Assignment 4 - Deep Learning Analysis on the MNIST Fashion Dataset

Tanushree Kumar | tanushree.kumar@outlook.com

DATA 640 - Fall 2024

Professor Steve Knode

University of Maryland Global Campus

Due: October 8, 2024

Introduction and Data Set Description

This analysis focuses on building a deep learning model for image recognition using the Fashion MNIST dataset. The problem being addressed is the accurate classification of fashion items into specific categories based on their visual characteristics. Unlike the classic MNIST dataset, which classifies handwritten digits, the Fashion MNIST dataset provides a more challenging task, as it contains images of various clothing items, such as shirts, trousers, and shoes. The purpose of the model is to develop a neural network that can distinguish between these 10 different categories of fashion items based on grayscale images. Traditional computer vision tasks, like recognizing clothing items in photos, have been challenging due to the variety of styles, shapes, and perspectives from which items can be viewed. Developing a model that can accurately classify fashion items can have practical applications in the fashion industry, such as enhancing the search functionality on e-commerce platforms, automating inventory management, and improving the user experience through better-personalized recommendations.

The dataset used for this assignment is the Fashion MNIST dataset, provided by Zalando Research. It consists of 70,000 grayscale images of fashion items, with each image having a resolution of 28x28 pixels. The dataset is divided into a training set of 60,000 images and a test set of 10,000 images. Each image is associated with a label that indicates one of 10 categories, including T-shirts, trousers, pullovers, dresses, coats, sandals, shirts, sneakers, bags, and ankle boots.

The images are stored as 2D arrays, with pixel values ranging from 0 to 255, where 0 represents black and 255 represents white. The labels are integers from 0 to 9, each corresponding to a specific fashion category. This dataset provides a standardized benchmark for testing deep learning models on image recognition tasks, offering a more complex challenge than

basic digit recognition of the original MNIST dataset due to the diversity in fashion items. A figure of the dataset can be seen in Appendix A which shows a few rows of pixel arrays alongside their corresponding labels, giving a glimpse into the data that the model will use for training and evaluation.

The method chosen for this analysis is deep learning, specifically using a convolutional neural network (CNN). This approach is highly suitable for image recognition tasks like the Fashion MNIST dataset because CNNs are designed to automatically and adaptively learn spatial hierarchies of features from images. CNNs excel in capturing patterns like edges, textures, and shapes, making them effective for classifying images into different categories. For this problem, a deep learning approach allows the model to learn complex features directly from the image data, reducing the need for manual feature extraction. The ability of CNNs to handle large image datasets makes them the ideal choice for this analysis, ensuring high accuracy in predicting fashion item categories.

CNNs consist of multiple layers, including convolutional layers that apply filters to extract features, pooling layers that reduce dimensionality, and fully connected layers that perform the final classification. The principle behind CNNs is to mimic the way humans perceive visual information by focusing on local patterns and gradually learning more abstract representations through deeper layers. Key parameters in the model include the number of filters, filter size, pooling size, and learning rate. Adjusting these parameters allows for tuning the model to achieve a balance between training time and classification accuracy, enabling the model to learn meaningful patterns while avoiding overfitting.

Data Cleansing and Preparation

The data used in this analysis is the Fashion MNIST dataset, hosted on GitHub by Zalando Research. It is a curated dataset that serves as a direct drop-in replacement for the original MNIST dataset of handwritten digits. Each data point is a 28x28 pixel grayscale image representing an article of clothing, such as a T-shirt, a coat, or a shoe. The pixel values range from 0 (black) to 255 (white), forming a 2D array. The target variable is a numerical label (0-9) that identifies the clothing category. For example, a label of "0" represents a T-shirt/top, while "1" represents trousers. The training set comprises 60,000 images, and the test set includes 10,000 images, making it a substantial dataset for training and evaluating a deep learning model.

General preprocessing included normalizing the pixel values from a range of 0-255 to 0-1, which is critical for improving the performance of neural networks by ensuring faster convergence. This was done by dividing the pixel values by 255. Additionally, the data was reshaped to include a channel dimension, a requirement for input into CNNs, which expect 3D input tensors. Specific preprocessing for this model also included converting the target labels into a one-hot encoded format, which allows the model to output probabilities for each class during training.

Development of Deep Learning Models

The data was already partitioned into a training and validation split of 60,000 and 10,000 images respectively. The majority of the data in the training set allows for better and more accurate models but also increases the likelihood of overfitting. To check if any of the models are overfitting, the accuracy for the training and validation set will be observed since an overfitting model performs well on the training set but poorly on the validation set.

The model from the original python script (Knode, 2024) was used as a base to compare against the models being built to assess the changes and tweaks in the parameters. The parameters can be adjusted in multiple ways such as tweaking: the number of layers; the type of layers; the order of the layers; the kernel size; the steps; pool size and average vs. max pooling; the epochs; and the batch size.

