An investigation of classification methods for Fashion-MNIST

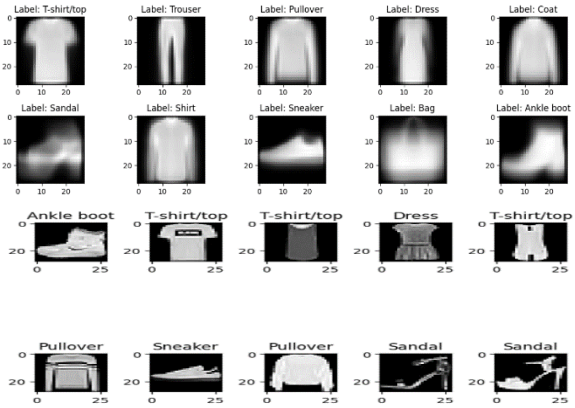
Haodong Zhang - 11045516, Zhe Yuan - 56719356, Yuxue Zhou - 12264536

Summary:

The dataset we used for our project is Fashion-MNIST. The classification methods we used are K-Nearest Neighbors, Logistic Classifier, Neural Network, and Convolutional Neural Network. After finishing training our models, we compared them with each other in terms of accuracy, precision, recall, f1, AUC, learning curve, and training time. In conclusion, Convolutional Neural Network does the best job of classifying the datasets. It has a significant advantage compared to the other models.

Data Description:

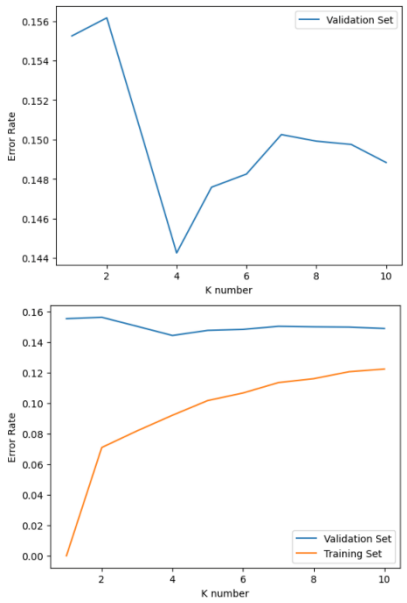
The data set our group chooses is the Fashion-MNIST. It consists of a training set of 60000 examples and a test set of 10000 examples, which has a total of 70000 examples. The size of each example is 28 * 28, which means it has 784 features for each sample. There are 10 classes from 0 to 9 in this data set. 0 is T-shirt/top, 1 is Trouser, 2 is Pullover, 3 is Dress, 4 is Coat, 5 is Sandal, 6 is Shirt, 7 is Sneaker, 8 is Bag, 9 is the Ankle boot. In the training set, each class has 6000 samples, and in the test set, each class has 1000 samples. Here shows the average shape of each class and the first nine samples from the training.



Classifier:

In this project, we have trained for different classifiers using different libraries in Python. KNN Classifier, Logistic Classifier, and Neural Networks from sklearn. Convolutional Neural Networks from PyTorch.

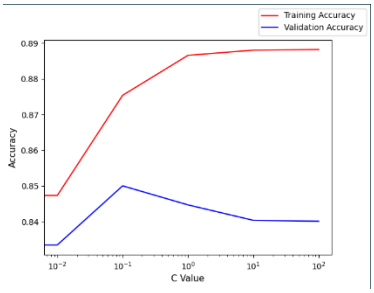
1. The first model is the **K-Nearest Neighbors(kNN)**. How it works is relatively simple, it is using a voting-like technique to make predictions. kNN lists out all the training datasets and finds k number of data that are closest to the predicting data. Then the predicting data would be predicted as the same as the highest appearance of the label among the k data. The hyperparameter in this algorithm is the value of k which indicates the number of neighbors that we need to consider. The higher the k the smoother the decision boundary is. To determine the proper value of k for this dataset. We use the error rate on the validation set. We tried the k value from 1-10 and check on their error rate.



