**Neural Networks in Python**

Jean P. Melendez Villanueva

University of Maryland

Data 655 Deep Learning and Neural Networks (9040)

Professor Jeremy Bolton

**Neural Networks in Python**

* 1. **Introduction**

**Objective**

The primary objective of this analysis is to construct, train, and evaluate an artificial neural network (ANN) capable of classifying images from the Fashion MNIST dataset into one of 10 clothing categories. The analysis focuses on supervised learning methods to achieve high classification accuracy. Specifically, the analysis seeks to answer the following questions:

* Can an ANN effectively learn patterns in Fashion MNIST images to classify them accurately?
* What are the performance characteristics of the ANN, and how can its architecture influence accuracy and generalization?
* How does the ANN perform on unseen test data, and what insights can be gained from its evaluation metrics?

This study also aims to demonstrate the practical applicability of ANNs in solving classification tasks and explores avenues for future improvement through advanced techniques.

**Problem Domain**

The Fashion MNIST dataset is a widely used benchmark in machine learning, comprising grayscale images representing 10 clothing categories such as T-shirts, trousers, pullovers, dresses, and shoes. Each image is a low-resolution (28x28 pixels) representation of a single clothing item. With 70,000 labeled images, divided into 60,000 training samples and 10,000 test samples, the dataset offers an ideal platform for experimenting with classification algorithms.

This dataset finds practical relevance in domains like e-commerce and retail, where accurate classification of clothing items can enhance inventory management, personalization, and search recommendations. Compared to simpler datasets like MNIST digits, Fashion MNIST poses a moderate challenge due to its visually similar categories, such as T-shirts and shirts or sneakers and ankle boots. These complexities make it a valuable resource for benchmarking classification models.

**Method Rationale**

ANNs are highly suitable for image classification tasks because of their ability to model non-linear relationships and learn intricate patterns in data. Unlike traditional algorithms, ANNs excel in learning hierarchical features directly from raw inputs, such as pixel values. The foundation of this approach lies in several key principles:

1. **Feedforward Architecture**: In this architecture, information flows in a unidirectional manner across interconnected layers of neurons—input, hidden, and output.
2. **Backpropagation Algorithm**: This method computes the gradients of the loss function and iteratively adjusts weights to minimize errors during training.
3. **Optimization Techniques**: The Adam optimizer is used to achieve efficient convergence by combining adaptive learning rates with momentum.

The Fashion MNIST dataset's moderate size and complexity make it well-suited for ANNs with dense layers. ReLU activation functions are employed to introduce non-linearity, while dropout layers mitigate overfitting by randomly deactivating neurons during training. Resources such as TensorFlow's documentation and tutorials (TensorFlow, n.d.) provided the foundational knowledge and practical guidance necessary for implementing this model.

* 1. **Analysis**

**Data Overview**

The Fashion MNIST dataset contains 70,000 images, each labeled with one of 10 categories. The images are 28x28 grayscale, making each input feature vector 784-dimensional after flattening. The labels correspond to the following categories: T-shirt/top, Trouser, Pullover, Dress, Coat, Sandal, Shirt, Sneaker, Bag, and Ankle Boot (Xiao et al., 2017). The dataset structure is as follows:

* **Training Set**: 60,000 images used for learning patterns and optimizing weights.
* **Test Set**: 10,000 images for evaluating the model’s performance on unseen data.

The labels are evenly distributed across categories, ensuring a balanced dataset. Each image’s pixel values range from 0 to 255 and are normalized to [0, 1] to enhance numerical stability during training. A one-hot encoding scheme was applied to labels, converting each into a 10-dimensional binary vector suitable for multi-class classification.

**Exploratory Analysis**

Initial exploratory analysis of the Fashion MNIST dataset involved several steps aimed at understanding the data and ensuring its suitability for training an artificial neural network (ANN). The first step was visualizing random samples from the dataset to confirm the diversity of labels and the quality of the images. Sample images from each category were plotted alongside their respective labels, providing an initial insight into the data’s structure. This visualization also highlighted potential challenges, such as the visual similarity between certain categories like T-shirts and shirts, which could lead to classification errors.

Next, histograms of pixel intensity distribution were created to examine the range and distribution of grayscale values. The analysis revealed that the pixel values span from 0 to 255, necessitating normalization to a range of [0, 1] for improved numerical stability during training. This step is critical in ensuring consistent weight updates and convergence during the optimization process.