To present, compare, and experiment with different CNN models for this dataset, the provided baseline model will be modified to create six additional CNN models. Each of these models builds upon the baseline by either increasing depth, adding more filters, using larger kernel sizes, or introducing additional dense layers. These variations are designed to explore the balance between model complexity, training stability, and generalization to improve the accuracy and robustness of image classification on the Fashion MNIST dataset. Table 1 in Appendix A shows the differences in each model.

The baseline model starts with a relatively simple architecture. It includes two Conv1D layers with 236 and 128 filters, each followed by a MaxPooling1D layer for down-sampling. Dropout layers (set at 0.2) are included to reduce overfitting, and batch normalization helps stabilize training by normalizing activations. The model ends with a Flatten layer to convert the pooled feature maps into a 1D array, followed by a dense output layer with 10 units (one per class) and a softmax activation function. This simple structure is designed to establish a starting point for the model's performance, focusing on basic feature extraction from the input images.

Model 1 expands on the baseline model by adding an extra Conv1D layer with 64 filters. The addition of this third convolutional layer allows the model to learn more complex features and patterns. It retains the same pooling, dropout, and batch normalization strategy as the baseline but with a slightly increased model depth. This additional depth may enable better

performance, particularly in capturing more detailed information from the input images, without making the model overly complex.

Model 2 modifies the architecture by increasing the number of filters in the Conv1D layers to 256, 128, and 64, respectively. This increases the model's capacity to learn more nuanced features by using a higher number of filters for feature extraction. With this change, the model becomes more sensitive to finer patterns within the images, although this can also increase the risk of overfitting. The model uses the same dropout rate (0.2) and retains batch normalization to mitigate the potential overfitting due to the increased parameter count.

Model 3 experiments with a larger kernel size of 3 in all three Conv1D layers. By expanding the kernel size, the model can capture broader spatial patterns in the images, which might help with recognizing larger features. While a larger kernel size can provide a better context for each pixel by considering more neighboring pixels, it also increases the computational cost of the model. Dropout (0.3) and batch normalization are used to stabilize training and maintain a balance between model complexity and generalization.

The first 3 models weren't as accurate as desired, which will be discussed in the Results section below. As a result, the next 3 models were built much differently to produce a desirable accuracy of 90% or even higher if possible. Models 4, 5, and 6 were built using the 2D convolutional layer. More parameters were also explored for more control over the model's outcome.

Model 4 used three convolutional layers, each with an increasing number of filters: 32, 64, and 64. The use of multiple convolutional layers allows for progressively deeper feature extraction, which can help capture more complex patterns in the data. Each convolutional layer is followed by a 2x2 max-pooling layer, which reduces the spatial dimensions, thereby lowering the

computational complexity. After the convolutional layers, the model uses a dense layer with 64 units to capture higher-level representations before outputting probabilities for each class through a 10-unit softmax layer. The model is trained for 10 epochs with a batch size of 64, using the Adam optimizer, which can adapt the learning rate dynamically for improved convergence.

Model 5 follows a similar structure as Model 4 but introduces an additional convolutional layer with 128 filters, making it more complex. The higher filter count allows this model to capture more intricate details and features from the images. After three convolutional layers and their corresponding max-pooling layers, the model uses a dense layer with 128 units, which offers a larger capacity for learning relationships before the final classification. Like Model 4, it uses the Adam optimizer and is trained for 10 epochs with a batch size of 64, making it suitable for similar computational capabilities but with a potentially higher capacity to learn complex features.

Model 6 builds on the architecture of Model 5 but is trained for more epochs—25 instead of 10—allowing for further refinement of weights through more iterations over the training data. Increasing the epochs was the only adjustment done since "accuracy tends to increase with the number of epochs as the model continues to refine its understanding of the training data" (*How Does Epoch Affect Accuracy in Deep Learning Model?*, 2023). This extended training helps the model better capture the relationships between features, potentially leading to improved accuracy, especially for more challenging data points. The model's structure remains identical to Model 5, with three convolutional layers (32, 64, and 128 filters) and a dense layer with 128 units, followed by a 10-unit softmax layer for output. The increased training time can help the model generalize better, but it may also require careful monitoring to avoid overfitting.

The first three models (Model 1, Model 2, and Model 3) focused on simplicity with fewer convolutional layers and filters, which made them computationally efficient and quick to train. However, their simpler architecture limited their ability to capture complex image features, resulting in relatively lower accuracy. For example, the Baseline model employed a Conv1D structure, which, while suitable for sequence data, was not fully optimized for the 2D spatial features in image data. This simplicity is advantageous in scenarios with limited computational resources but falls short for more intricate tasks.

The new three models (Model 4, Model 5, and Model 6) leverage a more sophisticated architecture with multiple Conv2D layers and an increasing number of filters, allowing them to learn from complex image patterns more effectively. The addition of a 128-filter layer in Models 5 and 6 further enhances their ability to identify fine-grained details. Moreover, the use of more epochs in Model 6 offers a chance for better optimization, although it comes with a risk of overfitting.

