EIN 4601C -Automation and Robotics

Instructor:

Dr. Susana Lai-Yuen

Semester Project

Image Classification using Machine Learning Models

Team 12:

Maria Villarreal

Antonietta Della Sala

Laiba Aurangzeb

Cagla Unsalan

April 28, 2025

Project Description

Main Problem

The objective of this project is to apply machine vision and supervised learning techniques to automatically classify clothing items from grayscale images. The Fashion-MNIST dataset, consisting of 28x28 pixel images across ten fashion categories (t-shirt/top, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, ankle boot), was used for training and evaluation of models (1). The project involved implementing and comparing two classification setups: Setup 1: K-Nearest Neighbors (KNN) Classifier and Setup 2: Multilayer Perceptron (MLP) Classifier with one hidden layer.

Following the concepts learned in the course modules, the models were trained on a labeled training set and evaluated using a separate test set. Hyperparameters for each model were optimized using cross-validation techniques. The performance of each of the final models was assessed using confusion matrices, classification metrics, and training time. Additionally, we analyzed which clothing items were most difficult to classify, the difference in classification results for the two classifiers using some image examples and proposed improvements for future work.

This project showcases the practical use of machine learning algorithms for industrial image classification tasks, which are essential in fields like robotics, retail, automation, and quality inspection etc.

Methodology

Implementation Strategy

This project was implemented by successfully executing our Python program inside the Jupyter Notebook environment, using the browser and cloud-based tool, Google Colab. Several open-source libraries were utilized to complete the work: TensorFlow from the Keras package for loading the Fashion-MNIST dataset, Scikit-learn for machine learning model development and evaluation, and NumPy for numerical data handling. Matplotlib and Seaborn were employed for visualizing the results through confusion matrices and sample image predictions. The code was structured into organized sections, clearly documenting the processes of data loading, preprocessing, model training, hyperparameter tuning, evaluation, and result analysis.

Preprocessing

Preprocessing steps were kept simple and consistent with the project's requirements. The Fashion-MNIST images, originally 28x28 pixel grayscale arrays, were normalized by scaling the pixel intensity values from [0, 255] to a [0, 1] range. This normalization ensured numerical stability during model training. Since the dataset already contained 60,000 images to serve as a training set and 10,000 images as a test set, so we did not have to do any such split. To prepare the data for the classifiers, each image was flattened into a 784-dimensional feature vector. No additional feature engineering, data augmentation, or dimensionality reduction techniques were applied, aligning with the project's instruction to use raw image data.

Cross-Validation Approach and Justification

Model evaluation was conducted using StratifiedKFold cross-validation to preserve the proportional distribution of classes within each fold, which is important for multi-class classification problems, to handle the class imbalance issue. For both setups, we used k = 5 (no. of folds) which means data was split into 5 folds, where model was trained using 4 folds (k-1) and then tested on the last remaining fold. GridSearchCV was employed to automate the hyperparameter tuning process, integrating StratifiedKFold cross-validation to systematically assess each hyperparameter combination. For the K-Nearest Neighbors (KNN) model, the number of neighbors (n_neighbors) was tuned by evaluating values of 3, 5, and 7. For the Multilayer Perceptron (MLP) setup, the grid search tested the following values for each of the 3 types of hyperparameters: size of hidden layer (32, 128, 224, 320), activation functions ('relu' and 'tanh') and learning rates (0.005 and 0.05).

Techniques/ Setups Evaluated

Each machine learning setup was trained on the training dataset and evaluated on a separate test set that was kept strictly isolated during model development. After the hyperparameters producing the highest validation accuracy were identified for each setup, the corresponding models were retrained on the full training dataset and evaluated on the test dataset to assess final model performance. Each model's performance was evaluated using accuracy, confusion matrices, and training time, followed by a comparative analysis that was conducted to identify strengths and limitations of each approach.

Identification of Best Hyperparameters

The table below summarizes the hyperparameters searched for both setups as well as the best hyperparameter found using cross-validation:

Model (setup)	Hyperparameter Evaluated	Values Tested	Best Hyperparameter
K-Nearest Neighbors (KNN)	number of neighbors ("n_neighbors")	3, 5, 7	7
Multilayer Perceptron (MLP)	Size of hidden layer ('hidden_layer_sizes')	32, 128, 224, 320	320
	Activation function ('activation')	'relu', 'tanh'	'relu'
	Learning rate	0.005, 0.05	0.005

Table 1: Identification of Best Hyperparameters for both setups

We believe that the no. of neighbors = 7 for the KNN model is a good choice as it strikes a balance between bias and variance. This hyperparameter value is neither too small nor too large, allowing it to smooth out noise better by making predictions using a decent number of surrounding points. Choosing a smaller value here might have caused overfitting and high variance as the model would be depending on very few neighbors and could easily be misled by the noise or outliers in the data.

