CS 178: Final Project

Investigating classification methods for Fashion-MNIST

Authors	Anthony Cusimano, Crystal Chiu, Junwei Hu
IDs	55588645**,** 33702086, 36593929
Dataset	Fashion-MNIST
Classification Methods	kNN, logistic regression, feedforward neural network, SVC (support vector classifier)

Team Name: Fashion Model(er)s

Date: June 10, 2024

I. Summary

We analyzed the Fashion-MNIST dataset and performed image classification of fashion items using four classification methods: k-nearest neighbors (kNN), logistic regression, feedforward neural network, and support vector classifier (SVC). On the full dataset, we find that SVC achieves the highest test accuracy at 90.5%, followed closely by the feedforward neural network at 89.6%. kNN had a reasonable accuracy at 86.2%, but it took considerably longer than the others to make predictions. Lastly logistic regression had the lowest accuracy at 84.6%. Overall, all four models struggled on differentiating between T-shirts, shirts, and pullovers, but the feedforward neural network shows the best balance between accuracy and computational efficiency.

II. Data Description

In this project, we utilized the Fashion-MNIST dataset, consisting of 70,000 images of fashion products from 10 categories (7,000 each). Each image is 28x28 pixels, so there are a total of 784 features representing each individual pixel, and there is no missing data in this dataset. Each pixel is represented by an integer value from 0-255 where 0 represents black and 255 represents white, with in-between values representing shades of gray. The target variable is an integer 0-9 representing the unique categories: t-shirt, trouser, pullover, dress, coat, sandal, shirt, sneaker, bag, and ankle boot. Figure 1 below shows the centroid for each class, and we notice that the pullover, coat, and shirt appear very similar which makes differentiating between them more difficult.

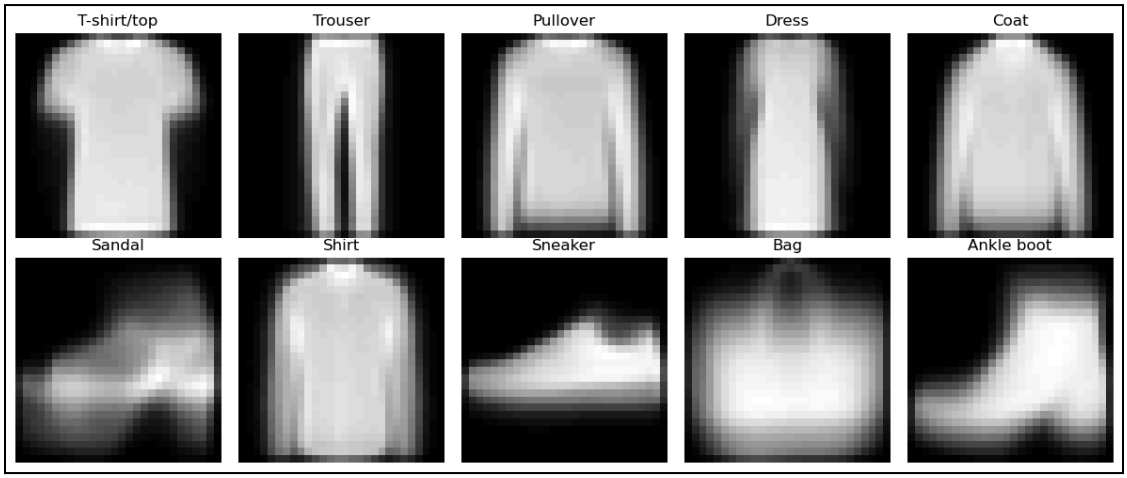

Figure 1: Plotting the centroid/average image for each class

A relevant paper is “An Architecture Combining Convolutional Neural Network (CNN) and Support Vector Machine (SVM) for Image Classification” by Abien Fred Agarap which uses an SVM model as an alternative to the softmax function to obtain a test accuracy of 91.86%. This study was more exploratory and did not perform any data preprocessing, and used one set of hyperparameters including a batch size of 128 and

10,000 steps. Although the results are fairly strong, SVM performed similarly to using softmax and the author suggests using preprocessing and a more sophisticated CNN model which provides a starting point to further investigate.