Results

The results table below shows the summary and relevant statistics for all the models.

Model Name	Loss	Accuracy	Precision	Recall	F1-Score
Baseline	0.9232	0.6832	0.8488	0.844	0.8427
Model 1	0.3822	0.8586	0.8664	0.8586	0.8609
Model 2	0.3197	0.8807	0.883	0.8807	0.8802
Model 3	0.3197	0.8832	0.8834	0.8832	0.8813
Model 4	0.2779	0.9025	0.9048	0.9025	0.9031
Model 5	0.2662	0.9126	0.9124	0.9126	0.9124
Model 6	0.5074	0.904	0.9048	0.904	0.9041

Table 2. Table of results for all 6 models.

The baseline model achieved a loss of 0.9232 and an accuracy of 0.6832. This model serves as a reference point to evaluate the effectiveness of the subsequent models. Its relatively

high loss and low accuracy indicate that the model struggles with classifying images correctly, due to its simple architecture. Improvements in model architecture should reduce loss and increase accuracy which can be seen in the subsequent models.

Model 1 improved upon the baseline significantly, achieving a loss of 0.3822 and an accuracy of 0.8586. This improvement suggests that the changes made to the architecture, such as adding more convolutional layers and tuning the learning rate, allowed the model to learn more meaningful features from the data. However, it still falls short of optimal performance, indicating room for further refinements.

Model 2 further reduces the loss to 0.3197 and increases the accuracy to 0.8807. The addition of another convolutional layer and possibly more filters enhances the model's ability to extract complex features, leading to better generalization. This is evident in the smaller gap between training and test accuracy, which suggests that the model is not overfitting significantly.

Model 3 builds upon the architecture of Model 2, offering a similar structure with more training iterations. It achieved a test accuracy of approximately 88.07%, indicating that it correctly classified about 88% of the examples in the test set. The loss of 0.3197 shows a lower error rate in its predictions compared to the baseline model. Model 3's performance is likely influenced by factors such as the architecture and the number of epochs. Compared to Model 2, the longer training time (more epochs) allowed the model to better learn from the training data, refining its ability to classify the Fashion MNIST dataset. However, this marginal gain over Model 2 suggests that the increased training did not drastically improve performance, possibly indicating that the model is reaching its learning capacity for this dataset with the given architecture.

Model 4 achieves a loss of 0.2779 and an accuracy of 0.9025, marking a significant improvement over previous models. The architecture includes three convolutional layers with 32, 64, and 64 filters respectively, each followed by max pooling. The addition of multiple convolutional layers allows the model to extract more complex patterns and textures from the images, which is crucial for differentiating between classes in a dataset like Fashion MNIST. This model strikes a balance between complexity and generalization, as evidenced by its relatively high precision and recall scores across most classes. Notably, the model struggles with the "Shirt" class (precision: 0.69, recall: 0.78), which might be due to similarities with other clothing categories like "Pullover" and "Coat."

Model 5 further refines the architecture by introducing a 128-filter convolutional layer, achieving a loss of 0.2662 and an accuracy of 0.9126. This increase in filter size allows the model to capture even more detailed features from the images. The improvement in accuracy demonstrates that the added complexity enables the model to better differentiate between more subtle features of the clothing items. Precision and recall scores are consistently high across all classes, including a marked improvement in the "Shirt" category (precision: 0.77, recall: 0.74). This suggests that the model is better able to distinguish shirts from similar categories, likely due to the additional feature extraction capabilities provided by the deeper convolutional layer.

Model 6 retains the same architecture as Model 5 but trains for a longer duration (25 epochs instead of 10). Despite this, it shows a higher loss of 0.5074 with an accuracy of 0.9040. The longer training period could have led to overfitting, where the model learned too many details of the training data but lost generalization capability when applied to the test set. This is further supported by the precision and recall scores, which are slightly lower for some classes compared to Model 5. While Model 6 maintains high performance in some categories like

"Sandal" and "Bag," its performance drops slightly in more challenging categories like "Shirt" and "Pullover." This result suggests that for some models, increasing training time alone is not sufficient for better generalization and may require further architecture adjustments or regularization techniques.

The first three models progressively improve upon the baseline performance by refining the model architecture with additional layers and more filters, resulting in higher accuracy and lower loss. However, these models do not surpass the 90% accuracy mark, indicating that they still struggle with learning the more nuanced features of the dataset.

In contrast, the last three models (Models 4, 5, and 6) introduce more complexity into the architecture through deeper layers and larger filter sizes, allowing them to reach over 90% accuracy. Model 5, in particular, achieves the highest accuracy (0.9126) with a balanced performance across most classes. The transition to deeper architectures clearly helps in extracting more detailed features, but it comes with the risk of overfitting, as seen in Model 6. While Model 6 trains for more epochs, it does not generalize as well as Model 5, illustrating that deeper models must be carefully managed with respect to training duration and regularization to achieve optimal performance.

