



# CS 178: Final Project

## CS 178 Final Project

<b>Names</b>	<b>Alonso Lopez</b>
<b>IDs</b>	<b>67467895</b>
<b>Dataset</b>	<b>Fashion MNist</b>
<b>Classification Methods</b>	<b>kNN, Logistic Regression, Feedforward Neural Network, and Convolutional Neural Network</b>

## Summary

This project focuses on analyzing the Fashion MNIST dataset. I employed several classification methods, including k-Nearest Neighbors (kNN), logistic regression, a feedforward neural network, and a convolutional neural network (CNN). Through comprehensive experimentation and analysis, I concluded that the convolutional neural network was most accurate. This is likely due to its convolutional layers which target local portions of the image to extract certain features.

# Data Description

For the project, I utilized the Fashion MNist dataset, which is a dataset composed of 28x28 grayscale images of 70,000 fashion products from 10 different categories. I explored this dataset by generating general statistics about the dataset, such as the mean and standard deviation. I also plotted histograms and scatter plots to visualize the various distributions of our dataset. During the analysis with the convolutional neural network, I found that I was able to reduce common misclassifications of the clothing labels that were common in the other models. This demonstrated the utility of convolutional backward propagation.

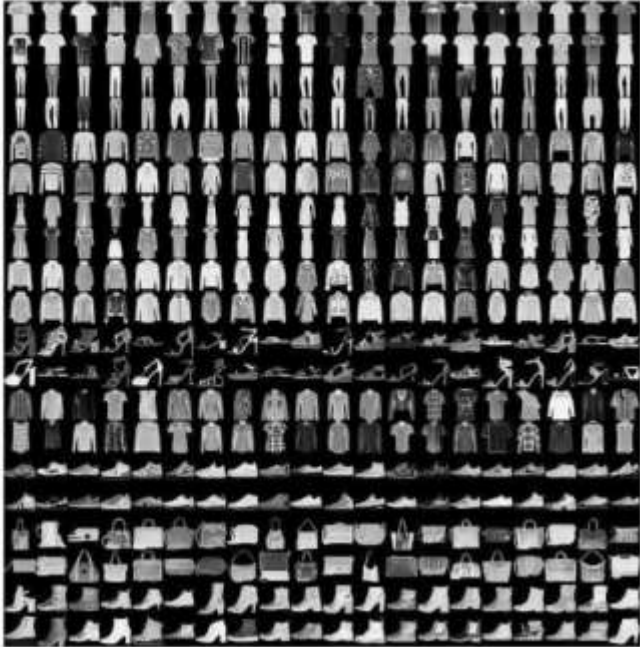
Label	Description	Examples
0	T-Shirt/Top	
1	Trouser	
2	Pullover	
3	Dress	
4	Coat	
5	Sandals	
6	Shirt	
7	Sneaker	
8	Bag	
9	Ankle boots	

Figure 0: Distribution of Labels in Fashion MNIST

In the article “Classification of Garments from Fashion MNIST Dataset Using CNN LeNet-5 Architecture”, the convolutional neural network LeNet-5 is trained and tested on the Fashion MNist dataset and achieved an accuracy score of 98% among other high performance metrics (Kayed et al.). This network used 6 hidden layers. My best performing convolutional neural network compares to the LeNet-5 network in the following ways: Both networks used 6 layers but mine used 2 convolutional layers, 1 pooling layer, and 3 forward layers whereas the LeNet-5 network used 2 convolutional layers, 2 pooling layers, and 2 forward layers. My best network was able to achieve 92% test accuracy compared to the LeNet-5 network’s 98%.

[https://www.researchgate.net/profile/Ahmed-Anter/publication/340237897\\_Classification\\_of\\_Garments\\_from\\_Fashion\\_MNIST\\_Data\\_set\\_Using\\_CNN\\_LeNet-5\\_Architecture/links/5eef3071a6fdcc73be909495/Classification-of-Garments-from-Fashion-MNIST-Dataset-Using-CNN-LeNet-5-Architecture.pdf](https://www.researchgate.net/profile/Ahmed-Anter/publication/340237897_Classification_of_Garments_from_Fashion_MNIST_Data_set_Using_CNN_LeNet-5_Architecture/links/5eef3071a6fdcc73be909495/Classification-of-Garments-from-Fashion-MNIST-Dataset-Using-CNN-LeNet-5-Architecture.pdf)

## Classifiers

The classification methods I used in this project are k-nearest neighbors(kNN), logistic regression, a feedforward neural network, and a convolutional neural network (CNN).

