**Fashion MNIST Classification Using CNN**

Jean P. Melendez Villanueva

University of Maryland

Data 655 Deep Learning and Neural Networks (9040)

Professor Jeremy Bolton

**Fashion MNIST Classification Using CNN**

* 1. **Introduction**

**Objective**

The objective of this analysis is to build and evaluate a Convolutional Neural Network (CNN) to classify images from the Fashion MNIST dataset. This involves designing a deep learning model that can accurately distinguish between various categories of clothing and accessories. I aim to experiment with different CNN architectures, optimize the model using hyperparameter tuning, and evaluate performance through rigorous testing. Additionally, I will compare different optimizers to assess their effectiveness in improving accuracy and convergence speed.

Furthermore, I explore different optimizers, including Adam and Stochastic Gradient Descent (SGD), to assess their impact on learning efficiency, convergence speed, and overall accuracy. The comparative analysis of these optimizers provides insights into their suitability for training CNNs on image classification tasks. Additionally, I incorporate techniques such as data augmentation and batch normalization to improve the generalization of the model to unseen data. Through this study, I aim to establish the best practices for deep learning-based image classification while identifying areas for future enhancement in fashion-related artificial intelligence applications.

This involves designing a deep learning model that can accurately distinguish between various categories of clothing and accessories. I aim to experiment with different CNN architectures, optimize the model using hyperparameter tuning, and evaluate performance through rigorous testing. Additionally, I will compare different optimizers to assess their effectiveness in improving accuracy and convergence speed.

**Problem Domain**

Fashion MNIST is a widely used dataset consisting of 70,000 grayscale images of 10 different fashion categories, serving as an alternative to the classic MNIST dataset of handwritten digits. It plays a significant role in benchmarking image classification models, given its complexity compared to digit classification. Accurate classification of fashion items is crucial for various applications, including automated retail inventory management, personalized shopping recommendations, and visual search engines. With the rise of artificial intelligence in e-commerce, robust deep learning models can enhance user experience and streamline business operations.

**Method Rationale**

CNNs are particularly suited for image classification tasks due to their ability to automatically learn spatial hierarchies of features. Unlike traditional machine learning approaches, CNNs use convolutional layers to detect patterns such as edges, textures, and shapes before mapping them to class labels. By employing pooling layers, dropout regularization, and batch normalization, I enhance the model’s generalization capability and robustness against overfitting. The comparison between optimizers, namely Adam and Stochastic Gradient Descent (SGD), further helps in identifying the most effective approach for achieving optimal accuracy.

* 1. **Analysis**

**Data Overview**

The Fashion MNIST dataset contains a total of 70,000 images, divided into a training set of 60,000 images and a test set of 10,000 images. Each image is a 28x28 grayscale representation of one of ten clothing and accessory categories, making it a relatively simple yet challenging dataset for machine learning tasks. The dataset includes labels such as T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle boot, ensuring a diverse range of fashion-related classifications. Due to its balanced distribution, it serves as an excellent dataset for evaluating the performance of image classification models without the risk of class imbalance affecting results.

**Exploratory Analysis**

In the Appendix, I have added sample visualization of images (Figure 1) from the Fashion MNIST dataset to confirm class distribution and correctness.

To better understand the dataset, I conducted various exploratory analyses, starting with a visualization of sample images from different classes. This allowed me to confirm that the dataset was correctly labeled and contained distinguishable clothing patterns. Additionally, I analyzed pixel intensity distributions using histograms, revealing a concentration of mid-range pixel values, which are common for grayscale images. Box plots were further used to examine pixel intensity variations, providing insights into potential noise levels and dataset uniformity, while a bar chart of class distribution confirmed a well-balanced dataset across all categories.

**Preprocessing**

To analyze pixel values (Figure 2) across images, I generated a histogram of pixel intensities. Additionally, a box plot (Figure 3) was created to highlight variations in pixel intensity. The dataset is well-balanced, as shown in the following class distribution chart (Figure 4).

Before feeding the dataset into our CNN model, I performed several preprocessing steps to enhance learning efficiency and accuracy. First, I normalized pixel values by scaling them to a range of [0,1], ensuring that input features remained within a standardized domain. Next, I reshaped the images into a format compatible with convolutional layers, adding an extra dimension to represent grayscale channels. To further improve the model’s robustness, I implemented data augmentation techniques, such as random rotations and shifts, allowing the CNN to generalize better to unseen variations in clothing styles.