The progression from the baseline model to Model 5 demonstrates the positive impact of increasing model complexity and training time. However, it also highlights the delicate balance required to prevent overfitting and maintain robust generalization on unseen data. Model 5 emerges as the best-performing model, striking a balance between architecture complexity, training duration, and overall accuracy.

Conclusions

This analysis aimed to develop a deep learning model for classifying fashion items using the Fashion MNIST dataset. The primary goal was to create a model capable of accurately categorizing images into one of ten fashion categories, improving upon a baseline model with more complex architectures. Through iterative adjustments to model parameters and architecture, Model 5 emerged as the champion model. Its balanced complexity and capacity for feature extraction allowed it to achieve high accuracy while maintaining generalization, making it the most effective model for the given task.

While the analysis successfully demonstrated improvements in classification accuracy, there were some limitations. The Fashion MNIST dataset, despite offering a more challenging task than digit recognition, is still limited in terms of image variety and resolution. This may restrict the models' ability to generalize to more diverse fashion datasets. Additionally, the analysis primarily focused on adjusting model architecture, but it did not fully explore hyperparameter tuning or alternative optimization techniques, which could have further improved performance. The CNN models used required significant computational resources, making it challenging to test a broader range of hyperparameter settings and network configurations.

Future work could address these limitations by experimenting with more advanced regularization techniques, such as dropout tuning or L2 regularization, to further mitigate overfitting. Additionally, employing hyperparameter optimization techniques like grid search or Bayesian optimization could help fine-tune learning rates, batch sizes, and other parameters for even better results. Expanding the dataset with more diverse images or using data augmentation

techniques could also improve model generalization. Exploring transfer learning with pre-trained models on larger fashion datasets could offer further accuracy gains while reducing the training time required.

References

fashion_mnist | TensorFlow Datasets. (2024). TensorFlow. https://www.tensorflow.org/datasets/catalog/fashion_mnist

How does Epoch affect Accuracy in Deep Learning Model? (2023, July 31). GeeksforGeeks. https://www.geeksforgeeks.org/how-does-epoch-affect-accuracy-in-deep-learning-model/

Keras. (2019). *Home - Keras Documentation*. Keras.io. https://keras.io/

Keras Team. (n.d.). *Keras documentation: Fashion MNIST dataset, an alternative to MNIST*. Keras.io. https://keras.io/api/datasets/fashion_mnist/

Knode, S. (2024). *Model Fashion MNIST Database.ipynb*. Google Colab; Google.com. https://colab.research.google.com/drive/1lTdNyCGyTW5V5bK4sfU70y6_LYMZMrMJ

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. *ArXiv:1708.07747 [Cs, Stat]*. https://arxiv.org/abs/1708.07747

Zalando Research, & Crawford, C. (2017). *Fashion MNIST*. Www.kaggle.com. https://www.kaggle.com/datasets/zalando-research/fashionmnist

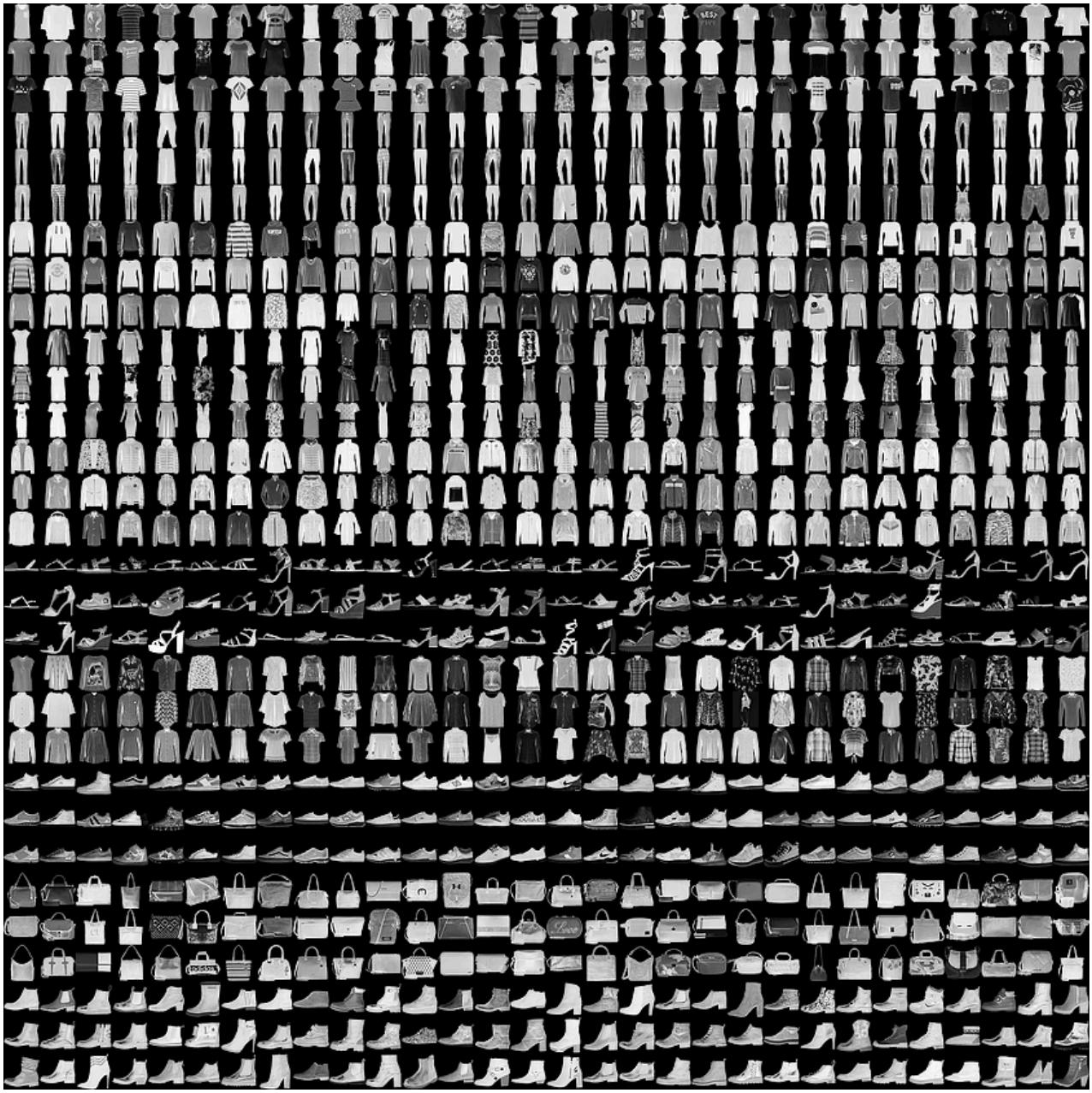
Zalandoresearch. (2017). *GitHub - zalandoresearch/fashion-mnist: A MNIST-like fashion product database. Benchmark*. GitHub. https://github.com/zalandoresearch/fashion-mnist?tab=readme-ov-file