Finally, category balance was evaluated by plotting the distribution of labels across all 10 classes. The results showed an even representation of categories, confirming that the dataset is balanced. This balance is essential to prevent bias during training and ensure that the model does not favor any specific class. Together, these exploratory steps provided a strong foundation for preprocessing and model development (TensorFlow, n.d.-a).

**Preprocessing**

Preprocessing steps were crucial in preparing the Fashion MNIST dataset for training the artificial neural network (ANN). The first step was normalization, where pixel values were scaled from their original range of 0 to 255 to a normalized range of [0, 1]. This transformation improved numerical stability and ensured faster convergence during training by preventing large gradient updates.

Next, the images were flattened from their original 28x28 matrix format into 784-dimensional vectors. This restructuring was necessary to make the data compatible with dense layers in the ANN, which process input features as one-dimensional arrays. Each vector served as a direct representation of the image's pixel intensities.

To prepare the labels for multi-class classification, one-hot encoding was applied. This step converted each categorical label into a binary vector with a length of 10, corresponding to the 10 clothing categories in the dataset. For instance, a label of 3 (Pullover) was encoded as [0, 0, 0, 1, 0, 0, 0, 0, 0, 0], where the fourth position indicates the category.

Finally, a validation split of 20% was applied to the training data. This partitioning ensured that model performance could be monitored on unseen data during training, helping to identify overfitting or underfitting early in the process. By splitting the data, the model's generalization ability was tested iteratively, enabling adjustments to hyperparameters and architecture as needed.

**Algorithm Intuition**

The ANN architecture consists of an input layer with 784 neurons corresponding to the flattened pixel values. Two hidden layers with 128 and 64 neurons, respectively, were used, each employing ReLU activation for non-linear transformations. Dropout layers with a rate of 0.2 were introduced to prevent overfitting. The output layer contains 10 neurons with SoftMax activation to output class probabilities. The model was compiled with the categorical cross entropy loss function and optimized using the Adam algorithm with a learning rate of 0.001 (TensorFlow, n.d.-b).

**Model Fitting**

The model was trained for 15 epochs with a batch size of 32. Training was conducted on 80% of the dataset, with the remaining 20% reserved for validation. The training process involved iteratively adjusting weights to minimize the categorical cross entropy loss using backpropagation. To improve performance, experiments were conducted with different dropout rates, batch sizes, and learning rates. The final configuration achieved a balance between training speed and accuracy. Video tutorials on TensorFlow’s YouTube channel provided additional guidance on training and optimizing the ANN (TensorFlow, 2023a, 2023b, 2023c).

1. **Results**

**Output**

The trained model achieved a test accuracy of 88%, indicating strong performance for a simple dense ANN architecture. Training accuracy reached 92.5%, while validation accuracy stabilized at 89.8%, showing minimal overfitting. These results demonstrate the effectiveness of the model in capturing patterns from the dataset and generalizing unseen data. The outputs from the final layer—probabilities for each class—were consistent with the target categories, further confirming the model’s predictive capability.

**Model Properties**

The final ANN model had approximately 120,000 parameters distributed across its layers. The input layer, with 784 neurons, directly mapped the pixel values from each image. Two hidden layers of 128 and 64 neurons provided hierarchical feature extraction, while dropout layers with a rate of 0.2 acted as regularization mechanisms to prevent overfitting. The softmax activation in the output layer ensured probabilistic interpretation of predictions. The simplicity of the dense architecture made training computationally efficient, averaging 2 minutes per epoch on Google Colab. Despite this simplicity, the model captured key relationships in the data and balanced computational efficiency with performance.

**Evaluation**

To evaluate the model, a variety of performance metrics and visualizations were utilized to provide both quantitative and qualitative insights into its effectiveness.

The confusion matrix played a central role in identifying areas of misclassification. This visualization highlighted specific challenges faced by the model, such as the frequent confusion between visually similar categories like T-shirts and shirts. However, other categories, such as sneakers and ankle boots, were classified with high accuracy, demonstrating the model’s capability to distinguish between more distinct features. By analyzing the confusion matrix, granular insights were gained into the strengths and weaknesses of the model’s predictions.

Metrics such as precision, recall, and F1-score were instrumental in evaluating the model’s overall performance. An overall precision of 0.91 indicated that the model was highly reliable in assigning correct labels to predicted classes. The recall, measured at 0.89, reflected the model’s ability to identify all relevant instances for each category. The F1-score, which balances precision and recall, was 0.90, underscoring the robustness and accuracy of the ANN across all categories.