For the MLP model, the largest hidden layer size of 320 from the values tested was selected as the best choice due to its contribution in providing the model with enough capacity to learn complex patterns well, preventing underfitting. This hidden layer size allowed the model to identify enough feature interactions while helping it to generalize well on unseen data without overfitting too much. The use of 'relu' activation function allowed for efficient model training and effectively handled non-linear data patterns (2). Additionally, although a smaller learning rate of 0.005 resulted in a longer training time for the MLP model, it allowed the model to converge steadily towards an optimal solution (higher final accuracy) without causing divergence as a small learning rate helps the model adjust its weights more gradually (3).

Results and Discussion

Confusion Matrices

The confusion matrices for both setups are presented below. They illustrate the number of misclassifications made by each model for each clothing item, providing a clear visualization of how each model struggles to correctly classify certain items across all classes. A misclassification occurs when the predicted label is not equal to the true label as shown by the values > 0 in the purple boxes in ***Figure 1*** and Error! Reference source not found..

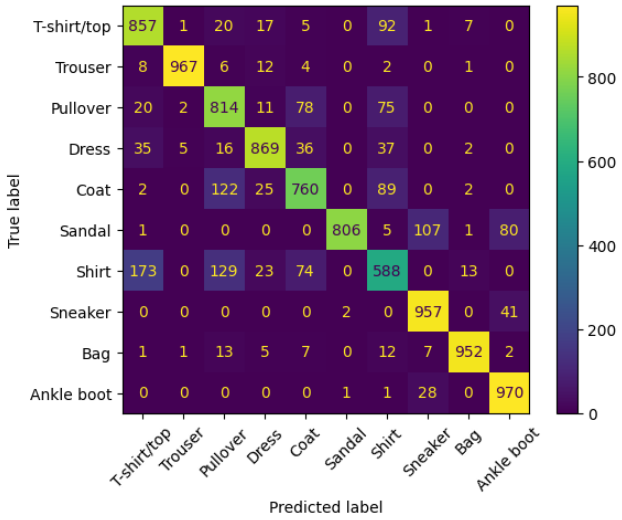

Figure 1: Confusion Matrix for K-Nearest Neighbors (KNN) model (setup 1)

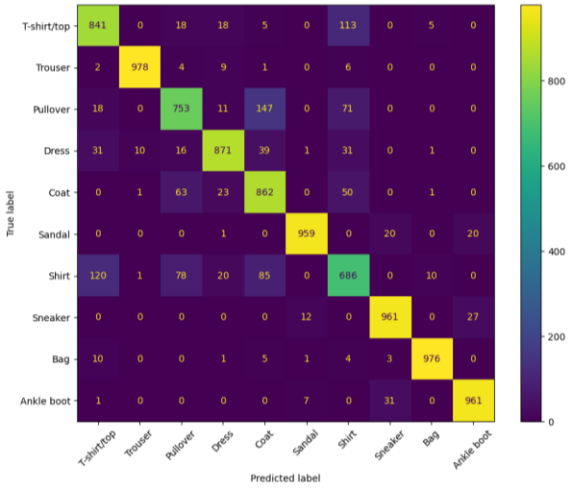

Figure 2:Confusion Matrix for Multilayer Perceptron (MLP) model (setup 2)

Challenging Items to Classify

An analysis of ***Figure 1*** and ***Figure 5*** indicates that the KNN model frequently misclassified shirts, often confusing them with t-shirts (highest misclassification value of 173 among all classes) and pullovers (i.e., second highest value of 129 incorrect predictions). The KNN model also struggled to classify sandals as it incorrectly predicted it as a sneaker 107 times and as an ankle boot 80 times. The MLP model showed better performance in these difficult categories, reducing the number of misclassifications significantly. It misclassified a shirt as a t-shirt 120 times and as a pullover 78 times which is considerably less than the KNN model's misclassifications for the same item.

There are multiple reasons for these repeated misclassifications, the primary one being the similar visual traits of certain clothing items like shirts, t-shirts, and pullovers which get even harder to identify in grayscale 28x28 pixel images with no color or texture information. These items may have similar features like sleeve length and neckline shape etc. Similarly, items like sandals, sneakers and ankle boots share identical design elements like straps and heel etc. These subtle differences are difficult to detect using mainly the pixel intensity patterns. Additionally, KNN's higher number of misclassifications than MLP can be attributed to its simple instance-based nature in contrast to MLP's deep learning nature. KNN depends significantly on nearby neighbors and can be quite sensitive to minute differences between classes whereas MLP models are capable of learning intricate, abstract features through its hidden layers, allowing it to differentiate between similar classes more effortlessly.