Appendix A

All figures and visualizations mentioned in the report can be seen below. The python script used to generate the model and the visualizations is attached as a separate file.

Figure 1

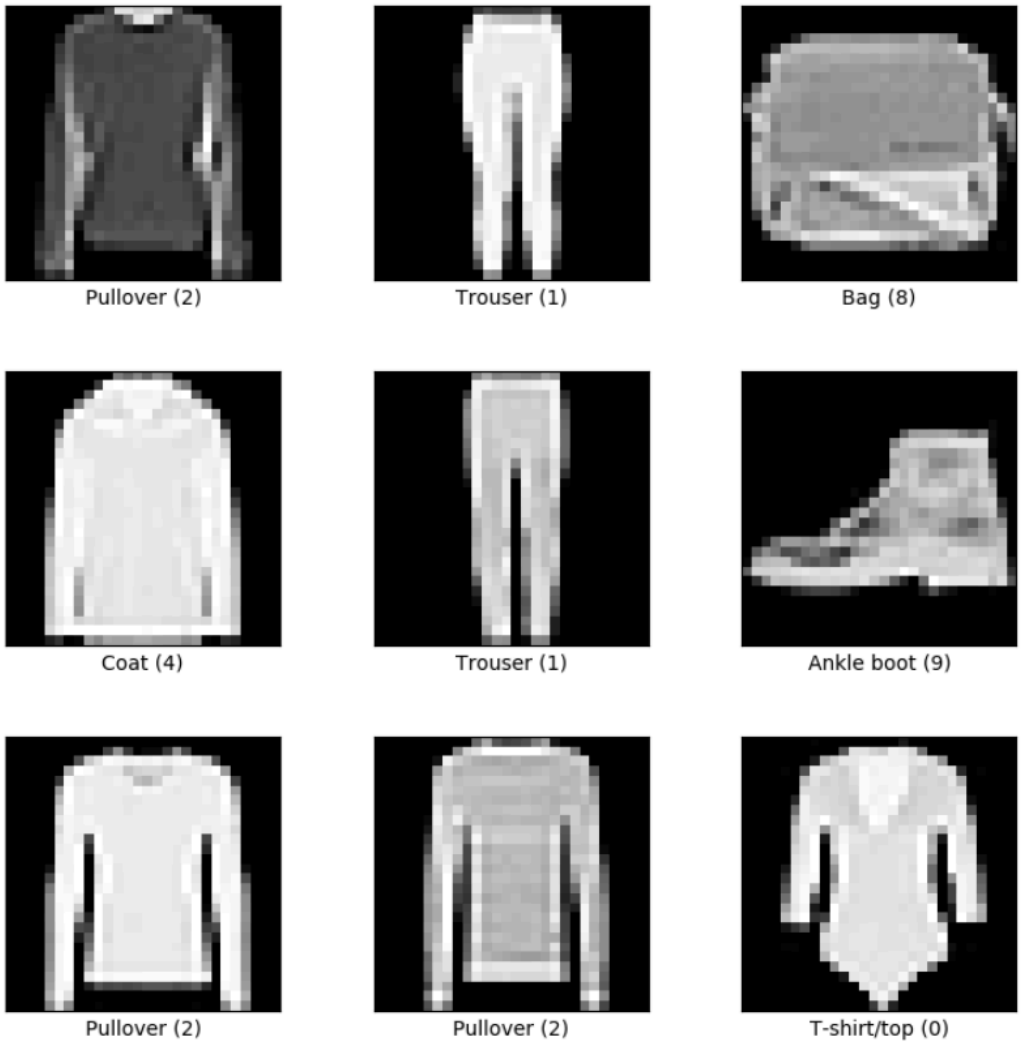
Figure of the dataset showing all 10 dataset variables in order: t-shirts/top, trousers, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. (Keras Team, n.d.)



