

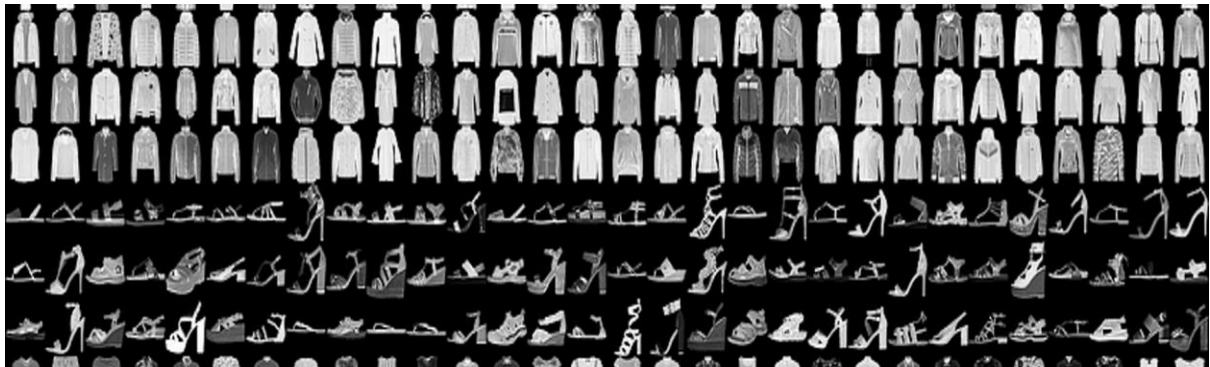
Report on ML Project Fashion MNIST Dataset Classification

Aarib Ahmed Vahidy 22K-4004

Arham Hussain Khan 22K-4080

Partham Kumar Chawla 22K-4079

Custom Neural Network for Fashion MNIST Classification



1. Introduction

The Fashion MNIST dataset consists of 70,000 grayscale images of 28x28 pixels, each representing one of ten classes of clothing items. The goal of this project was to design and implement a custom neural network to classify these images accurately. This report outlines the approach used, the results obtained, and the steps taken to optimize the model's performance.

2. Approach

Dataset Preparation

The dataset was split into training, validation, and testing sets:

Training set: 55,000 samples

Validation set: 5,000 samples

Testing set: 10,000 samples

Report: Fashion Mnist Classification

The pixel values were normalized by dividing by 255 to scale them between 0 and 1, improving convergence during training.

Data Augmentation

To improve the model's generalization and reduce overfitting, data augmentation was applied to the training data. This technique involves creating additional variations of the original images by applying random transformations, such as rotation, flipping, and zooming. These transformations increase the diversity of the training set, enabling the model to learn more robust features.

The augmented dataset was generated in real-time during training using TensorFlow's ImageDataGenerator. These augmented images replaced the original training set during the training process, effectively creating a larger and more diverse dataset.

3. Model Design

For this project, three different neural network architectures were tested to classify images from the Fashion MNIST dataset. The models were designed to learn patterns from images of clothing items and predict their respective categories.

1. Initial Model: The initial model started with a simple structure consisting of two dense layers. The first layer flattened the 28x28 pixel images into a single one-dimensional vector to be fed into the network. The first dense layer applied the ReLU activation function, followed by a second dense layer with the Tanh activation function. The output layer used the softmax activation function to generate class probabilities, allowing the model to predict one of the ten clothing categories.
2. Batch Normalized Model: To improve stability and convergence during training, a second model incorporated batch normalization. Batch normalization helps in stabilizing the learning process by normalizing the activations of each layer. This architecture had the following structure:
 - a. Input layer: Flattened the 28x28 pixel images into a one-dimensional vector.
 - b. Hidden layers: The first hidden layer had 512 neurons with a ReLU activation function, followed by batch normalization. The second hidden

layer had 256 neurons, again with ReLU activation and batch normalization. The third hidden layer had 128 neurons with ReLU activation.

- c. Output layer: The output layer had 10 neurons, corresponding to the 10 possible categories, and used the softmax activation function to generate class probabilities.

- 3. **Optimized Model:** The third model was designed with a more complex architecture, consisting of three dense layers. Each dense layer applied the ReLU activation function, which is effective for learning complex patterns in the data. The model had more neurons in the layers compared to the initial model: the first layer had 300 neurons, the second had 100 neurons, and the output layer had 10 neurons (one for each class). This design aimed to improve the model's learning capacity, allowing it to perform better on the classification task. This is the model we decided to use due to its good results and shorter training time. If we were to use a more complex model it would give better results but would lead to higher training times and would require greater computational resources.

