Decision

Tree model. The model demonstrates high accuracy in categories like ”Sneaker,” ”Trouser,” ”Sandal,” ”Bag,” and ”Ankle boot.” However, there is significant confusion between ”Shirt,” ”T-shirt/top,” and ”Pullover,” as well as some misclassifications in the ”Coat” category. Overall, the Decision Tree model performs well but struggles with distinguishing between certain similar categories.

3.4. Random Forest

The Random Forest model was evaluated with various configurations, and the best hyperparameters were found to be 200 estimators and a maximum depth of 20. This configuration achieved a training accuracy of 99.46% and a validation accuracy of 88.13%. The test accuracy of the Random Forest model is 87.34%.

The percentage difference between the training accuracy (99.46%) and the test accuracy (87.34%) is approximately 12.12%, indicating that the model fits the training data almost perfectly but significantly overfits. Additionally, the training time was considerably slow due to the complexity and number of trees in the model.

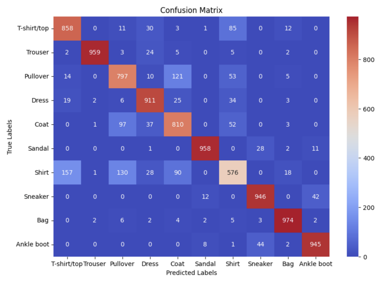

Figure 4. Confusion matrix of Random Forest

Figure 4 shows the confusion matrix for the Random Forest model. The model demonstrates high accuracy in categories like ”Sneaker,” ”Trouser,” ”Sandal,” ”Bag,” and ”Ankle boot.” However, there is significant confusion between ”Shirt,” ”T-shirt/top,” and ”Pullover,” as well as notable misclassifications in the ”Coat” category. The ”Shirt” category is the worst classified, with many instances misclassified as ”T-shirt/top” and ”Pullover.” Despite high training accuracy, the considerable misclassifications and overfitting indicate that the model struggles with distinguishing between certain similar categories and does not generalize well to unseen data.

3.5. SVM

The Support Vector Machine (SVM) model was evaluated with various configurations, trying both linear and RBF kernels. The best hyperparameters were found to be an RBF kernel with $C = 4$. This configuration achieved a training accuracy of 94.84% and a validation accuracy of 90.15%. The test accuracy of the SVM model is 89.25%.

The percentage difference between the training accuracy (94.84%) and the test accuracy (89.25%) is approximately 5.59%, indicating that the model overfits the training data. Considering all models we have implemented so far, the SVM model shows the second highest degree of overfitting, with the Random Forest model being the most overfitted. Additionally, the SVM model was the slowest to train due to the computational complexity of the algorithm.

Between the two kernels tried, the RBF kernel outperformed the linear kernel in terms of validation accuracy.

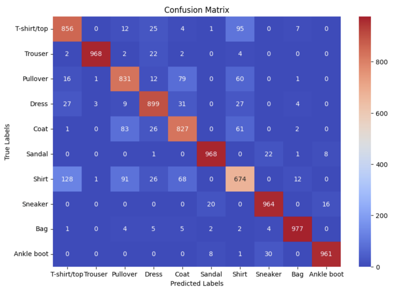

Figure 5. Confusion matrix of SVM

Figure 5 shows the confusion matrix for the SVM model. If we compare our two most overfitting models, the SVM and the Random Forest, we can see that the SVM model demonstrates high accuracy in categories like ”Sneaker,” ”Trouser,” ”Sandal,” ”Bag,” ”Ankle boot,” and ”Dress.” However, there is significant confusion between ”Shirt,” ”T-shirt/top,” and ”Pullover,” as well as some misclassifications in the ”Coat” category. The SVM model has fewer misclassifications overall compared to the Random Forest model and handles certain categories, like ”Bag” and ”Ankle boot,” better. However, both models struggle with the ”Shirt” category, highlighting a common challenge for all classifiers we have used so far in distinguishing similar types of clothing items with overlapping features.

3.6. Dense NN

For the Fashion MNIST dataset, we designed a dense neural network using a funnel approach. Starting with 784 input neurons to match the 28x28 pixel images, the network gradually reduces the number of neurons: from 256 in the first dense layer, down to 128, then 64, and finally to 10 output neurons, which correspond to the number of classes. We applied batch normalization after each dense layer to stabilize and improve training performance.

To optimize the training process, we employed early stopping and learning rate reduction on plateau. Early stopping monitored the validation loss and stopped training when no further improvement was detected, preventing overfitting. The learning rate reduction on plateau decreased the learning rate when the validation accuracy plateaued, helping the model to converge more effectively.

After experimenting with different batch sizes, we settled on a learning rate of 0.001 and a batch size of 16 for the best performance.