(Zalandoresearch, 2017)

Figure 2

Figure of the dataset showing no missing values and summary statistics with no missing and high skew values.



(Fashion_mnist | TensorFlow Datasets, 2024)

Table 2

Table of the model summary outlining key details and differences for the baseline model and models 1-6.

Model Name	Convolutional Layers	Filter Number	Neurons	Kernel Size	Pooling Layers	Training Epochs	Training Batch Size	Dropout Rate	Comments
Baseline	2 (Conv1D)	236, 128	10	2	2 (MaxPooling1D)	10	64	0.2	Basic structure using Conv1D layers. Focuses on simple feature extraction from image data. Serves as the starting point.
Model 1	3 (Conv1D)	236, 128, 64	10	2	3 (MaxPooling1D)	10	64	0.2	Added an extra Conv1D layer with 64 filters for deeper feature extraction. Improved feature learning compared to the baseline.
Model 2	3 (Conv1D)	256, 128, 64	10	2	3 (MaxPooling1D)	10	64	0.2	Increased the number of filters in each Conv1D layer to enhance feature extraction. Allows the model to capture more nuanced patterns.
Model 3	3 (Conv1D)	256, 128, 64	10	3	3 (MaxPooling1D)	10	64	0.3	Increased the kernel size to 3 for broader spatial pattern recognition. Adjusted dropout rate to 0.3 to prevent overfitting.
Model 4	3 (Conv2D)	32, 64, 64	64	3x3	3 (MaxPooling2D)	10	64	0.2	Transitioned to Conv2D layers for better handling of spatial data in images. Increased depth with more dense layer neurons.
Model 5	4 (Conv2D)	32, 64, 128	128	3x3	3 (MaxPooling2D)	10	64	0.2	Added a fourth Conv2D layer with 128 filters for more detailed feature extraction. Increased dense layer capacity for better learning.
Model 6	4 (Conv2D)	32, 64, 128	128	3x3	3 (MaxPooling2D)	25	64	0.2	Trained for more epochs (25 vs. 10) to refine the model's understanding of the data. Aimed to improve performance but risked overfitting.

Figure 3

Figure of the confusion matrix for Model 4.

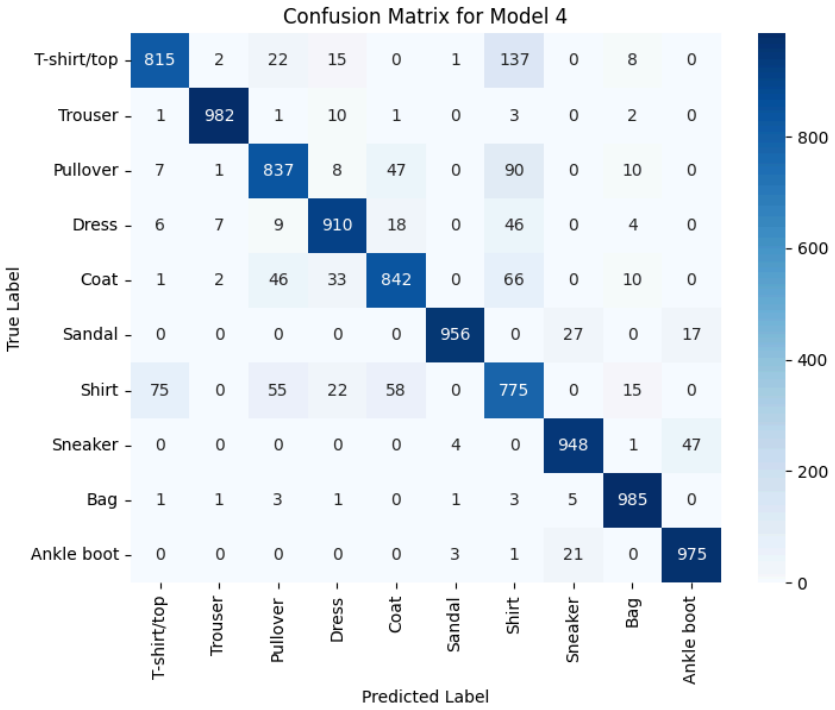


Figure 4

Figure of the confusion matrix for Model 5.