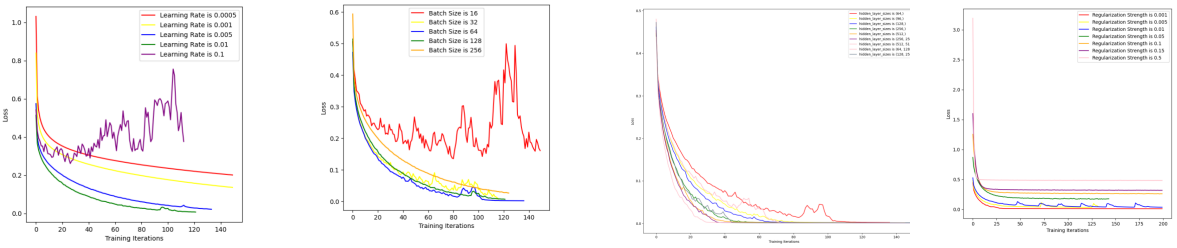
2. From the graph, we can see that in this situation, the proper k value to use is 4 because it has the lowest error rate on the validation set.

1. The second model we used is **Logistic Classifier**. Different from the linear model, the logistic model creates a "sigmoid" to predict the classes. As in the linear model, the logistic model trains the coefficient for each feature. If we have N features, then the logistic model trains (N + 1) coefficients (including the intercept). The logistic model uses the linear function $z(x;\theta) = \theta_0 + \sum_{i=1}^{N} \theta_i x_i$ and transforms the function to a sigmoid function by using $z(x;\theta)$ in $f(x;\theta) = \frac{1}{1+e^{-z(x;\theta)}}$ to create a sigmoid boundary. In a binary case, the output of the sigmoid function will be a class probability of a datapoint to be 1. In case of more than 2 classes (k classes), we have k sigmoid functions to compute k probabilities of a datapoint and use the SoftMax function to normalize the probabilities.
2. The hyperparameters involved in the logistic model are “penalty,” “C,” “solver,” “fit_intercept,” “max_iter,” and “random_state.”

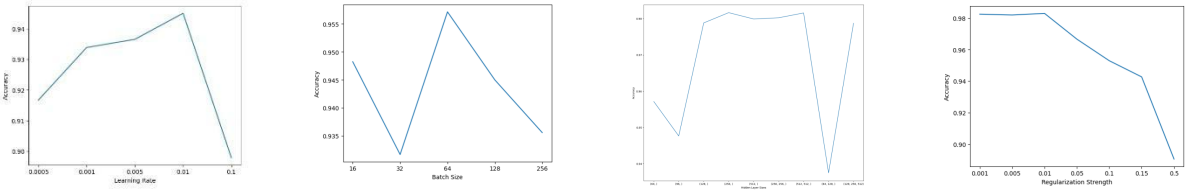
3. In this project, we set the solver = “saga” instead of using other algorithms. Saga has faster speed when training with large data sets, and the data set we used, Fashion Mnist, has 70000 data points with 784 features and 10 class labels. When we choose the saga algorithm, the logistic model will use Stochastic Average Gradient descent to train the coefficients.
4. We have also set the max_iter to 5000 in this project. When we set the max_iter less than 3000, the coefficients cannot be converged because this is a large data set with 785 coefficients that need to be trained. Therefore, we decided to set max_iter to 5000.
5. The main hyperparameter we focused on is the regularization strength (C). This parameter helps us to find an appropriate point between underfit and overfit. We have tried six different values for C, which are 0.01, 0.1, 1, 10, and 100. Because sklearn takes the inverse of C, a smaller value means larger regularization. According to the graph of accuracy (tested on validation data by each model), 0.1 is the best C value which has the best validation accuracy (0.85).