Model Comparison

Model (setup)	Accuracy %	Training Time (Seconds)	Average Precision	Average Recall
K-Nearest Neighbors (KNN)	85%	442.72	86%	85%
Multilayer Perceptron (MLP)	**88%**	5211.87	**89%**	**88%**

Table 2: Comparison of results b/w the two setups

Both models performed well on this classification problem and had high values for each of the evaluation metrics. A prominent performance difference between the two is that the MLP classifier achieved a higher accuracy, average precision and recall compared to the KNN model but required a considerably longer training time as seen in ***Table 2***. However, the performance improvement justifies the extra computational cost, especially for tasks such as image classification that require higher accuracy.

We also conducted a side-by-side model comparison to visualize the differences in image classifications between the two models i.e., identify some example images which are correctly classified by one setup and incorrectly classified by another setup. Results show that both models have disparities in their prediction results as some of the same clothing items were correctly classified by one model but incorrectly classified by another.

Choosing the Best Setup

We found that the Multilayer Perceptron (MLP) was the best setup because it delivered the highest classification accuracy and was better at generalizing across all fashion categories as seen in ***Figure 3***. Its hidden layer design allowed it to pick up on complex decision patterns that KNN failed to capture. The MLP also showed a stronger ability to recognize subtle patterns in the data, helping it makes clearer distinctions between categories that looked very similar such as t-shirts/ tops, shirts, pullovers as well as sandals, sneakers, and ankle boots. Setup 2 (MLP) is not only better at reducing false positives (as shown by its higher average precision value of 89% than setup 1's precision of 86%) but is also better at minimizing false negatives (as depicted by its higher recall value) according to ***Figure 4***, leading to more reliable and balanced performance across all classes. MLP's ability to adjust internal parameters during training leads to its high performance when it comes to generalizing to unseen data as opposed to KNN's habit to memorizing the training data, reducing the latter model's generalization ability (4). Although MLP takes more time to train, it delivered more accurate and precise results compared to KNN, so the performance gain is worth the computation time.

Opportunities for Improvement

Looking ahead, there are a few ways the model could be improved. One approach is building a deeper neural network by adding more hidden layers, as it can help the model capture more complex patterns in the data. Another way to improve the model would be using lower learning rates to help the model converge more consistently. Although it may result in a longer training time, it would also mean more stable and reliable convergence. A lower learning rate helps the model find the optimal solution at a measured pace (3). Additionally, applying dimensionality reduction techniques like Principal Component Analysis (PCA) can improve feature extraction. This can enhance the model's ability to focus on the most important features while reducing noise and irrelevant information. PCA can also reduce overfitting and make the model more efficient (5). And lastly, increasing the number of training epochs can boost learning and performance. This way the model will have more time to adjust its internal weights, improving its accuracy and performance.

Ethical, Economic, and Societal Impacts

Automated image classification systems are accompanied with significant advantages such as accelerated processing, enhanced consistency, higher accuracy, and straightforward scalability for industrial applications, thus streamlining the image classification process. Nonetheless, ethical issues emerge when models are developed using biased or unbalanced datasets, potentially resulting in unjust or incorrect outcomes (6). A significant ethical concern of this type of automation is the leaking or storing of sensitive data such as personal identifiers (e.g., car of a person), leading to privacy infringement of the said individual. Moreover, the use of automated image classification systems is risky in sensitive areas such as healthcare where a wrong disease diagnosis (incorrect prediction) can either cause the patient his/ her life if a disease fails to get diagnosed by the system or cause them extreme stress if they are diagnosed with a disease they do not have in actual.

The major economic impact of automation is its role in exacerbating the threat of displacing employees in roles historically held by humans such as an operator doing manual sorting of items in a manufacturing facility. On the contrary, automation is favorable for companies that aim to reduce their labor expenses so altogether, it depends on which perspective you are looking from.

On a wider societal context, automation can enhance product quality by performing the quality control job better than humans. This is due to the automated image classification systems' greater ability to detect defects in the product as compared to a human eye, making the product safer for use. This eventually paves the way for increased customer satisfaction and stronger brand image (7). At the same time, automation contributes to a digital divide worldwide as countries with advanced technological infrastructures and stronger economies have better opportunities to develop stronger AI systems. The heavy dependence on technology not only makes people lazy but also limits their creative abilities, making it harder for them to perform a task during technical crashes. Excessive technological reliance of industries on automated image classification systems puts their operation systems at a higher risk during technical failures. That being noted, it is essential to apply these technologies with caution to ensure fairness in data management and minimize the negative effects.

Appendix A.