K-nearest neighbor is a supervised learning algorithm utilizing nearby known data points to predict an unknown point, adjustable by distance metric and the number of reference points, or k. I employed the KNeighborsClassifier from sklearn library due to its familiarity and utility in class settings.

Logistic regression, a binary classifier, estimates class probabilities using a logistic function, fine-tuned by regularization parameters, solver choice, and other settings like max\_iter and class\_weight. Sklearn's LogisticRegression, with 'multinomial' setting, is suitable for non-binary classification tasks like Fashion MNIST.

A feedforward neural network, versatile for classification and regression, consists of interconnected layers without cyclic connections, optimized by adjusting hidden layers, learning rates, and activation functions. I employed MLPClassifier from sklearn.neural\_network for its flexibility and ease of tuning.

A convolutional neural network (CNN) is specialized for image and video tasks, employing layers like convolutional, pooling, and fully connected layers for feature extraction and prediction. Performance optimization involves adjusting parameters like filter numbers, sizes, and strides in convolutional layers, along with learning rate and activation functions like ReLU or sigmoid. Techniques such as dropout and batch normalization mitigate overfitting, while parameters like batch size and epochs regulate effective training.

## Experimental Setup

For our experiments, I split the dataset into a training set of 80%, a training set of 20% and a testing set of 20%.

### **K-Nearest Neighbors (k-NN):**

I used 5-fold cross-validation to optimize the hyperparameter k for our k-NN classifier. This involved training and validating the model on different folds of the data and

selecting the k value that yielded the highest cross-validation accuracy. The final model was trained on the entire training set with this optimal k. For evaluation, I measured classification accuracy, precision, and recall on the test set to assess performance comprehensively.

### **Logistic Regression:**

I standardized the data and tuned the hyperparameter C using 3-fold cross-validation. C values ranging from  $10^{-3}$  to  $10^3$  were utilized and I selected the best C based on the mean cross-validation accuracy. The final model, trained with the optimal C, was evaluated on the test set using accuracy, a classification report, and a confusion matrix.

### **Feedforward Neural Network (FNN):**

I experimented with different learning rates ([0.0005,0.001,0.005,0.01]), batch sizes (256), and network architectures (single and multi-layer with varying nodes). Each hidden layer used ReLU activations. By comparing loss curves and performance, I identified the optimal network configuration that balanced complexity and generalization.

### **Convolutional Neural Network (CNN):**

Using 5-fold cross-validation, I trained the CNN model, optimizing hyperparameters like the number of epochs (10), learning rates ([0.0005,0.001,0.005,0.01]), and batch size (256). The architecture included two convolutional layers (32 and 64 filters), ReLU activations, a pooling layer, and two fully connected layers. I tested variations with different numbers of layers and nodes to find the best configuration.

### **The code for the experiments:**

[https://colab.research.google.com/drive/1wWYuO4qIt2Xh185C0RMikCBnMORFVbZ\\_?usp=sharing](https://colab.research.google.com/drive/1wWYuO4qIt2Xh185C0RMikCBnMORFVbZ_?usp=sharing)

# Experimental Results

## KNN Classifier

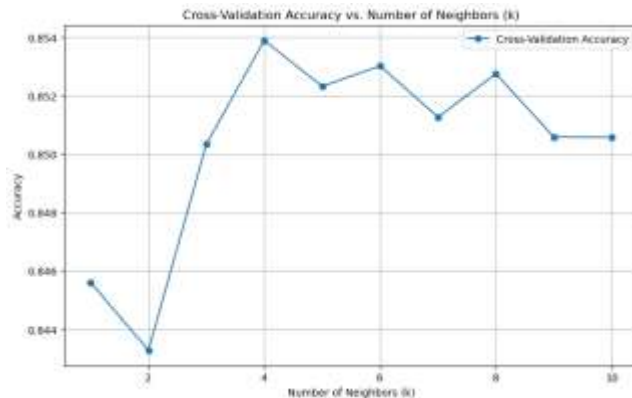


Figure 1.1: This figure gives the average accuracy score of the cross validation for the K Nearest Neighbors classifier with k values between 1 and 10

Final Test Accuracy: 0.851  
Final Test Precision: 0.854747590142785  
Final Test Recall: 0.8500597006490095