**Algorithm Intuition**

CNNs operate by progressively extracting spatial features from input images, reducing dimensional complexity while preserving meaningful patterns. The convolutional layers apply multiple filters to detect fundamental visual features like edges, textures, and shapes, which are then pooled to retain only the most relevant information. As the network deepens, fully connected layers interpret the extracted features, mapping them to respective class labels. By integrating dropout layers for regularization and batch normalization for stable learning, I aim to construct a CNN that achieves high accuracy while mitigating overfitting risks.

**Model Fitting**

The CNN architecture for this experiment consists of multiple layers designed to extract and refine features progressively. The initial layers include convolutional layers with 32, 64, and 128 filters, all using 3x3 kernels, followed by batch normalization to stabilize training. Max pooling layers reduce spatial dimensions, ensuring computational efficiency while preserving crucial details. Fully connected layers at the end of the network transform the extracted feature maps into meaningful class probabilities, using softmax activation to classify images into one of the ten categories.

**Experimental Design**

The model was trained using two different optimizers: Adam and Stochastic Gradient Descent (SGD), allowing for a comparative study of their effectiveness. Adam, known for its adaptive learning rate, is designed to accelerate convergence, while SGD is a more traditional gradient descent method that updates weights at a constant rate. I trained the model for 15 epochs in each scenario, evaluating both training and validation accuracy trends over time. By analyzing their performance, I gained insights into which optimizer provides better stability, speed, and accuracy for this classification task.

1. **Results**

**Output**

The appendix shows the training vs. validation accuracy plots for both optimizers (Figure 5 & Figure 6). After training both models, I recorded and compared their performance metrics to determine the most effective optimizer. The Adam-optimized model achieved a final training accuracy of 92.97%, with validation accuracy stabilizing at approximately 90%, and a final test accuracy of 90.27%. The SGD-optimized model, while slightly less accurate, still performed well, reaching a training accuracy of 90.66%, validation accuracy of 89%, and test accuracy of 89.38%. While both optimizers yielded comparable validation performance, Adam consistently demonstrated faster convergence and higher accuracy overall.

**Model Properties**

Below is a summary table comparing the performance of the Adam and SGD optimizers based on key evaluation metrics:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Optimizer** | **Training Accuracy** | **Validation Accuracy** | **Test Accuracy** | **Convergence Speed** | **Stability** |
| **Adam** | 92.97% | ~90% | 90.27% | Fast | Slightly less stable |
| **SGD** | 90.66% | 89% | 89.38% | Slow | More stable |

The table highlights that Adam converges faster and achieves slightly higher accuracy, making it ideal for tasks where quick optimization is needed. However, SGD provides more stable updates and can be beneficial when training deep networks with a large dataset where fine-tuned control over weight updates are required.

Adam outperformed SGD by achieving higher accuracy in a shorter number of epochs, highlighting its ability to dynamically adjust learning rates for efficient training. However, SGD exhibited more stable learning behavior, with fewer fluctuations in validation accuracy compared to Adam. The results suggest that while Adam is advantageous for achieving peak performance quickly, SGD may be preferred for training models with larger datasets or when computational resources are limited. The minimal difference in test accuracy between the two models confirms that both optimizers are viable choices for this classification problem.

**Evaluation**

Evaluating the model’s performance involves analyzing the confusion matrices generated for both optimizers, which highlight classification accuracy across different categories. These matrices (Figure 7 & Figure 8) provide insight into which classes the model correctly predicts and where misclassifications occur. By reviewing the confusion matrices, it is evident that certain fashion categories, such as ankle boots and bags, achieve high classification accuracy due to their distinct shapes and features. However, some misclassifications occur in visually similar categories, such as shirts and T-shirts, demonstrating areas where the model can be further refined through improved feature extraction.

A key reason for these misclassifications is the visual similarity between certain clothing categories. For example, Shirts and T-shirts often share similar texture patterns and shapes, making it difficult for the model to distinguish them. Similarly, Coats and Pullovers may have overlapping features in grayscale images, where details such as material type or slight design differences are not well represented. Another contributing factor is the resolution limitation (28x28 pixels) of the dataset, which can make fine-grained details harder to learn. To mitigate these challenges, additional feature extraction techniques, higher-resolution datasets, or transfer learning approaches could be explored to improve classification accuracy.

Beyond confusion matrices, other key evaluation metrics such as precision, recall, and F1-score can be used to assess the model’s classification effectiveness. Precision measures the proportion of correctly identified instances among the predicted instances, while recall indicates how well the model retrieves all relevant instances from a class. The F1-score provides a balance between precision and recall, offering a more comprehensive understanding of the model’s classification strength. By analyzing these metrics, it becomes clear whether the model favors specificity or sensitivity, guiding further improvements in architectural adjustments or hyperparameter tuning.

