STAT426 Final Report
James Augustine Matthew Hamilton Mooney
119394311

INTRODUCTION

In this report, we investigated and compared the performance of seven supervised machine learning models on the Fashion-MNIST dataset available directly through the torchvision python package. This dataset contains 70,000 28x28 grayscale images of fashion products classified by 10 categories. The dataset is already split between a 60,000 item training set and a 10,000 item test set (https://arxiv.org/abs/1708.07747).

Prior to the application of these machine learning models and methods, we first imported the necessary packages to acquire, analyze, and manipulate the MNIST data before preparing said data. We first flattened it, before applying Principal Components Analysis (PCA) to project the 784-dimensional (28 x 28 = 784) dataset into a lower dimension. We first find the projection suitable for the training data, before applying said projection to the test data. The purpose of this PCA is to reduce the dimensional space required, and the random noise and computational cost incurred, while performing our analyses. We were able to use only 69 dimensions to account for 90% of the variance, compressing the data by over 91%.

For the majority of this investigation, we limit our model training to a 1000 item subset of the greater total training dataset. Additionally, we define a helper function to evaluate test accuracy for each class investigated. In the case of several models (K-Nearest Neighbors, Linear and Nonlinear Support Vector Machines), we opted to use 5-fold Cross Validation to fine tune the parameters (k, C, γ) of each model, scored via accuracy.

In this investigation, we evaluate the performance of the following methods:

- K-Nearest Neighbors Classification
- Linear Support Vector Machine Classification
- Nonlinear Support Vector Machine Classification
- Random Forest
- Linear Discriminant Analysis
- Quadratic Discriminant Analysis
- Convolutional Neural Network

For each model, we shall first describe the general method and its structure, before providing a more in depth investigation regarding the model's performance on the MNIST dataset and any additional quirks and intricacies worth noting.

K-NEAREST NEIGHBORS CLASSIFICATION

The K-Nearest Neighbors (K-NN) classifier assigns each test point a label in accordance with the majority label of the k-nearest training points to x.

In this report, we tuned the parameter k by evaluating each k in k_vals = [1,3,5,7,9,11] using the GridSearchCV function from the scikit-learn library to perform 5-fold cross validation upon both the full and PCA-reduced training datasets. For each k, the model was trained on four folds and validated on the fifth, while the k that attained the highest accuracy was chosen to be fitted onto the test data.

In both executions (full and PCA-reduced datasets), k = 1 yielded optimal results, while total accuracy varied from 75.06% in the case of the full flattened dataset to 75.59% in the case of the PCA-reduced dataset. Given that the use of the PCA data not only yields greater accuracy, but also reduces the number of relevant dimensions by over 91%, along with reducing associated computational costs, it is our natural recommendation to use the PCA-reduced test dataset.

LINEAR SUPPORT VECTOR MACHINE CLASSIFICATION

Following our investigation of the K-NN classification model, we tested the performance of support vector machine (SVM) classification models. To start, we implemented a linear support vector machine model by selecting model SVC(kernel = "linear"), which seeks a hyperplane $w^T x \pm b = 0$ that maximizes distance between classes.

In the case of our investigation, we tuned the parameter C by evaluating each C in C_vals = [0.001,0.01,0.1,10.0,100.0] using the GridSearchCV function from the scikit-learn library to perform 5-fold cross validation upon the PCA-reduced training dataset. For each C, the model was trained on four folds and validated on the fifth, while the C that attained the highest accuracy was chosen to be fitted onto the test data.

In the case of our linear SVM model, we found that parameter value C = 0.01 yielded the best performance, and this model yielded a total accuracy across classes of 79.58% on the PCA test dataset.

NONLINEAR SUPPORT VECTOR MACHINE CLASSIFICATION

Following our investigation of the linear SVM model, we then analyzed the performance of a nonlinear SVM. More specifically, we implemented a nonlinear SVM using the radial-basis-function, $K_\gamma(x,x') = \exp(-\gamma \, || \, x - x' \, ||^2 \,)$, to implicitly map data points to a higher dimensional space, allowing for easier separation of data that is non-linearly separable in lower dimensions.

Like the prior linear SVM, in the case of our nonlinear SVM we again tuned the parameter C by evaluating each C in C_vals, but did so while also tuning γ by evaluating each gamma in gamma_vals. This doubled form of 5-fold cross validation meant that 25 total configurations were evaluated. We found that in the case of our nonlinear SVM model, the parameter values C = 10.0 and gamma = 0.001 yielded the best performance, with the model itself yielding a total accuracy across classes of 80.74% on the PCA test dataset.

RANDOM FOREST

After analyzing the performance of linear and nonlinear SVM methods, we analyzed the performance of the random forest method, which takes bootstrapped resamples of the training data, fits decision trees onto this data, and averages their classifiers.