Figure 1.2: Final test scores for accuracy, precision and recall

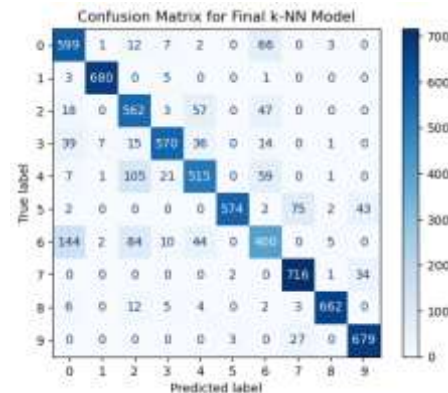


Figure 1.3: Confusion matrix for final model on test dataset

The k-NN classifier on the Fashion-MNIST dataset achieved a final test accuracy of 85.26%, with precision and recall metrics at 85.56% and 85.31%, respectively, indicating robust performance in correctly identifying and minimizing misclassifications. The validation accuracy peaked at k=4, suggesting this as the optimal number of neighbors. Consistent cross-validation accuracies affirm the model's stability. The confusion matrix highlights strong overall performance but reveals minor misclassifications between visually similar classes. Further analysis could explore error overlaps with other classifiers to identify common pitfalls.

# Logistic Regression

Final Model Accuracy: 0.8577				
	precision	recall	f1-score	support
0	0.79	0.83	0.81	698
1	0.98	0.96	0.97	689
2	0.77	0.77	0.77	687
3	0.85	0.87	0.86	682
4	0.78	0.78	0.78	709
5	0.94	0.93	0.94	698
6	0.65	0.61	0.63	689
7	0.98	0.93	0.92	753
8	0.96	0.95	0.96	694
9	0.95	0.94	0.95	709
accuracy			0.86	7000
macro avg	0.86	0.86	0.86	7000
weighted avg	0.86	0.86	0.86	7000

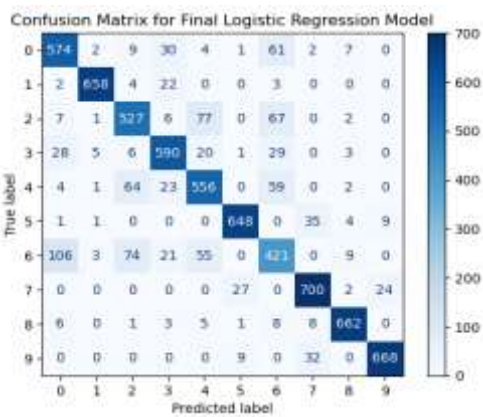
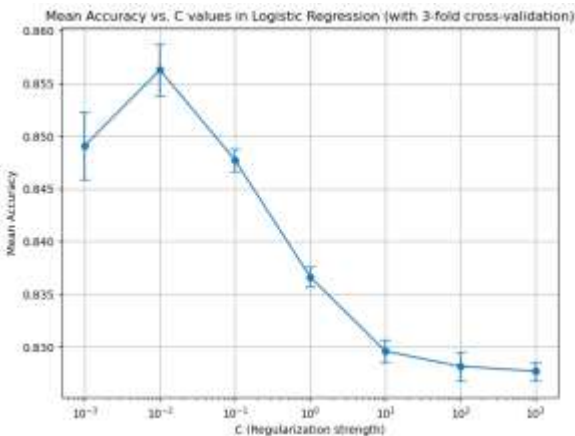


Figure 2.1: This figure gives the classification report for the Logistic Regression model

Figure 2.3: This figure demonstrates the mean accuracy of the cross validation compared with the different C values in the Logistic Regression classification (Right)



The Logistic Regression classifier on the Fashion-MNIST dataset achieved a final test accuracy of 85.77% with a precision and recall average of 86%. I analyzed the cross-validation accuracy of the model when tuning the different C values and the highest accuracy was seen when the C value was 10^-2. The confusion matrix reveals an overall strong performance of the classification with the majority of data points being within the diagonal.

## Feedforward Neural Network

In my experiment with the feed forward neural network architecture on the Fashion MNIST dataset, I evaluated various configurations by altering the number of layers and nodes in each layer. The architecture with 1 layer and 128 nodes achieved the highest test accuracy of 90.28%, while other configurations also performed well but with slightly lower test accuracies.