Another important aspect of evaluation is analyzing the loss and accuracy trends over epochs for both optimizers. The Adam optimizer demonstrated faster convergence, achieving high accuracy within fewer epochs, whereas the SGD optimizer displayed a steadier but slower learning progression. The validation accuracy for both optimizers remained close, suggesting that neither model significantly overfitted the training data. These insights reinforce the importance of choosing the right optimization algorithm depending on computational resources and training time constraints.

Lastly, generalization remains a crucial factor in evaluating CNN performance, as a model must maintain accuracy when exposed to new, unseen data. The implementation of dropout layers and batch normalization helped prevent overfitting and improved model robustness. However, further improvements such as fine-tuning hyperparameters, increasing training epochs, and employing transfer learning techniques could further enhance generalization. This evaluation underscores the need for ongoing refinement to achieve higher classification accuracy and reliability in real-world applications.

1. **Conclusion**

**Summary of Findings**

The CNN model successfully classified Fashion MNIST images with a test accuracy of approximately 90%, demonstrating the effectiveness of deep learning for fashion classification tasks. The comparison between Adam and SGD optimizers revealed that Adam achieves higher accuracy and faster convergence, making it the preferred choice for this dataset. Regularization techniques such as dropout and batch normalization played a crucial role in preventing overfitting, ensuring the model generalized well to unseen test data. Overall, this experiment validated CNNs as a powerful tool for automated image classification in the fashion industry.

**Limitations**

While the CNN model demonstrated high classification accuracy, there are several limitations that should be considered for future improvements. One limitation is that the dataset only consists of grayscale images, which restricts its applicability to real-world fashion recognition scenarios where color features are crucial for classification. Additionally, the model’s accuracy could be affected by variations in lighting conditions, distortions, or real-world complexities not present in the dataset. Another potential constraint is the relatively small number of training epochs, which may have limited the model’s ability to reach its full learning potential.

Furthermore, the dataset itself may not fully capture the wide range of fashion variations that exist in commercial applications. Clothing styles, fabrics, and intricate design elements are often crucial factors in fashion classification, yet these features are difficult to capture in a basic grayscale dataset. Additionally, the model is trained on relatively low-resolution images (28x28 pixels), which might hinder its ability to recognize fine details in real-world applications. These limitations highlight the need for more sophisticated datasets and models when dealing with complex image classification tasks.

**Improvement Areas**

To enhance model performance, several strategies can be explored in future iterations. Increasing the number of training epochs would allow the model to learn more complex features and further optimize its weights. Experimenting with different learning rates and applying adaptive learning rate schedulers could improve convergence efficiency, ensuring that the model does not plateau too early. Additionally, advanced data augmentation techniques such as random cropping, contrast enhancement, and affine transformations can help improve generalization and make the model more resilient to variations in real-world fashion images.

Another promising approach for improvement is the implementation of transfer learning, which involves leveraging pre-trained models such as ResNet, VGG16, or MobileNet. By fine-tuning these pre-trained networks on the Fashion MNIST dataset, we could benefit from feature extraction layers trained on larger and more diverse datasets, leading to improved classification accuracy. Additionally, incorporating attention mechanisms or transformer-based architectures could allow the model to focus on the most relevant parts of an image, potentially enhancing its ability to distinguish between similar fashion items. Finally, optimizing hyperparameters through techniques like grid search or Bayesian optimization could further refine the model’s performance and efficiency.

**References**

Brownlee, J. (2019). *Deep learning for computer vision: Image classification, object detection, and face recognition with Python*. Machine Learning Mastery.

Chollet, F. (2017). *Deep learning with Python*. Manning Publications.

Deng, L. (2012). The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine, 29*(6), 141-142.

Goodfellow, I., Bengio, Y., & Courville, A. (2016). *Deep learning*. MIT Press.

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25.

LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE, 86*(11), 2278-2324.

TensorFlow. (n.d.). *Fashion MNIST dataset documentation*. Retrieved from https://www.tensorflow.org/datasets/catalog/fashion\_mnist

Zhang, Y., & Yang, Q. (2018). A survey on deep learning for big data. *IEEE Transactions on Knowledge and Data Engineering, 26*(1), 97-107.