1. The third model we used is the **Neural Network**. The Neural Network is combined with the hidden layers and hidden units and activated values in each layer by the activation function like “ReLU” or “Sigmoid”. Using methods like backpropagation or the Stochastic Gradient Method, the Neural Network can be trained and optimized to fit the requirement of the task needs.
2. The hyperparameters are the Learning Rate with the range of 0.0005, 0.001, 0.005, 0.01, 0.1, Batch Size with the range of 16, 32, 64, 128, 256, Hidden Layer Size with the range in (64,), (96,), (128,), (256,), (512,), (256, 256,), (512, 512), (64, 128,), (128, 256, 512,), Regularization Strength (alpha) with the range in 0.001, 0.005, 0.01, 0.05, 0.1, 0.15, 0.5, and Solver with the values *sgd* and *adam*.
3. After testing the model with a different set of parameters, the model with the best performance has the following parameters: Learning Rate: *0.01*; Batch Size: *64*; Hidden Layer: *(512, 512,)*; alpha: *0.01*; Solver: *sgd*.
4. To choose the best hyperparameter, the method we used to compare all of them is putting them into the same environment, which means all other hyperparameters are the same and then calculating the accuracy in the validation dataset.
5. The Loss curve of parameters: learning rate, batch size, hidden layer size, and regularization strength.



The accuracies in the validation of the same four parameters.

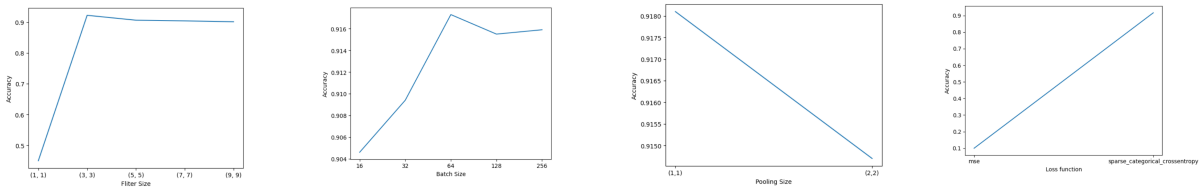


6. By comparing the accuracies between different solve, *sgd* with 0.9877 accuracy and *adam* with 0.953, we pick *sgd* to be the solver in this project.

1. The last model we trained is the **Convolutional Neural Network**. It is an advanced Neural Network that uses filters to get the feature of the sample, especially pictures. After experiencing multiple convolutions, it will get

the feature maps, then it will flat these feature maps and input them to an MLP to get the result. The Hyperparameters are Filter Size with the range in (1,1), (3, 3), (5, 5), (7, 7), (9, 9), Batch size with the range in 16, 32, 64, 128, 256, pooling size with the range in (1, 1), (2, 2), (4, 4), (5, 5), (6, 6), and loss function with the values *mse* and *sparse_categorical_crossentropy*.

2. After testing the model with different sets of parameters, the model with the best performance has the following parameters: Filter Size: *(3, 3)*, Batch Size: 64, Pooling Size: *(2, 2)*, Loss function: *sparse_categorical_crossentropy*.
3. To choose the best hyperparameter, the method we used to compare all of them is putting them into the same environment, which means all other hyperparameters are the same and then calculating the accuracy in the validation dataset.
4. The hyperparameters' accuracy in the validation set



5. When the pooling size is bigger than (2, 2), the picture cannot be reduced in size anymore after some convolutions. Therefore, we choose to calculate the size of (1, 1) and (2, 2). When the size is (1, 1), its accuracy is 0.0034 higher than (2, 2). However, if the pooling size is (1,1), it is equivalent to not using pooling in the model. This will cause the number of variables to grow exponentially because we cannot reduce the size of the feature map through pooling. Therefore, we choose (2,2)
6. Visualizing the filter and the feature map in the first convolution layer

Experimental Setup:

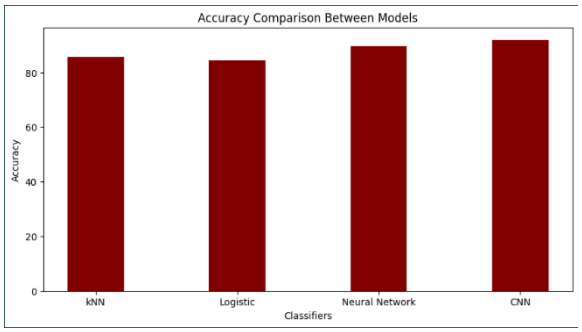
1. We split the 60000 training data into 48000 (80%) for training data and 12000 (20%) for validation data. The remaining 10000 data is used for testing data. The random seed is set to 1234 for reproducibility.
2. In this project, we used an **Accuracy graph**, **confusion matrix**, **precision-recall curve**, **f1 score**, **AP score**, and **ROC curve** to evaluate the classification models.
3. The accuracy graph evaluates the overall performance of different models on the data set using the equation $\frac{T_p+T_n}{T_p+T_n+F_p+f_n}$. The accuracy gives the percentage of the number of labels predicted exactly match the corresponding true label.
4. The confusion matrix is an N * N matrix used for evaluating the performance of a classification model. Where each row is the true label, and each column is the predicted label. By definition, a confusion matrix C is such that $C_{i,j}$ is equal to the number of observations known to be in group *i* and predicted to be in group *j*.
5. Precision means the ability of the classifier not to label as positive sample that is negative and is calculated by $\frac{T_p}{T_p+F_p}$. Recall is the ability of the classifier to find all the positive samples and is calculated by $\frac{T_p}{T_p+F_n}$.
6. In some cases, precision would be more important and in some cases otherwise. However, in this project, precision and recall are equally important because we want to retrieve more positive samples and want the

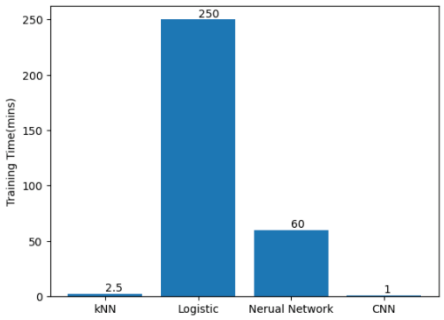
precision to be high enough. In this case, we compute the f1 score for each class label. The f1 score is the harmonic mean of precision and recall, which is an optimal blend of precision and recall, and calculated by $2 * \frac{precision * recall}{precision + recall}$.

7. With the value of precision and recall, we could plot the Precision-recall Curve, which is a direct representation of the recall (x-axis) and precision (y-axis). The area under the ROC curve is called the Area Under Curve (AUC), which evaluates the overall performance of the model. By visualizing the precision-recall curve, we can see how well each model did on both precision and recall.
8. The AP score refers to the Average Precision score. While precision refers to precision at a specific threshold, average precision gives an overall precision at all possible thresholds. It is a way to summarize the precision-recall curve into a single value representing the average of all precisions.
9. With the value of False Positive, number of Negatives, and True negatives, we can compute the False Positive Rate (FPR), which is calculated by $\frac{F_p}{F_p + T_n}$, and True Negative Rate (TNR), which is calculated by $\frac{T_n}{T_n + F_p}$. With FPR and TNR, we can plot the ROC curve (Receiver Operating Characteristic Curve), which illustrates the diagnostic ability of the binary classifier as the threshold varies. The area under the ROC curve (AUC) also evaluates the overall performance of the model, but from a different perspective.