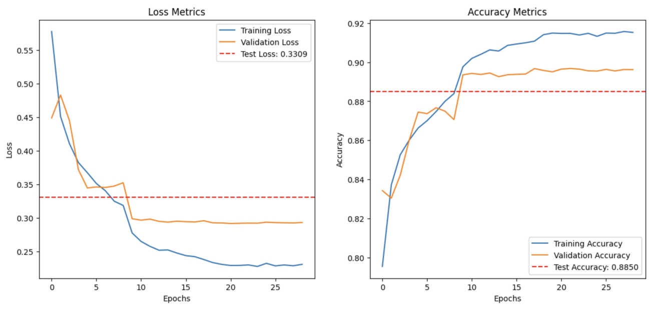

Figure 6. Dense NN Accuracy Metrics and Loss

The dense neural network demonstrated a solid learning curve. We first set the epochs at 100, but training stopped due to the early stopping mechanism. The loss graph on the left shows both training and validation loss continuously reducing, with the final test loss being 0.3309. The accuracy graph on the right indicates that training and validation accuracy improved dramatically in the early epochs before plateauing. The ultimate test accuracy was 88.50%. When comparing the training accuracy, which peaked at around 91%, to the test accuracy of 88.50%, the difference is only about 2.5%, demonstrating that the model generalises effectively to new data. However, because the validation measures have plateaued, there may still be some potential for fine-tuning.

3.7. Convolutional Neural Networks

Our final model to evaluate is a convolutional neural network (CNN). As expected, this model took longer to train compared to the others due to its complexity and additional layers. We used the same batch size of 16 and learning rate of 0.001 as in our dense neural network, and the choice of optimizer was Adam.

The CNN starts with a convolutional layer with 32 filters and a 3x3 kernel, followed by a ReLU activation and max-pooling layer. This is followed by another convolutional layer with 64 filters and a similar setup. The output is then flattened and passed through a dense layer with 128 neurons and ReLU activation. Finally, the output layer has 10 neurons with a softmax activation to ensure the probabilities sum to 1, which is crucial for multi-class classification.

Again, we used batch normalization to keep training stable and a learning rate scheduler to reduce the learning rate when improvements plateaued. Despite the longer training time, this model's complexity helps it capture detailed patterns in the data, making it a strong choice for our task.

Figure 6 presents the confusion matrix for our convolutional neural network (CNN) model, which stands out as our top-performing model. The CNN shows high accuracy

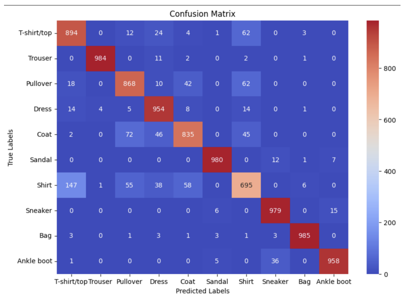

Figure 7. CNN Confususion Matrix

across most categories, excelling in "Sneaker," "Trouser," "Sandal," "Bag," and "Ankle boot."

A notable improvement is seen in the "Shirt" category, with more correct classifications compared to earlier models. Although there is still some mix-up between "Shirt" and "T-shirt/top," it's significantly reduced.

Categories like "T-shirt/top" and "Pullover" also perform well, despite some occasional misclassifications into similar categories. The "Dress" category continues to achieve high accuracy, and "Coat" shows good performance with some expected confusion into similar items like "Pullover" and "Dress."

Overall, this confusion matrix showcases the CNN model's superior ability to classify images accurately, making it the best model we've tested. It captures detailed patterns more effectively than the others, significantly enhancing performance in challenging categories like "Shirt."

Figure 7 shows the training and validation metrics for our convolutional neural network (CNN) model. In the left graph, both the training and validation loss decrease steadily, with the final test loss at 0.2477. The right graph highlights the accuracy metrics, where the training accuracy climbs to about 96% and the validation accuracy stabilizes around 91%, leading to a test accuracy of 91.16%.

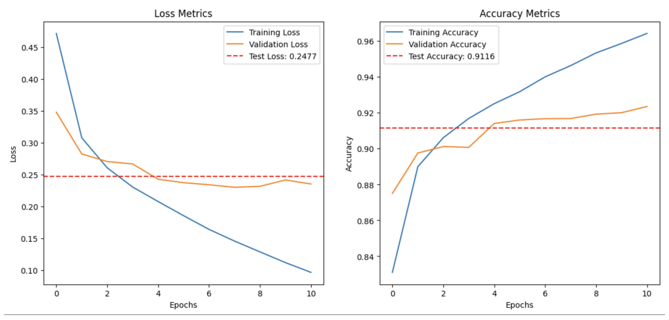

Figure 8. CNN Accuracy Metrics and Loss

The small difference between the training and test accuracy, roughly 4.84%, suggests there is no substantial overfitting. Compared to our dense neural network, the CNN converges much faster, reaching high accuracy within just a few epochs. This quick convergence and strong performance make the CNN our best model yet.

3.8. Conclusions

To wrap things up, we tested a range of models including Logistic Regression, Perceptron, Decision Tree, Random Forest, SVM, Dense Neural Network, and Convolutional Neural Network (CNN). Among these, the CNN stood out, delivering the highest accuracy on both validation and test sets. This suggests that CNNs are particularly well-suited for image classification tasks like those in the Fashion MNIST dataset. Looking ahead, there's potential to make these models even better by fine-tuning hyperparameters and trying out new techniques.

Model	**Validation Accuracy**	**Test Accuracy**
Logistic Regression	83.21%	82.65%
Perceptron	83.21%	82.65%
Decision Tree	80.74%	79.73%
Random Forest	88.13%	87.34%
Support Vector Machine (SVM)	90.15%	89.25%
Dense Neural Network	89.10%	88.50%
Convolutional Neural Networks (CNNs)	91.00%	91.16%

Table 1. Model Accuracies for Validation and Test Sets

fancyhdr [C]5