Analysis of loss and accuracy trends provided further validation of the model’s effectiveness. Plots of training and validation accuracy over epochs showed steady improvements, with minimal divergence between the two curves. This indicated that the model was learning effectively without significant overfitting. Similarly, the convergence of training and validation loss plots confirmed stable optimization during training, further reinforcing the model’s reliability.

Finally, visualizations such as heatmaps of the confusion matrix and sample predictions added an intuitive layer to the evaluation. These tools not only showcased the model’s strengths but also revealed specific misclassification patterns, such as the challenges in distinguishing T-shirts from shirts. These visual tools provided actionable insights for refining the model in future iterations.

TensorFlow’s built-in functions, including tf.math.confusion\_matrix, were instrumental in calculating and visualizing these metrics. These resources further validated the model’s robustness and effectiveness in handling the Fashion MNIST dataset (TensorFlow, n.d.-c).

1. **Conclusion**

**Summary of Findings**

The ANN successfully classified Fashion MNIST images with an 88% test accuracy, showcasing its effectiveness in addressing moderate complexity image classification tasks. This performance highlights the model's ability to generalize patterns from training data to unseen test data. The model achieved robust learning, as evidenced by a close alignment between training accuracy (92.5%) and validation accuracy (89.8%), with minimal overfitting. The results demonstrate the potential of ANNs in efficiently handling real-world classification problems that require both accuracy and computational feasibility. Furthermore, the analysis validates the practical applicability of ANNs for classification tasks within e-commerce and similar domains.

**Limitations**

Despite the success of the ANN, there are notable limitations. Architecture constraints were observed, as the dense layers utilized in the model lack the ability to extract spatial relationships within the image data. This limitation hindered the model’s ability to differentiate visually similar categories, such as T-shirts and shirts, which require a more nuanced understanding of spatial features. Additionally, the dataset’s grayscale format simplifies the problem and does not fully emulate real-world scenarios involving color images. Another limitation was the absence of data augmentation, which could have enhanced the model’s ability to generalize by simulating variations such as rotations, flips, and other transformations that occur in practical settings.

**Improvement Areas**

Several strategies can address these limitations to further improve the model’s performance. Incorporating convolutional neural networks (CNNs) would allow the model to better capture spatial hierarchies within the image data, significantly improving its classification capabilities, particularly for visually similar categories. Data augmentation should be employed to increase dataset diversity artificially, introducing variations that would help the model generalize more effectively to unseen data. Additionally, transfer learning leveraging pre-trained models like ResNet could boost accuracy by applying knowledge from larger and more diverse datasets.

Advanced hyperparameter tuning, including grid search or random search, could optimize factors such as learning rates, batch sizes, and layer configurations. Lastly, expanding the analysis to include colored datasets would better represent real-world applications, providing insights into how the model handles more complex and realistic data.

**References**

TensorFlow Documentation: https://www.tensorflow.org/api\_docs/python

TensorFlow Dataset Catalog: https://www.tensorflow.org/datasets/catalog/fashion\_mnist

TensorFlow Tutorials: https://www.tensorflow.org/tutorials/keras/classification

TensorFlow Dataset Keras Example: https://www.tensorflow.org/datasets/keras\_example

TensorFlow Beginner Quickstart: https://www.tensorflow.org/tutorials/quickstart/beginner

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

TensorFlow (2023a). TensorFlow for Beginners: Video Tutorials. Retrieved from https://www.youtube.com/watch?v=\_VTtrSDHPwU&list=PLQY2H8rRoyvyK5aEDAI3wUUqC\_F0oEroL&index=3

TensorFlow (2023b). Building Neural Networks in TensorFlow. Retrieved from https://www.youtube.com/watch?v=JmSNUeBG-PQ&list=PLQY2H8rRoyvyK5aEDAI3wUUqC\_F0oEroL&index=5

TensorFlow (2023c). Training and Evaluating Models in TensorFlow. Retrieved from https://www.youtube.com/watch?v=-vHQub0NXI4&list=PLQY2H8rRoyvyK5aEDAI3wUUqC\_F0oEroL&index=6

**Appendix**

**Appendix A –** Refer to the accompanying Jupyter Notebook file for code implementation and detailed comments.

**Appendix B** – Visualizations

A collection of different types of clothing

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of a graph of a training

Description automatically generated with medium confidence

A graph with numbers and symbols

Description automatically generated

A screenshot of a computer

Description automatically generated

A graph of different colored lines

Description automatically generated