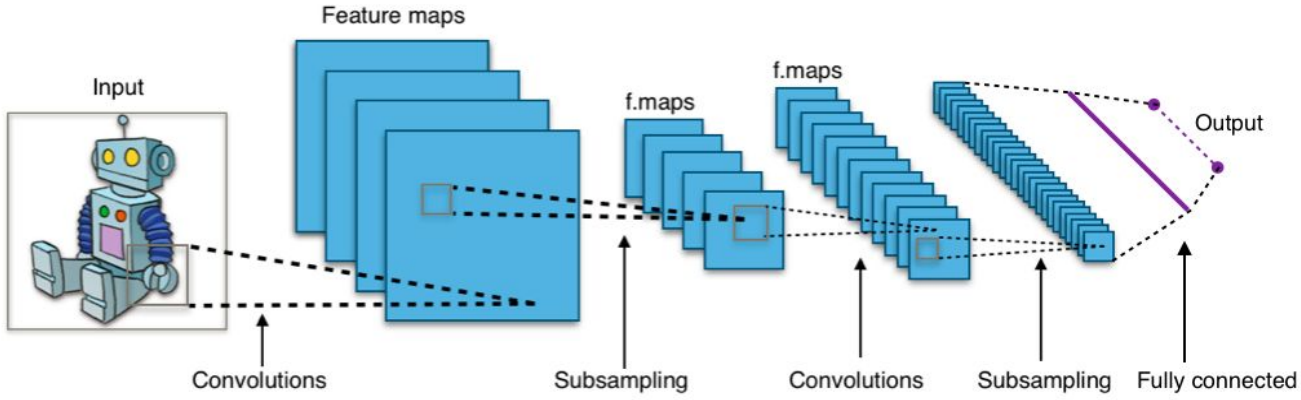
# Fashion MNIST Report

**Group Member: Weizi Zeng, Shuyi Ye, and Junhao Zhang**

## Model Performance

| Algorithm | Accuracy (training data ) | Accuracy(validation) | Accuracy (leaderboard) |
|---|---|---|---|
| CNN(epoch = 15, dense= 15) | 0.972 | 0.96625 | 0.91207 |
| CNN(epoch = 10) | 0.9752777778 | 0.9046428571 | 0.90885 |
| CNN(epoch = 1) | 0.8651587302 | 0.8576785714 | 0.87771 |
| KNN+PCA(n_component=0.95) | 0.9028809524 | 0.8477619048 | 0.85942 |
| Softmax Regression | 0.6278095238 | 0.820595238 | 0.8335 |

## CNN




For the convolutional neural network, we still use original data as raw inputs. The first step is data manipulation. We use the built-in function train_test_split to split X_train into X_train* and X_test*, y_train into y_train* and y_test*, with test_size=0.4. Then, we reshape the size of X_train* from (25200,784) to (25200, 28, 28, 1) and the size of X_test* from (16800, 784) to (16800, 28, 28, 1). After that we transform all the data in X_train* and X_test* to float 32 type and divide each of them by 255 to make it smaller. Then, we implement the CNN algorithm and fit in X_train* and y_train to train the model. After all those setups, we compute the distance between X_train* and X_test* and label it as y_pred_test* using built-in function predict. Using it to compare with y_test* by checking the same labels to get the cross-validation accuracy. We use the same way to get the prediction on training data and final y_prediction for X_test.

For the CNN model, the key parameter is the number of epochs. We first choose one as the epoch number and get the performance on training and validation. Both performances are less than 90 percent and the final accuracy in the leaderboard is also less than 90 percent. Then we guess that increasing the number of epochs would increase the accuracy, so we change the

epoch number to ten and get the performances. Compared with all the performances we get when epoch is one, increased epoch number does help increase the accuracy.

**KNN+PCA**

We use raw inputs to implement K-Nearest-Neighbor with PCA. First, we use the built-in function train_test_split to split X_train into X_train* and X_test*, y_train into y_train* and y_test*, with test_size=0.5. Then, apply PCA(n_components=0.95) to reduce dimensions in order to speed up the learning algorithm. Since the Fashion-MNIST data set is large, we choose vectorization of KNN, which performance is better than the built-in KNN function(using for loop). After those setups, compute the distance between X_test* and X_train*, and choose 5 nearest neighbors to vote. Label it to create y_pred_test*and then use it to compare with y_test* by checking the number of same labels in order to get the accuracy. Last, we get the final predict solution for X_test in the same way.

The thought of combining KNN and PCA is to accelerate the process. PCA can discard some features that do not have a strong relationship with labels, which can reduce much computing work in KNN.

**Softmax Regression**

When we applied the Softmax Regression, we import the Logistic Regression model. We still use the original data to apply the model. Since there are multiple features, which means it is a multi-class case and we change the multi_class to 'multinomial' to use the cross-entropy loss. We set the model as LogisticRegression(random_state=0, solver='lbfgs', tons =1e-5, max_iter=2000, verbose=True, muti_class='multinomial'). We applied the model to the training data and it takes us 3.8 min to fit the model. Then we use X_test to find y_pred, which is the output of the model. Then we split the original x,y training data into 2 training and 2 testing data (X_train1,X_test1,y_train1,y_test1) with test_size=0.4, random state =0 to do the cross-validation . By checking the accuracy between the y_pred1 predicted data which is from the model of training data X_train1,y_train1, and the y_test1 data, we get an accuracy about 0.820595.

Softmax Regression algorithm is the first algorithm we learned in class which can use multiple features and linear regression model to make the prediction. As the time of Softmax Regression show, it spends a long time to predict. Since we have a lot of data in this data, the process will be very slow as the KNN model. And the accuracy score also showed that the accuracy of Softmax Regression is very low, because it is a simple linear regression model which is insufficient to predict the final project.

## Conclusion

At the beginning of this project, we thought CNN should performance best because of its high accuracy, and it is known for image classification. However, we did not choose CNN first since we have limited knowledge about CNN. We began with Softmax Regression and KNN. The outcome showed that they were not good enough to do these complex classification. Especially for Softmax Regression, it is a linear model, which indicates that it is apparently not sufficient for this data set. Fashion-MNIST cannot be linear, so it worked poorly. For KNN, it worked okay, but it was still not good enough. It might because the data set is not perfectly divided. Then, we turned to CNN. We find the Convolution Neural Network worked best among those three algorithms. Even though we get 91% accuracy, the model is still needed to improve. Before that, we all need to learn how exactly CNN work, and then we can refine it with better key hyperparameters for this data set.