III. Classifiers

Our first classifier is k-nearest neighbors (kNN) which assigns a data point to the most common class among the k points nearest to it. For this model, the hyperparameters we tuned are the number of neighbors to consider, the distance metric to use (Euclidean or Manhattan), the weight of each neighbor being 'uniform' (all neighbors contribute equally) or 'distance' (closer neighbors have a greater influence), and the algorithm to use ('auto', 'ball_tree', 'brute', and 'kd_tree'). We used the KNeighborsClassifier from scikit-learn to implement this model.

Next, we used logistic regression which puts the hyperparameters and a bias term into a linear combination. It then uses a sigmoid function to map the output of that equation to a probability which decides the class it belongs to. The hyperparameters used for our logistic regression model were L2 regularization (adds penalty), fit_intercept (added a bias term), C (inverse regularization strength), max iterations (maximum to converge), and two different solvers (lbfgs and saga). LogisticRegression from scikit-learn was used to implement the model.

Our third classifier is the feedforward neural network which has a one-directional flow of information from the input nodes, to the hidden layers, and to the output nodes, outputting the most likely class for each data point. There are many tunable hyperparameters, and we experimented with varying hidden layer sizes (neurons per layer), activation functions, solver, learning rates, batch sizes, and maximum iterations. We used the MLPClassifier from scikit-learn to implement this model.

Finally, the last classifier used was a Support Vector Classifier (SVC). It aims to find the hyperplane that best separates the data points by the largest margin, as the decision boundary with a larger margin will likely be the best general case for unseen data. The hyperparameters used were the C (regularization parameter), gamma scale (influence of a single training example adjusts based on data), and rbf kernel (non-linear decision boundary). It was implemented using SVC from scikit-learn.

IV. Experimental Setup

Before training the models, we split the data into 3 sets: 60% as training data, 20% as validation, and 20% as testing. This allowed us to iteratively optimize the hyperparameters using the validation set, and use the test set at the very end to get more accurate metrics of unseen data. To ensure reproducibility, we set the random seed to 1234 for all random processes in this project. All models were trained using a 4-fold cross validation.

To select the best hyperparameters for each model, we used grid search to attempt many possible hyperparameter combinations and selected the model with the highest validation accuracy. Once we selected the optimal models, we aimed to compare their strengths and weaknesses using testing accuracy, prediction time, and their confusion matrices. These provide a fairly comprehensive overview of each model's performance in a real-world scenario, as well as when to employ each one. For accuracy and prediction time, we also investigated how these metrics change for each model as the training or testing sets, respectively, increase in size.

For the feedforward neural network, we performed the additional preprocessing step of standardization by scaling the features to have a mean of 0. Because the neural network is sensitive to the data scale, this ensures that the network converges more quickly.

Ultimately, the following are our optimal models. For kNN, we used 5 neighbors with Manhattan distance and weighted points by the inverse of their distance, as well as the 'auto' algorithm to compute nearest neighbors. For the logistic classifier, we used an L2 penalty term, regularization parameter C = 0.01, 10,000 maximum iterations, and a 'saga' solver. In the feedforward neural network, we used 3 hidden layers with sizes 356, 128, and 64 as well as the ReLU activation function, 'adam' solver, an alpha of 0.001, batch size of 256, and a constant learning rate initialized at 0.0005. The maximum iterations were capped at 300. Lastly, our SVC model uses a regularization parameter of 10, a 'scale' gamma, and the 'rbf' (radial basis function) kernel which transforms the data into a higher-dimensional space to more easily find a linear separation.

V. Experimental Results

The metrics used to compare the four classifiers above were test accuracy, test set prediction time, and the ability to differentiate between T-shirts, shirts, and pullovers as they appear very similar. The table below summarizes the quantitative results on the full dataset. All prediction times were measured on a Windows PC (Intel i5-2500 processor).

Model	Test Accuracy	Test Set Prediction Time (seconds)
kNN	0.862	**1125.70**
Logistic Regression	**0.846**	**0.03**
Feedforward Neural Network	0.896	0.34
Support Vector Classifier	**0.905**	107.27

Table 1: Comparison of models on the full Fashion-MNIST dataset

We see that the SVC has the highest test accuracy, but takes a relatively large amount of time to make predictions while the feedforward neural network has a very similar accuracy with significantly faster prediction times. Logistic regression had the lowest test accuracy but also the fastest predictions at only 0.03 seconds, highlighting the tradeoff between accuracy and efficiency. Lastly, kNN had a decent accuracy but with many calculations being performed during testing, its prediction time took over 1125 seconds (18 minutes) which is significantly slower than the others. This stark difference in time complexity is clear in Figure 2 below which compares prediction times by the training set size.