We applied a model constructed of 500 decision trees created by the RandomForestClassifier where each tree is fitted onto bootstrapped resamples of the PCA training data. We then fit and applied these decision trees onto the PCA test dataset. In our investigation, we found that the random forest model yielded a total accuracy across classes of 78.81% on the PCA test dataset.

LINEAR DISCRIMINANT ANALYSIS

Generally, linear discriminant analysis (LDA) models each class with a normal distribution and a common covariance matrix among all classes. The LDA seeks a linear combination of features that separate classes, allowing for linear decision boundaries.

In our investigation, we applied the LinearDiscriminantAnalysis() method, estimating prior class conditional densities, to the PCA training data and used this fit to predict the classes of test data points. We found that the LDA model yielded a total accuracy across classes of 77.5% on the PCA test dataset.

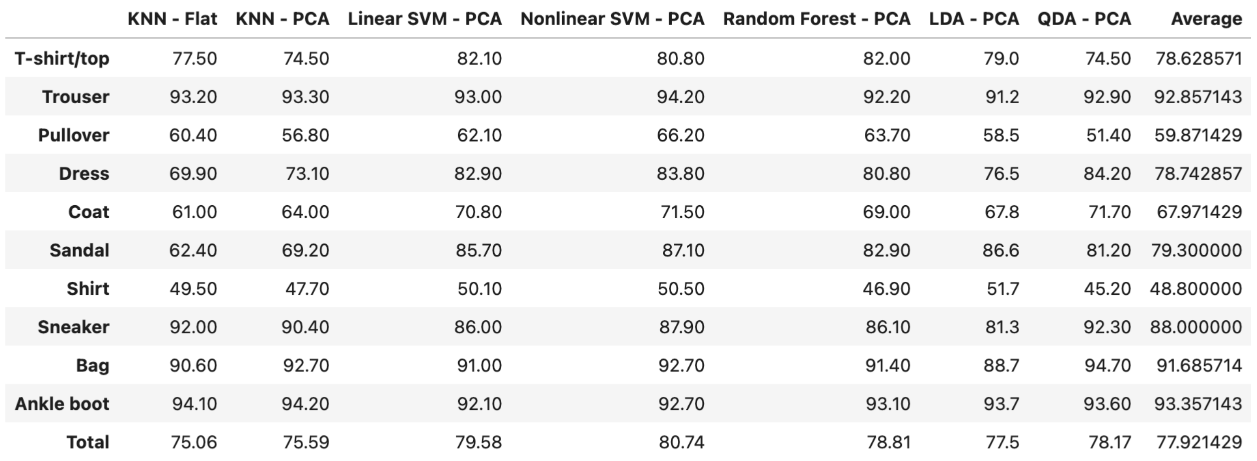
QUADRATIC DISCRIMINANT ANALYSIS

Similarly, we applied the quadratic discriminant analysis (QDA) method to analyze our data. The QDA method follows a very similar procedure to the LDA method, while relaxing its assumption regarding classes' covariance matrices, instead generalizing to allow each class to have its own covariance matrix and uses Bayes' rule to fit gaussian class conditional densities to the data.

In our investigation, we applied the QuadraticDiscriminantAnalysis() method to the PCA training data and used this fit to predict the classes of test data points. We found that the QDA model yielded a total accuracy across classes of 78.17% on the PCA test dataset.

COMPARISON OF NON-DEEP LEARNING MODELS

Before investigating the methods and performance deep-learning methods applied in our investigation, we first ought to take stock of the performance and comparison between the methods discussed so far. Provided below is a table outlining the accuracy of each non-DL method across each class:



	KNN - Flat	KNN - PCA	Linear SVM - PCA	Nonlinear SVM - PCA	Random Forest - PCA	LDA - PCA	QDA - PCA	Average
T-shirt/top	77.50	74.50	82.10	80.80	82.00	79.0	74.50	78.628571
Trouser	93.20	93.30	93.00	94.20	92.20	91.2	92.90	92.857143
Pullover	60.40	56.80	62.10	66.20	63.70	58.5	51.40	59.871429
Dress	69.90	73.10	82.90	83.80	80.80	76.5	84.20	78.742857
Coat	61.00	64.00	70.80	71.50	69.00	67.8	71.70	67.971429
Sandal	62.40	69.20	85.70	87.10	82.90	86.6	81.20	79.300000
Shirt	49.50	47.70	50.10	50.50	46.90	51.7	45.20	48.800000
Sneaker	92.00	90.40	86.00	87.90	86.10	81.3	92.30	88.000000
Bag	90.60	92.70	91.00	92.70	91.40	88.7	94.70	91.685714
Ankle boot	94.10	94.20	92.10	92.70	93.10	93.7	93.60	93.357143
Total	75.06	75.59	79.58	80.74	78.81	77.5	78.17	77.921429