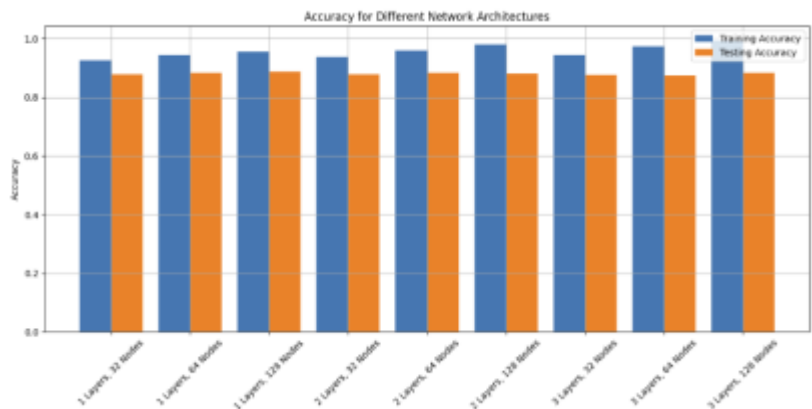


Figure 3.1: Training and Testing accuracies for various architectures

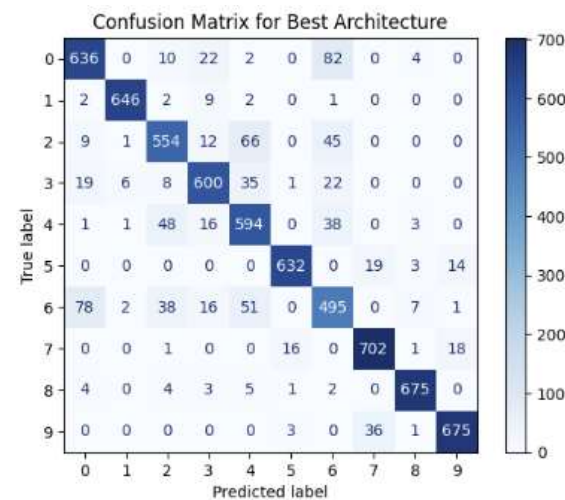


Figure 3.3: Confusion matrix for best performing network with one 128 node layer

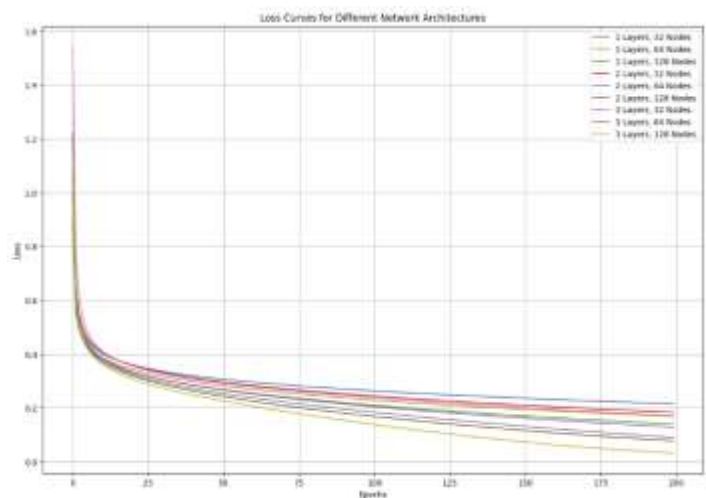


Figure 3.2: Loss Curves for a range of

The corresponding loss curves illustrate that the models converge effectively, with the architecture having 3 layers and 128 nodes showing the fastest convergence to a lower loss value, indicating efficient learning.

Additionally, the confusion matrix for the best architecture (1 layer, 128 nodes) reveals that the model performs well across most categories, though there are some misclassifications, particularly in categories that are visually similar, which is typical in challenging datasets like Fashion MNIST.