We also compared how the accuracy changes as the training size increases, and in Figure 2, it is evident that all four models exhibit similar behavior with a rapid increase in accuracy up to around 10,000 training examples and then a slower increase after that point. Interestingly, the lines do not cross, indicating that the training size does not change the models' relative performance rankings based on accuracy.

Lastly, Figure 3 in the appendix displays the confusion matrices for each model which explains which examples each model struggled on. All four models appear to have difficulty classifying T-shirts, shirts, pullovers, and coats which all appear to be similar in the images. Between the top two performing models, SVC and the neural network, the neural network does a better job at classifying shirts as the SVC more often mislabels them as T-shirts. However, the SVC does a better overall job at classifying all of the other categories.

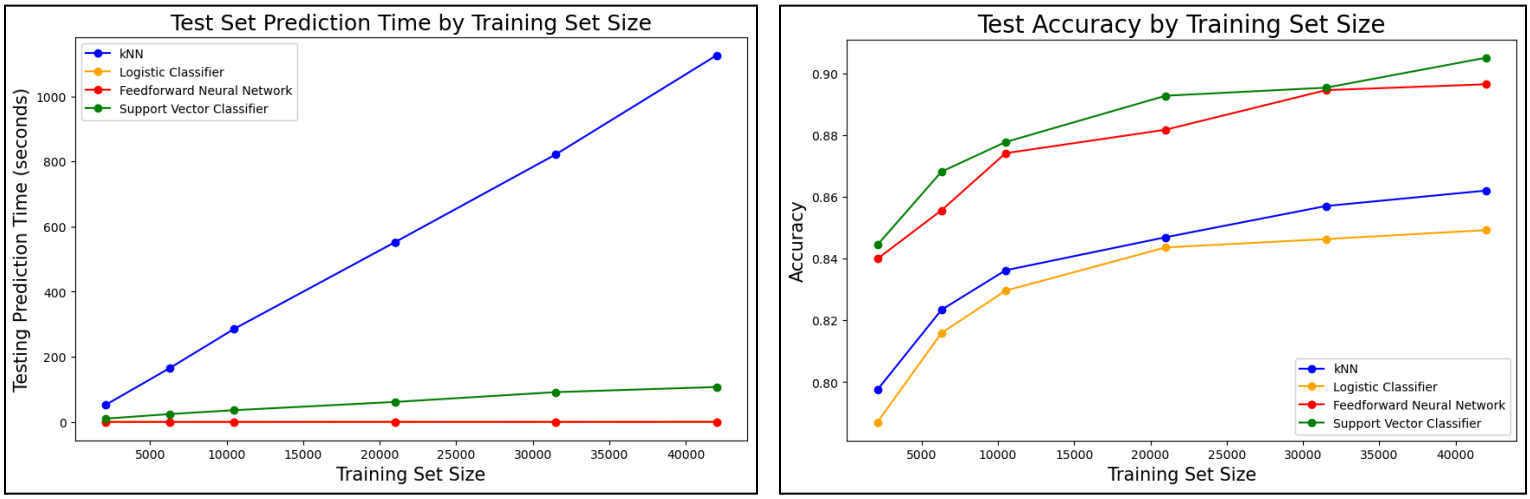

Figure 2: Test set prediction time and accuracy by training set size

Note: Logistic Classifier hidden under red line (Prediction time vs training size)

VI. Insights

After this experiment, we gained more insight on the advantages and disadvantages of each classifier. For example, although kNN works well for smaller

datasets, its computational cost makes it inefficient for large datasets, as it has to calculate the distance from one example to all others in the training set. We also saw that the linear classifier (logistic classifier) performed relatively poorly; the data, being high-dimensional images, was not easily linearly separable and too complex for a linear classifier to draw a good decision boundary. Non linear classifiers such as the SVC with an RBF kernel, and the feed forward neural network were better suited to capture the patterns and relationships of the data. Surprisingly, the neural network had strong performance while also having very fast prediction times making it a viable choice for real-world use cases. Finally, when examining the examples the classifiers struggled with, there tended to be ambiguity on what the image represents, even to a human. For example, many shirts look similar to t-shirts. Classification performance, particularly on non-linear classifiers, would likely improve if tested on higher resolution images, since the pixelation on some examples creates more ambiguity. However, the computational time would increase, slowing down training and prediction.

