# Final Project Report

Group 9

| Model | Best Blended | Training Performance | Validation | Kaggle Data |
|---|---|---|---|---|
| Neural Network | + CNN | 7 mins 40 secs | 98% | 95.1% |
| k-Nearest Neighbors | + PCA | 1.4704 secs | 96% | 91.4% |
| | + K-Means | 0.1894 secs | 84% | |
| Logistic Regression | | 19 mins 11 secs | 80% | |
| | + PCA | 4 mins 50 secs | 82% | 70% |
| Stochastic Gradient Descent | | 4 mins 13 secs | 73% | |
| | + PCA | 1 mins 14 secs | 76% | |

**Description**

➢ *k-Nearest Neighbors + PCA*

Using all samples from X_train, the data first get processed through PCA for dimension reduction. In order to retained 95% of the variance for the data, n_components is set to 0.95 and svd_solver is set to ‘auto’ such that PCA will automatically give us the eigenvectors. The transformed data then get split into 60% training and 40% testing. Training data gets fit to the kNN classification using 10 neighbors and minkowski distance scaling with the implementations of weighted voting and tree-based neighbor. Minkowski distance improve the accuracy of the algorithm for large dataset and k-d trees help with a faster processing time. Weighted voting is implement for the reason of wanting closer neighbors to have a greater impact in comparison to neighbors further away.

➢ *k-Nearest Neighbors + K-Means*

All samples of X_train are processed through the K-Means clustering algorithm as a way to reduce the dimension and to clean out outliers. The dataset is divided into 12 clusters with a maximum iteration of 5000 and gets transform to a cluster-distance space aftering fitting. After splitting the data into 70% training and 30% testing, it can then be trained on the kNN classification using 7 neighbors and minkowski distance scaling with the implementations of weighted voting and tree-based neighbor (same reasons as above).

➢ *Stochastic Gradient Descent*

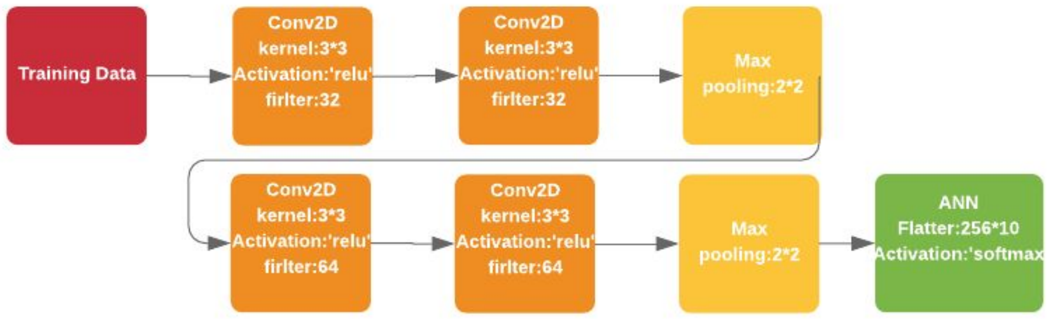
This model is implemented with sklearn linear classifiers with SGD training. Without reducing the dimension and splitting the whole X_train datasets into 80% training and 20% testing, we can get an accuracy of 0.7251 by choosing alpha to be 1e-3 with a maximum iteration of 5000 and tolerance of 1e-5. The regularization term is set to default using L2 as changing it does not improve the model as we would like. Executing PCA prior to the fitting does not increase the score by a lot, but it should be noted that the implement of PCA helps reduce data fitting time by 75%.

➢ *Logistic Regression*

We naively initiate the experiment by linear classifier. Since the input and output variable are high dimensional data, the boundary for each class maybe highly non-linear. Therefore, the linear classifier such as logistic regression may not be a ideal solver for this problem. Based on our test, we randomly shuffled the training set and slip into 0.6 for training and 0.4 for testing. With PCA we found the training time significantly reduce and performance increase.

➢ *CNN*

Convolutional neural network(CNN) provides unique approach for segmentation and object detection, and achieves highest R squares scores among all the algorithms we experimented. We implement our simple CNN by python library Keras and Tensorflow.  To reduce the dimension and capture more characteristic detail from each image. Since for Kuzushiji-MNIST dataset there are 10 categories so we fully-connected neural network (which is also Artificial Neural Network) by softmax activation function to classify it. Besides, for Kuzushiji-49 we should change ANN's output layer from 10 neurons to 49 neurons since it has 49 classes. Eventually we achieve 98% accuracy for cross-validation and 95% accuracy on leaderboard.


Training Data
Conv2D kernel:3*3 Activation:'relu' firlter:32
Conv2D kernel:3*3 Activation:'relu' firlter:32
Max pooling:2*2
Conv2D kernel:3*3 Activation:'relu' firlter:64
Conv2D kernel:3*3 Activation:'relu' firlter:64
Max pooling:2*2
ANN Flatter:256*10 Activation:'softmax

**Conclusion**

Different types of machine learning algorithms all have different pros and cons. For this specific problem, a convolutional neural network is a great option but also comes with the downside of needing more computing power and the black box nature. KNN, especially while using PCA as a means to preprocess the data, is not quite as accurate as the CNN but is much faster and easier to understand how it is achieving the result. SGD doesn't work quite as well due to the size of the data and reducing the dimension using PCA results in a better but still lacking compared to KNN or a Neural Network. And finally Logistic Regression did a little better at dealing with the size of the data than the SGD however it took a very long time for a not so accurate result and is not very good for this application. Overall the best is definitely the Neural Network and a Convolutional Neural Network specifically is great for NLP applications such as this.