## CNN

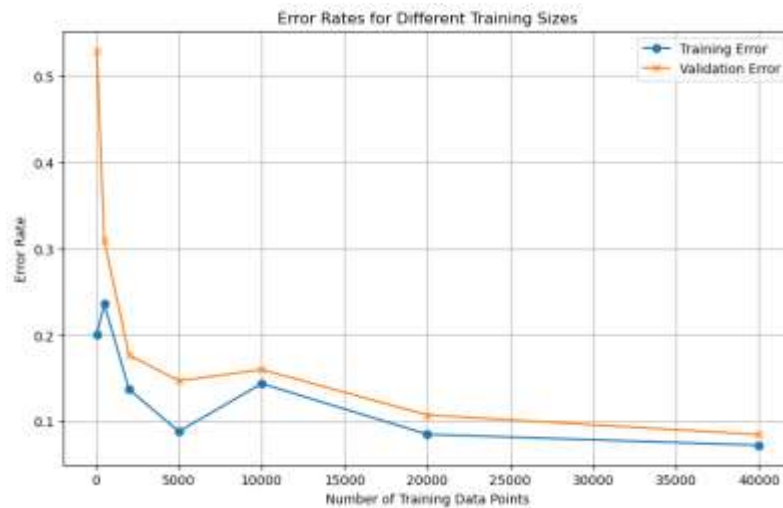


Figure 4.1: Error rate as you increase the amount of training data. As you can see, the validation error started to plateau around 0.1 at ~20,000 training data points

Here are accuracy and loss values achieved from utilizing KFold cross validation with 10 epochs - [Cross Validation Loss per Epoch](#). The mean cross-validation train accuracy was 0.9307. The mean cross-validation validation accuracy was 0.9013. The final test accuracy of my first cross-validated model was 0.8935.

Classifier	# of Layers	# of Nodes (excluding pooling layer)	Test Accuracy (Average)
CNN1	5	16, 32, 128, 10	0.9016
CNN2	6	32, 64, 256, 128, 10	0.9180
CNN3	6	64, 128, 256, 128, 10	0.9104
CNN4	7	32, 64, 128, 256, 128, 10	0.8980
CNN5	7	16, 32, 64, 128, 64, 10	0.8953

Figure 4.2: Test accuracies for Convolutional Neural Network with different #'s of hidden layers and nodes

I also decided to create additional Convolutional Neural Networks where I tried to use different configurations for the convolutional and forward hidden layers. CNN2 gave the best test accuracy of **91.8%**, This network used two convolutional layers of sizes 32 and 64. There were three fully connected (forward) layers of sizes 256, 128, and 10.



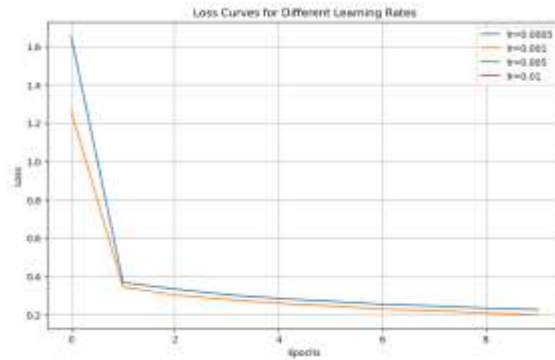
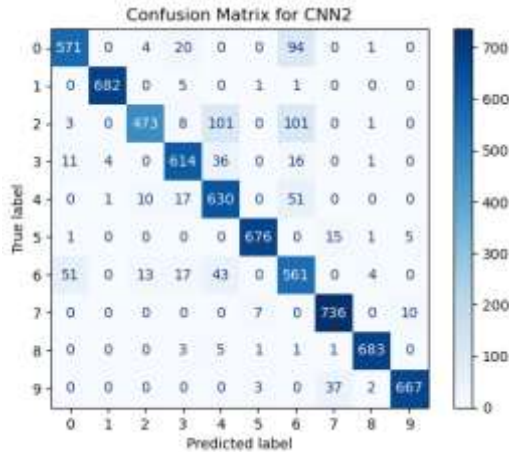


Figure 4.4: Loss curves for different learning rates

The confusion matrix performs similarly well to previous experiments, and the loss curve shows that the lower the learning rate, the higher the loss.

### Final Test Accuracy for each Classifier

k-NN	0.8525
Logistic Regression	0.8577
Feed Forward Neural Network	0.9028
Convolutional Neural Network	0.9180

## Insights

One common observation I found across classifiers is that certain labels were commonly mistaken for other certain labels. For example label 6 was commonly predicted to be label 0. This makes sense as label 6 are “T-shirts” whereas label 0 is “shirts” so they share many of the same features. Interestingly, the best performing Convolutional Neural Networks were significantly better at distinguishing label 0 and 1, with about half the amount of false predictions. This demonstrates its complexity in recognizing specific features. Another common observation was that the higher the value of any parameter, the less likely it is to achieve accuracy. This is evident when the C values in the Logistic Regression classifier were increased, there was a decrease in

accuracy. Similarly, this trend held true for learning rates in CNN and the number of nodes in the Feedforward Network.