In all models, the optimizer used was SGD (Stochastic Gradient Descent), which adjusts the model's weights based on the gradients computed during training, SGD uses a constant learning rate by default. The loss function was sparse categorical crossentropy, which is suitable for multi-class classification problems like FASHION MNIST, and the metric used for evaluation was accuracy

This design allowed for experimentation with different levels of complexity and regularization to optimize the model's performance on the Fashion MNIST dataset.

4. Results & Optimization

Model Performance

After training the custom neural network on the Fashion MNIST dataset, the model was evaluated on the test set. The results indicate that the model achieved an accuracy of approximately 87 % on the test set, with a loss of 0.36. This is a good performance, though there is room for further improvement through a more complex model.

Report: Fashion Mnist Classification

To assess the quality of the model's predictions in more detail, several metrics were used:

Classification Report: The classification report provides precision, recall, and F1-score for each class. It shows that the model performs well in identifying certain clothing types, such as "Trouser," "Sandal," and "Sneaker," with high recall and precision. However, the model struggled with classes like "Pullover" and "Shirt," where both precision and recall were relatively low. According to us this could be due to the similarity in the pictures for pullover and shirt in the dataset.

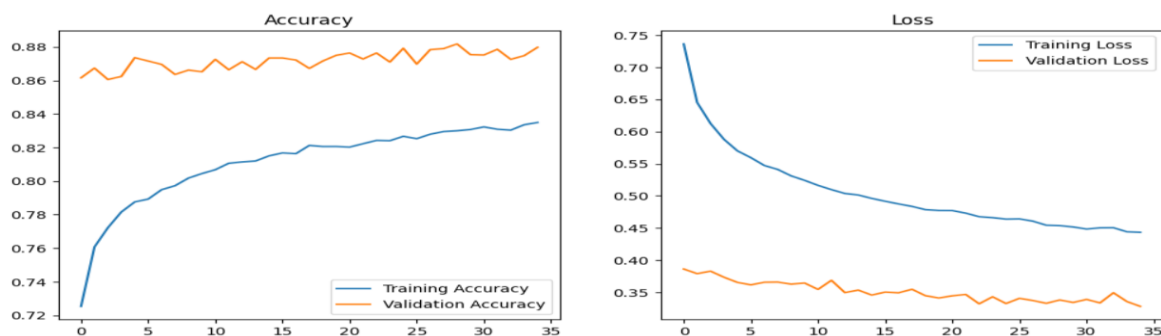
Confusion Matrix: The confusion matrix visually demonstrates how well the model is able to distinguish between the different classes. It helps identify any systematic errors in the model's predictions. For example, the "Pullover" and "Shirt" classes seem to be confused with others in the dataset, which is something that can be addressed in future iterations of the model.

ROC Curve and AUC: The ROC curve provides insight into the trade-off between true positive rate (TPR) and false positive rate (FPR) across different thresholds. In this case, the ROC AUC for the first class was calculated, showing the model's ability to discriminate between classes. The AUC of the ROC curve is a useful metric for multi-class classification problems, as it summarizes the model's performance across all possible classification thresholds.

Model Evaluation Plots:

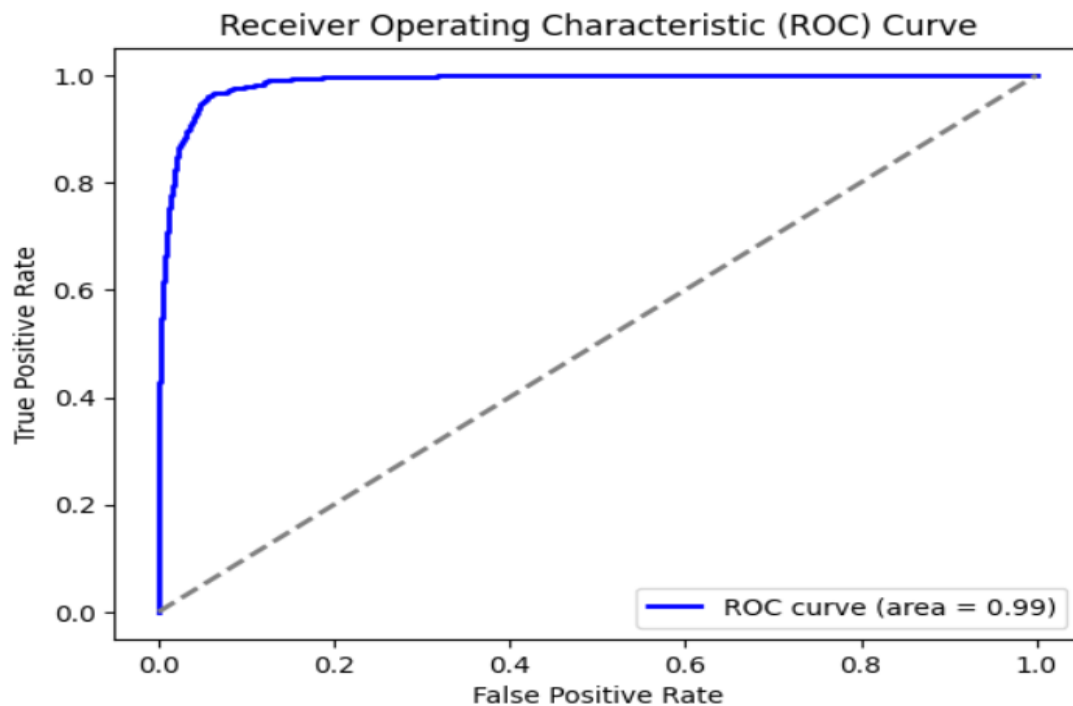
Several evaluation plots were created to better understand the model's training progress:

Accuracy and Loss Curves: These plots show how the model's accuracy and loss changed over time during both training and validation. The accuracy plot indicates a consistent increase in model performance, while the loss plot demonstrates a steady decrease, signalling successful learning. By comparing training and validation curves, it's clear that the model did not overfit, as the validation loss and accuracy follow the trends in the training loss and accuracy.



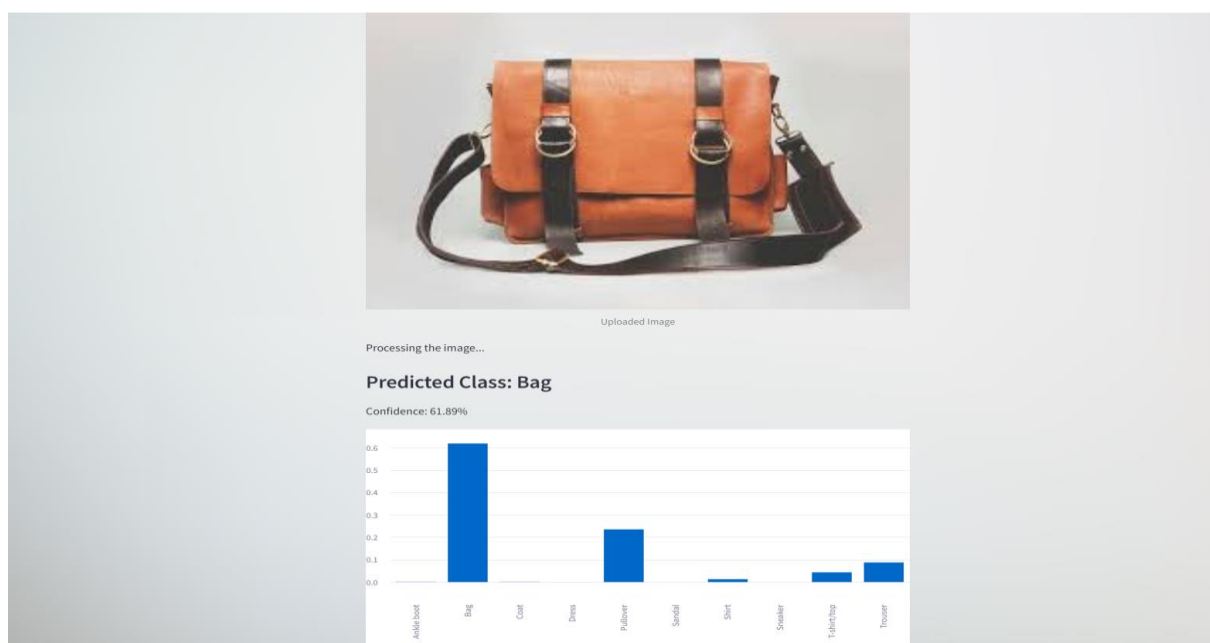
Report: Fashion Mnist Classification

ROC Curve: The ROC curve for the first class in the dataset was plotted to visualize the model's ability to distinguish between different clothing types. The area under the curve (AUC) for the class provides a quantitative measure of the model's classification performance across different thresholds.



These metrics and plots provided a comprehensive view of the model's performance and helped identify areas where further optimizations could be made.

Furthermore, Model's accuracy was also tested by providing test pics to the model:



Conclusion

In this project, we successfully built a custom neural network for classifying images from the Fashion MNIST dataset. The model achieved a good balance between complexity and performance. With an accuracy of 87 % on the test set, the model demonstrates the potential of using neural networks for image classification tasks.

Despite the promising results, several improvements can be made to increase accuracy further. Hyperparameter tuning, data augmentation, and regularization techniques could help address areas where the model currently struggles, such as distinguishing between certain clothing types.