Experimental Results:

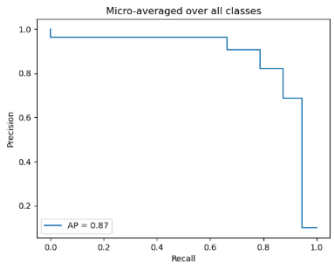
Comparison between each model:

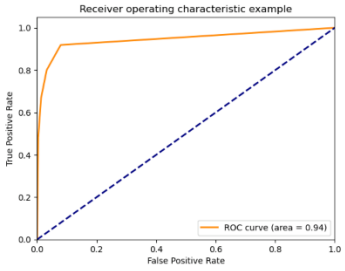




The metrics of each model

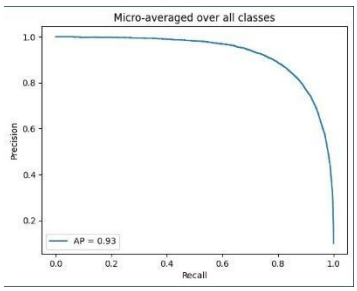
KNN Classifier:

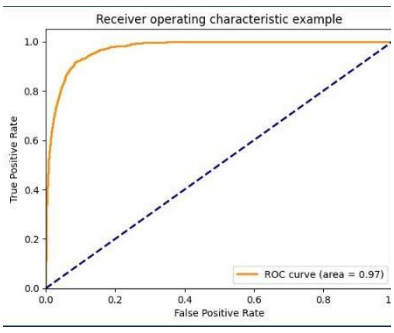




	precision	recall	f1-score	support
0	0.74	0.88	0.80	1000
1	0.99	0.97	0.98	1000
2	0.73	0.80	0.76	1000
3	0.89	0.86	0.88	1000
4	0.77	0.74	0.75	1000
5	0.98	0.87	0.92	1000
6	0.63	0.53	0.58	1000
7	0.88	0.96	0.92	1000
8	0.98	0.92	0.95	1000
9	0.92	0.94	0.93	1000
accuracy			0.85	10000
macro avg	0.85	0.85	0.85	10000
weighted avg	0.85	0.85	0.85	10000

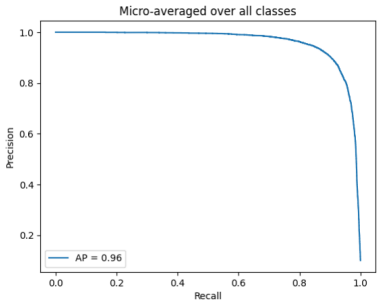
Logistic Classifier

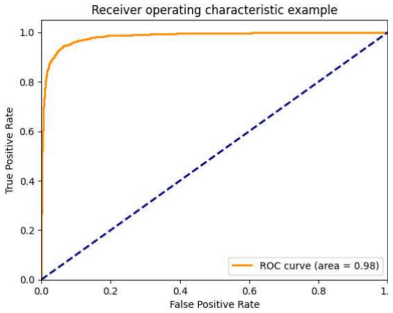




	precision	recall	f1-score	support
0	0.81	0.81	0.81	1000
1	0.98	0.96	0.97	1000
2	0.74	0.72	0.73	1000
3	0.83	0.87	0.85	1000
4	0.73	0.76	0.74	1000
5	0.94	0.92	0.93	1000
6	0.62	0.57	0.59	1000
7	0.91	0.94	0.92	1000
8	0.93	0.94	0.93	1000
9	0.94	0.94	0.94	1000
accuracy			0.84	10000
macro avg	0.84	0.84	0.84	10000
weighted avg	0.84	0.84	0.84	10000