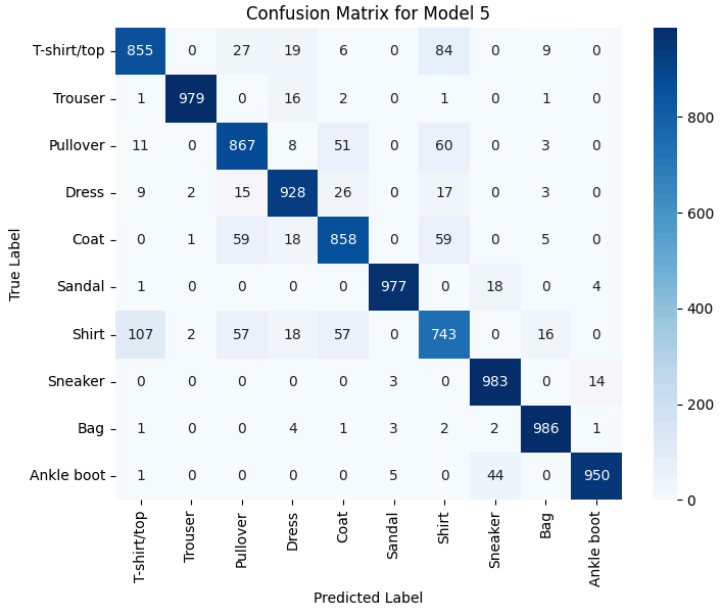


Figure 5

Figure of the confusion matrix for Model 6.

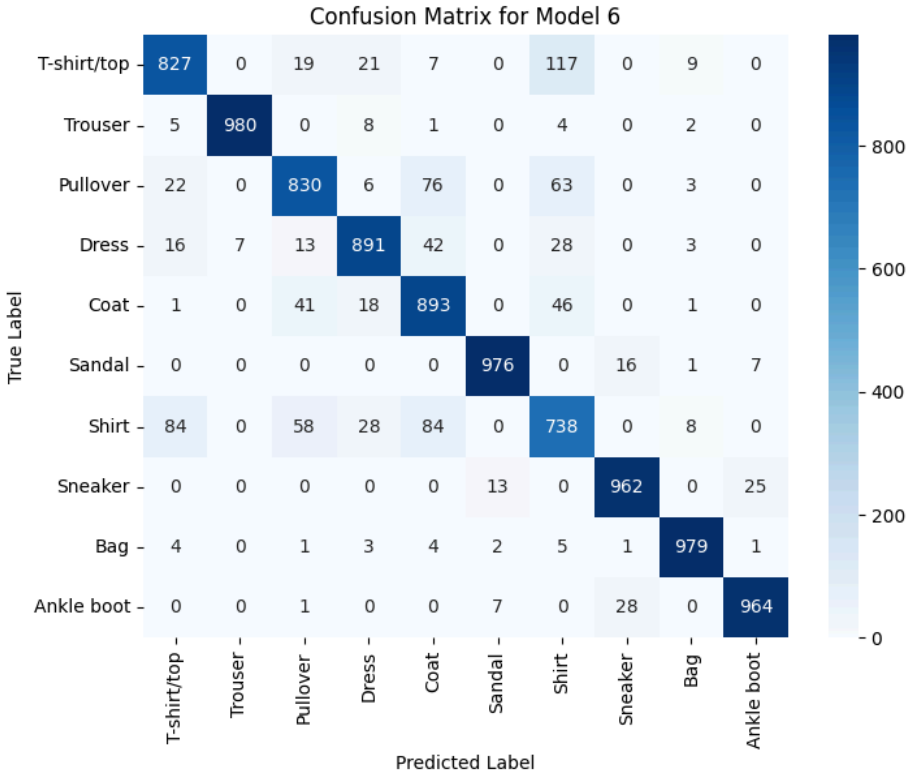


Figure 6

Figure of the classification report for Model 4.