References

1. Team, K. (n.d.). *Keras Documentation: Datasets*. https://keras.io/api/datasets/
2. Brownlee, J. (2020, August 20). A gentle introduction to the rectified linear unit (ReLU). MachineLearningMastery.com. https://machinelearningmastery.com/rectified-linear-activation-function-for-deep-learning-neural-networks/
3. What is learning rate in machine learning? Pure Storage. (n.d.). https://www.purestorage.com/knowledge/what-is-learning-rate.html#:~:text=A%20high%20learning%20rate%20can,get%20stuck%20in%20suboptimal%20solutions.
4. Equbal, M. T. (2024, October 17). *Overcoming common pitfalls in Multilayer Perceptron: A guide to understand & handling overfitting...* Medium. https://medium.com/the-modern-scientist/overcoming-common-pitfalls-in-multilayer-perceptron-a-guide-to-understand-handling-overfitting-8131b5e94e47#:~:text=Overfitting%20occurs%20when%20an%20MLP%20learns%20the,during%20training%20but%20poor%20performance%
5. IBM. (2025, April 17). *What is dimensionality reduction?* IBM. https://www.ibm.com/think/topics/dimensionality-reduction#:~:text=Dimensionality%20reduction%20techniques%20such%20as,the%20original%20data's%20meaningful%20properties.&text=This%20amounts%20to%20removing%20irrelevant,through%20variable%20extraction%20or%
6. Matthew G. Hanna a b, M. G. H., Liron Pantanowitz, Brian Jackson, Octavia Palmer, Shyam Visweswaran, Joshua Pantanowitz, Mustafa Deebajah, Hooman H. Rashidi. (2024, December 16). Ethical and bias considerations in Artificial Intelligence/Machine Learning. Modern Pathology. https://www.sciencedirect.com/science/article/pii/S0893395224002667#:~:text=The%20choices%20made%20in%20algorithmic%20design%2C%20such,existing%20disparities%20when%20deployed%20in%20real%2Dworld%20applications.
7. Filippi, E., Bannò, M., & Trento, S. (2023, March 10). *Automation Technologies and their impact on employment: A review, synthesis and future research agenda*. Technological Forecasting and Social Change. https://www.sciencedirect.com/science/article/pii/S0040162523001336

Appendix B.
Supplementary Material

	precision	recall	f1-score	support
T-shirt/top	0.82	0.84	0.83	1000
Trouser	0.99	0.98	0.98	1000
Pullover	0.81	0.75	0.78	1000
Dress	0.91	0.87	0.89	1000
Coat	0.75	0.86	0.80	1000
Sandal	0.98	0.96	0.97	1000
Shirt	0.71	0.69	0.70	1000
Sneaker	0.95	0.96	0.95	1000
Bag	0.98	0.98	0.98	1000
Ankle boot	0.95	0.96	0.96	1000
accuracy			0.88	10000
macro avg	0.89	0.88	0.88	10000
weighted avg	0.89	0.88	0.88	10000

Figure 3: Detailed results for the MLP model

	precision	recall	f1-score	support
T-shirt/top	0.78	0.86	0.82	1000
Trouser	0.99	0.97	0.98	1000
Pullover	0.73	0.81	0.77	1000
Dress	0.90	0.87	0.89	1000
Coat	0.79	0.76	0.77	1000
Sandal	1.00	0.81	0.89	1000
Shirt	0.65	0.59	0.62	1000
Sneaker	0.87	0.96	0.91	1000
Bag	0.97	0.95	0.96	1000
Ankle boot	0.89	0.97	0.93	1000
accuracy			0.85	10000
macro avg	0.86	0.85	0.85	10000
weighted avg	0.86	0.85	0.85	10000

Figure 4: Detailed results for the KNN model

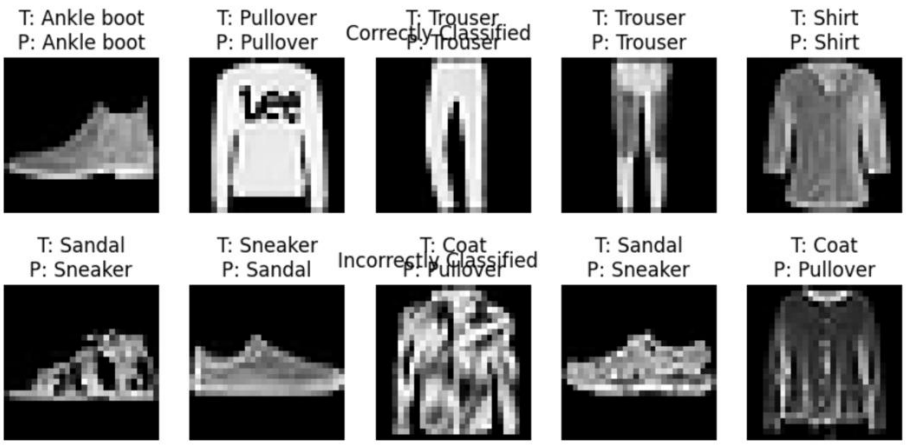

Figure 5: Actual vs. predicted outcomes for some image examples using KNN model