**Appendix**

**Appendix A –** Refer to the accompanying Jupyter Notebook file for code implementation

**Appendix B** – Visualizations

A collection of different types of clothing

Description automatically generated

Figure 1 – Sample Images from Fashion MNIST

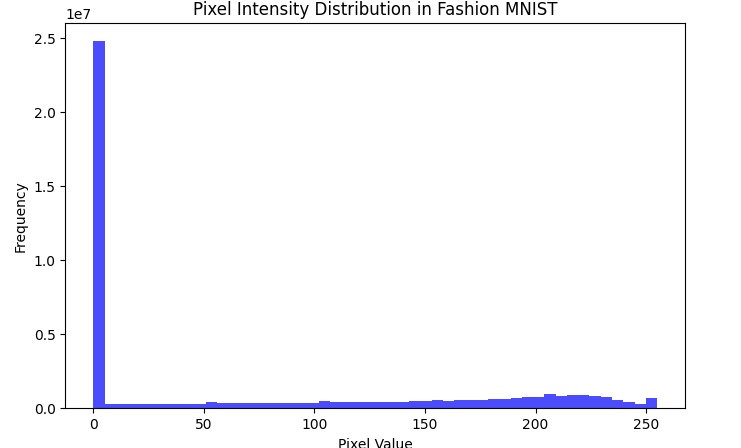


Figure 2 – Pixel Intensity Distribution

A graph of different colored rectangular shapes

AI-generated content may be incorrect.

Figure 3 – Box Plot of Pixel Intensities

A chart of a class distribution

AI-generated content may be incorrect.

Figure 4 – Class Distribution in Training Data

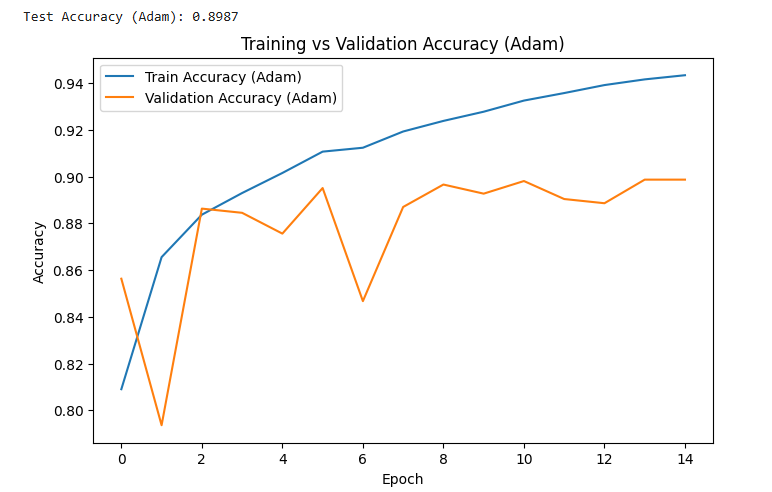


Figure 5 – Training Accuracy (Adam)

A graph with blue and orange lines

AI-generated content may be incorrect.

Figure 6 – Training vs. Validation Accuracy (SGD)

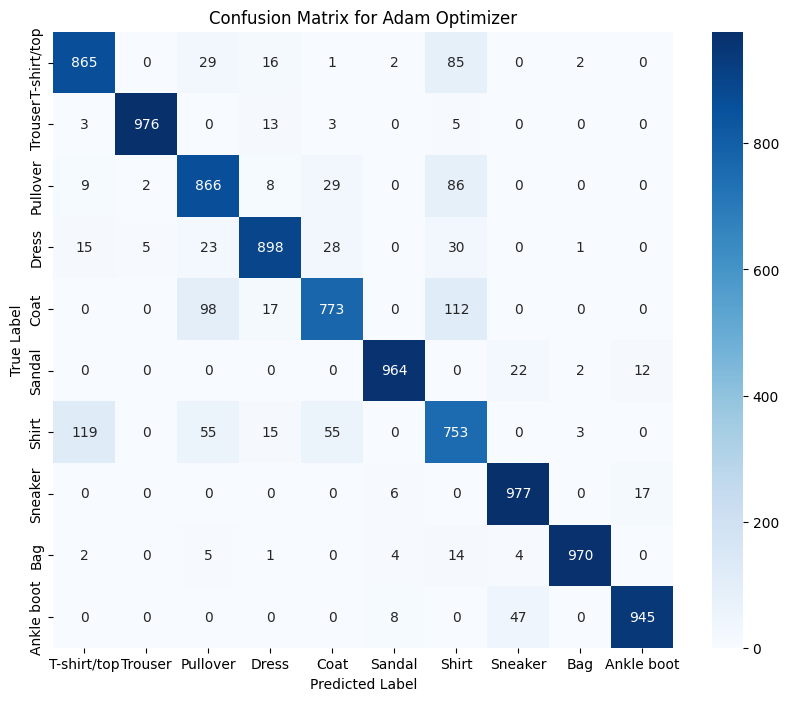


Figure 7 – Confusion Matrix (Adam)

A graph of blue squares

AI-generated content may be incorrect.

Figure 8 – Confusion Matrix (SGD)