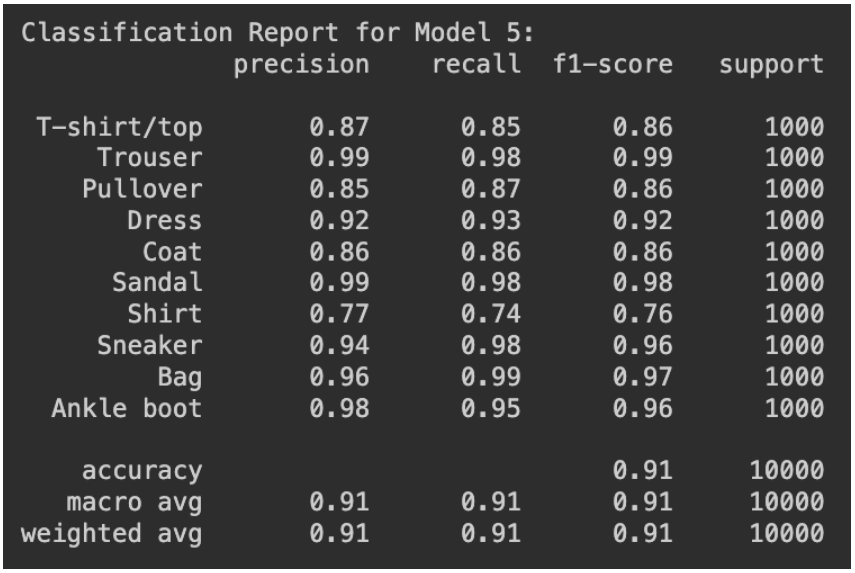
```
Classification Report for Model 4:
              precision    recall  f1-score   support

 T-shirt/top       0.90      0.81      0.86      1000
     Trouser       0.99      0.98      0.98      1000
    Pullover       0.86      0.84      0.85      1000
       Dress       0.91      0.91      0.91      1000
        Coat       0.87      0.84      0.86      1000
      Sandal       0.99      0.96      0.97      1000
       Shirt       0.69      0.78      0.73      1000
     Sneaker       0.95      0.95      0.95      1000
         Bag       0.95      0.98      0.97      1000
  Ankle boot       0.94      0.97      0.96      1000

    accuracy                           0.90     10000
   macro avg       0.90      0.90      0.90     10000
weighted avg       0.90      0.90      0.90     10000
```

Figure 7

Figure of the classification report for Model 5.

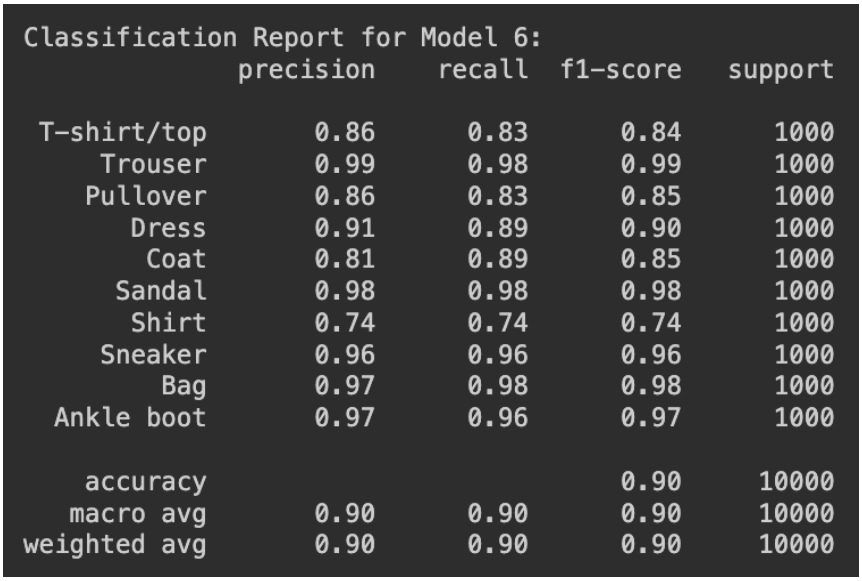


Classification Report for Model 5:

	precision	recall	f1-score	support
T-shirt/top	0.87	0.85	0.86	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.85	0.87	0.86	1000
Dress	0.92	0.93	0.92	1000
Coat	0.86	0.86	0.86	1000
Sandal	0.99	0.98	0.98	1000
Shirt	0.77	0.74	0.76	1000
Sneaker	0.94	0.98	0.96	1000
Bag	0.96	0.99	0.97	1000
Ankle boot	0.98	0.95	0.96	1000
accuracy			0.91	10000
macro avg	0.91	0.91	0.91	10000
weighted avg	0.91	0.91	0.91	10000

Figure 8

Figure of the classification report for Model 6.



Classification Report for Model 6:

	precision	recall	f1-score	support
T-shirt/top	0.86	0.83	0.84	1000
Trouser	0.99	0.98	0.99	1000
Pullover	0.86	0.83	0.85	1000
Dress	0.91	0.89	0.90	1000
Coat	0.81	0.89	0.85	1000
Sandal	0.98	0.98	0.98	1000
Shirt	0.74	0.74	0.74	1000
Sneaker	0.96	0.96	0.96	1000
Bag	0.97	0.98	0.98	1000
Ankle boot	0.97	0.96	0.97	1000
accuracy			0.90	10000
macro avg	0.90	0.90	0.90	10000
weighted avg	0.90	0.90	0.90	10000

Figure 9

Figure of the ROC curve for models 5-8.

Figure 10

Figure of the ROC curve for models 9-12.

Figure 11

Figure of the ROC curve for models 13-16.

Figure 12

Figures of the Cutoff curves for models 13-16. Top left: Ensemble Average; top right: Ensemble Maximum; bottom left: Voting Average; and bottom right: Voting Proportion.

Table 2

Table of the model summary outlining key details and differences for Ensemble II, Ensemble I, and SVM models.