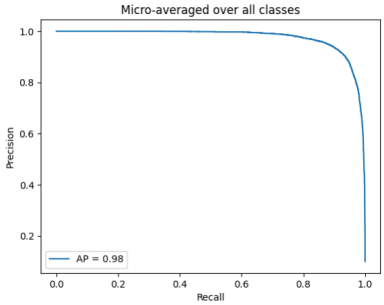
Neural Network Classifier

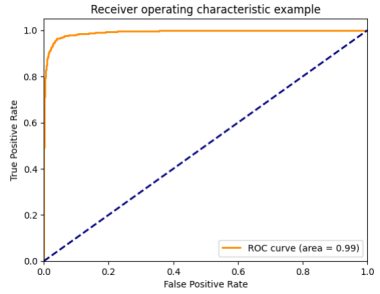




	precision	recall	f1-score	support
0	0.84	0.84	0.84	1000
1	0.99	0.97	0.98	1000
2	0.81	0.83	0.82	1000
3	0.89	0.90	0.90	1000
4	0.82	0.82	0.82	1000
5	0.97	0.96	0.97	1000
6	0.72	0.70	0.71	1000
7	0.95	0.96	0.96	1000
8	0.97	0.96	0.96	1000
9	0.96	0.96	0.96	1000
accuracy			0.89	10000
macro avg	0.89	0.89	0.89	10000
weighted avg	0.89	0.89	0.89	10000