VII. Contributions

Anthony performed data investigation and literature review, and wrote the data description section above. He also focused on tuning the kNN and SVC models, and contributed to writing the experimental setup.
Crystal implemented and tuned the logistic regression model, wrote the classifier summary for the logistic and SVC models, and wrote the insights section.
Junwei performed tuning of the feedforward neural network, wrote the experimental results, and contributed to tuning the final SVC model.

VIII. Appendix

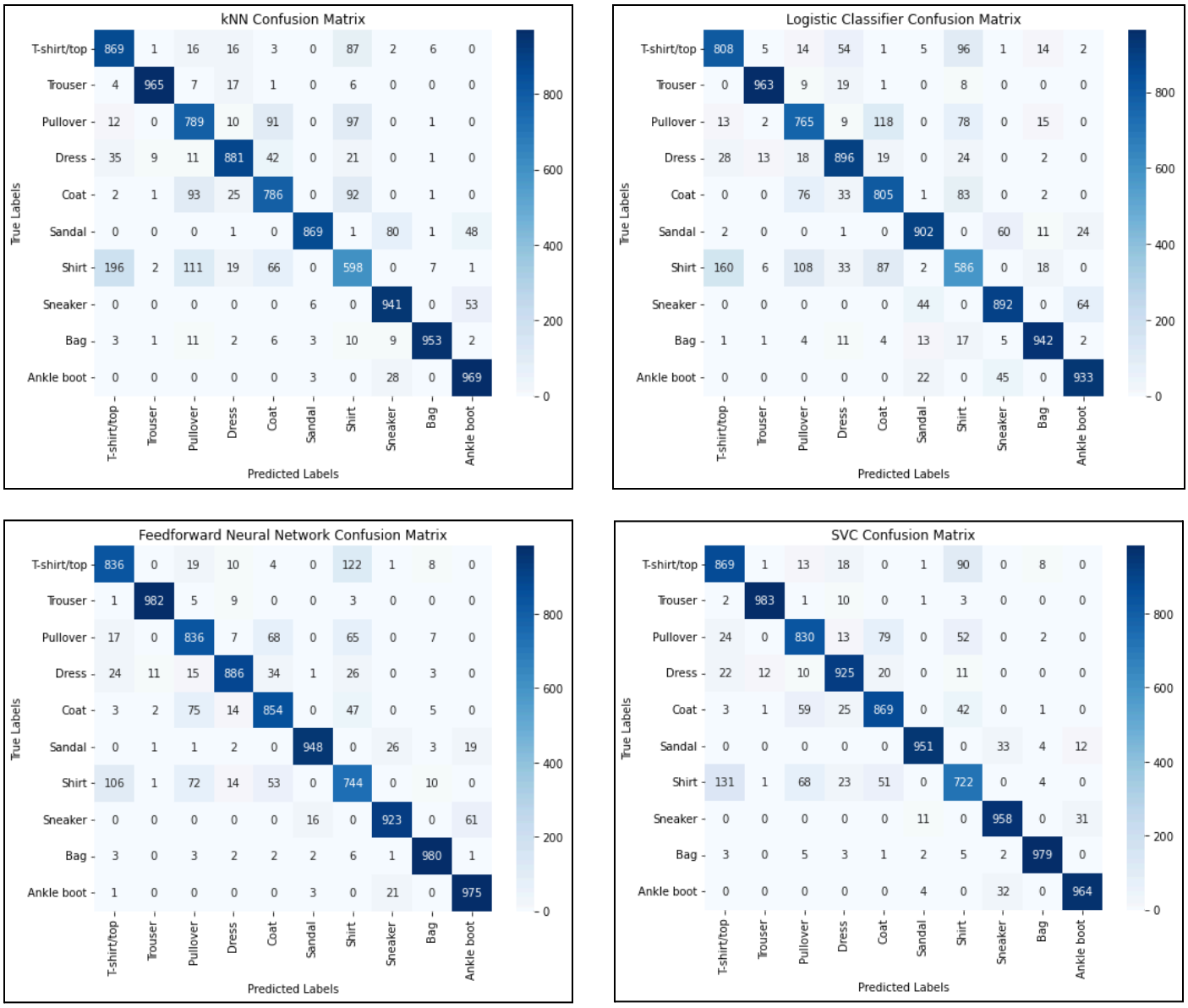

Figure 3: Confusion matrices for the four models on the full dataset