Following this analysis, we came across the following notable conclusions:
1. The easiest classes to predict were Ankle Boot (93.36% Test Accuracy) and Trouser (92.86% Test Accuracy)
2. In contrast, the hardest classes to predict were Shirt (48.80% Test Accuracy) and Pullover (59.87% Test Accuracy)
3. Yielding a total Test Accuracy of 80.74% across classes, the Nonlinear (RBF) SVM model had the best performance across all classes. It is is for this reason that we recommend using the Nonlinear SVM Model above the other non-DL methods

Finally, though this does not change our recommendation, it is interesting to note that, despite both yielding fairly similar and middle-of-the-road total Test Accuracy results (ranked 4th and 5th, respectively), the QDA method outperformed all other methods on the most classes (4), while the LDA method underperformed all other methods on the most classes (4).

DEEP NEURAL NETWORKS

In this final section, we examine the structure and performance of Convolutional Neural Networks, a particular class of Deep Neural Networks.

Convolutional Neural Networks (CNN) are deep architecture models composed of many convolution and pooling layers of neurons that capture information and condense data, respectively. Moreover, activation layers are supplemented to existing layers to introduce nonlinearity to the model, allowing it to learn more complex patterns that exist in the data.

In the case of our implementation of a CNN, we loaded the Fashion-MNIST dataset into trainloader and testloader, kept batch size limited to 100 and maintained shuffling of the trainloader. We then defined a CNN with two convolution layers, each with ReLU activation functions attached. After confirming its architecture and shape, we trained the network with Adam optimizer for 50 epochs with cross-entropy loss, and printed the average training loss of each epoch. Upon completion of training, the CNN was implemented over the testloader to produce predictions. We found that our CMM model yielded a Total Accuracy across classes of 82.98%, outperforming every non-DL model tested prior.

We then trained the same CNN, now onto the full training set with 60,000 items. Because of the computational (and temporal) costs this requires, we only ran this network for 10 epochs. We then found that our CNN model yielded a Total Accuracy across classes of 90.63%, outperforming every other model, to include the previous version of CNN, by a staggering margin.

CONCLUSION

Throughout this report, we investigated and compared the performance of seven supervised machine learning models on the Fashion-MNIST dataset of 70,000 28x28 grayscale images classified by 10 categories. While some methods certainly faired better than others, the variance in test accuracy was generally far greater when comparing two classes analyzed by the same method, than this variance was when comparing two methods analyzing the same class, or the training data at large.

With respect to non-Deep Learning methods, the model with the lowest mean total test accuracy was the K-Nearest Neighbors method on the flattened original dataset (75.06%), while the non-DL model with the highest mean total test accuracy was the Nonlinear Support Vector Machine method applied to the PCA-adjusted data (80.74%). This difference in test accuracy (5.68%) is dwarfed by the difference between the accuracy of the final Convolutional Neural Network (90.63%), and every other method investigated in this report (differences ranging between 7.65% and 15.57%).

It is for this reason that we suggest that when the training dataset is sufficiently large, and computational budgets are sufficiently high, the use of CNN methods are highly recommended.

References

Amidi, Afshine, and Shervine Amidi. “Convolutional Neural Networks Cheatsheet.” *CS 230*, Stanford University, stanford.edu/~shervine/teaching/cs-230/cheatsheet-convolutional-neural-network.

“GridSearchCV.” *SciKit-Learn*, scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html.

“Pandas.DataFrame.Loc.” *Pandas*, pandas.pydata.org/docs/reference/api/pandas.DataFrame.loc.html.

“QuadraticDiscriminantAnalysis.” *SciKit-Learn*, scikit-learn.org/stable/modules/generated/sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.html#sklearn.discriminant_analysis.QuadraticDiscriminantAnalysis.

Ray, Deep. “CV”, 27 February 2025, University of Maryland, College Park, MD.

Ray, Deep. “CNN_MNIST.ipynb”, 1 May 2025, University of Maryland, College Park, MD.

Ray, Deep. “DNNs”, 17 April 2025, University of Maryland, College Park, MD.

Ray, Deep. “Nonlinear_SVM”, 7 April 2025, University of Maryland, College Park, MD.

Ray, Deep. “Project_Description.ipynb” University of Maryland, 7 May 2025

Ray, Deep. “4.Classification”, 6 February 2025, University of Maryland, College Park, MD.

“RBF.” *SciKit-Learn*, scikit-learn.org/stable/modules/generated/sklearn.gaussian_process.kernels.RBF.html.

Xiao, Han, et al. “Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms.” *arXiv.Org*, Cornell University, 15 Sept. 2017, arxiv.org/abs/1708.07747.