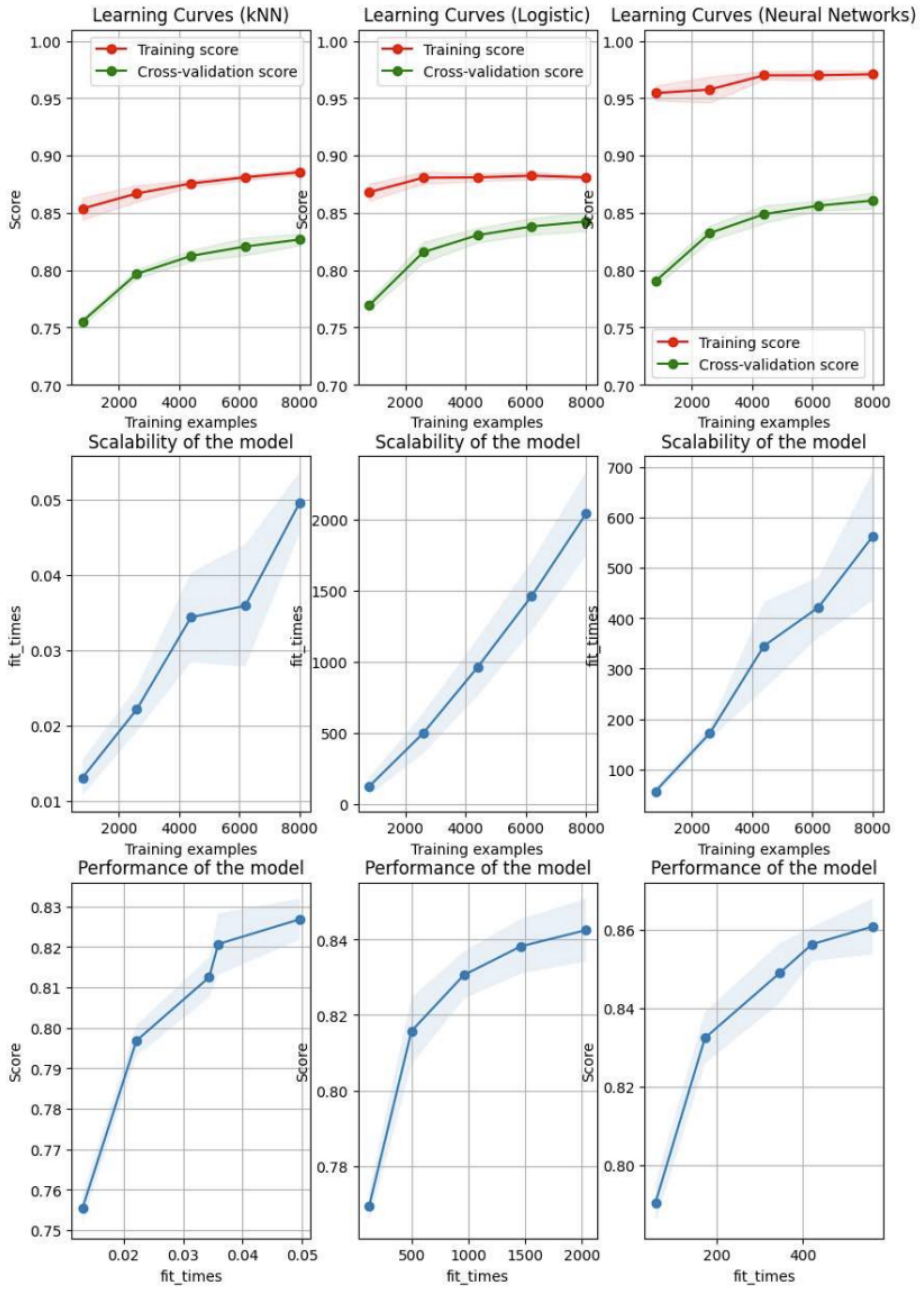
CNN





	precision	recall	f1-score	support
0	0.86	0.90	0.88	1000
1	0.97	0.99	0.98	1000
2	0.92	0.85	0.88	1000
3	0.94	0.89	0.91	1000
4	0.85	0.90	0.87	1000
5	0.98	0.98	0.98	1000
6	0.76	0.78	0.77	1000
7	0.95	0.99	0.97	1000
8	0.99	0.98	0.98	1000
9	0.99	0.95	0.97	1000
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

Learning Curve Graph



According to the learning curve graphs (included in the appendix), the kNN classifier has an increasing tendency of the score as the training examples increase. Whereas the logistic classifier stops increasing after roughly 3000 examples and Neural Network stops after roughly 4000 examples. According to the scalability of each model (included in the appendix), the kNN classifier has the shortest overall fit times, and the logistic classifier has the longest overall fit times. According to the performance curves (included in the appendix), the kNN classifier has an accuracy score of 0.83 at 0.05 fit time, and the Neural Network has an accuracy score of 0.86 at 500 fit times. Although the logistic classifier has the final accuracy score of 0.84, it needs 2000 fit times.

In summary, CNN performs the best from all metrics with the highest value and shortest training time. The Logistic Classifier performs the worst with the longest training time. All of the models have poor performance in classifying class 6 compared to other classes due to the similarity between classes 0, 2, 4, and 6. Therefore, the error of the Logistic Classifier must include the part of the error of CNN.

Insights:

1. The most impressive result from this training is that CNN has the best accuracy result with the shortest training time (about 1 min). There are two reasons we analyzed why CNN is extremely excellent in processing the image dataset. The first one is its inductive bias, and the second one is its unique structure.
2. *Inductive Bias:* It is a set of (explicit or implicit) assumptions made by a learning algorithm in order to perform induction. To CNN, the induction is *Locality* and *Spatial Invariance/Translation Equivariance*. The locality is that neighboring pixels tend to be correlated, while faraway pixels are usually not correlated. In reality, similar things will go together and the different things will go far away. This gives the CNN ability to catch the feature quickly of an image dataset. Spatial Invariance/Translation Equivariance is applying the same filter bank F to input patches at all locations which give the possibility for the CNN to use the same filter to train the whole image.
3. *Unique Structure:* Because of the inductive Bias of CNN, the structure of CNN can get the best effect itself.
4. Compare to CNN, the Logistic Classifier does not have such an advantage. It is trained the longest time (about 250 mins) to converge and gets the lowest accuracy. After analyzing, the first reason is that we do not scaler the data the first time. However, the most important reason we think is that Logistical Classifier does not have the ability to catch features.
5. In the data set, class 6 is most likely misclassified as class 0, 2, or 4 by all models. This is reasonable since T-shit, pullovers, and Coats always looked like shirts. Someone who does not shop a lot cannot distinguish them very well.
6. In the real world, KNN and Logistic Classifier will not perform well since everything in the real world has color, which triples the number of features with RGB channels. However, CNN will do very well because of the reasons analyzed above.

Appendix:

CS178 Final Project Appendix