Sammy Pardes
IST 718
Week Lab 9
9/5/2021

Introduction

Image recognition is a technique that can be applied to nearly every industry. In retail and fashion, companies are popping up that could greatly benefit from accurate image recognition models. StitchFix and Trunk Club are two businesses that come to mind. These companies provide customized boxes of clothing to their customers based on their individual preferences, obtained from showing their clients various images of clothing. With accurate and efficient models, these companies could leverage image recognition on articles of clothing to provide their customers with clothes aligned with their styles.

About the Data and Data Cleaning

The fashion data set was obtained from keras.datasets by importing fashion_mnist.

This data set contained 70,000 images of various articles of clothing and their numeric labels. Each observation was a 28x28 pixel image matrix of a piece of clothing. Within each matrix were 784 numeric values representing the color of each pixel in the given image. There were 10 different categories of clothing in the entire data set. Below is a sample of 60 images pulled from the fashion data.

Using the load_data() function, the images were split into testing and training data sets. The training data contained 60,000 images while the testing data encapsulated the remaining 10,000.

The images were then flattened from 28x28 matrices to arrays of 784 pixels using the reshape() function. One the images were flat, they were normalized by dividing by the total number of pixel values (255).

Modeling

MLP Neural Network

The first approach I took to classify the fashion data was using the neural network MLPClassifier from the sklearn package. The parameters for this model included hidden_layer_sizes = 150, max_iter = 1000, alpha = 0.0001, verbose = 10, tol = 0.0001, random_state = 1, and learning_rate_init = 0.01. For this first attempt, I set “adam” as the solver

This model took approximately 12 minutes to run and yielded an accuracy of 86.8% on the testing data.

I tried another version of the MLP model by changing the solver to “sgd”. Doing this took a bit longer at around 16 minutes. Although the accuracy on the training data was 99.975%, using SGD improved the testing accuracy to 88.69%.

Random Forest

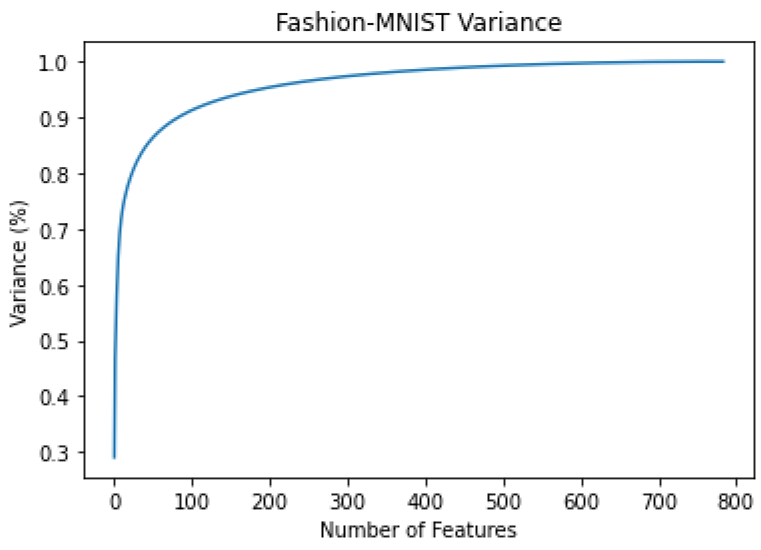
Next, I tried classifying the images using a RandomForestClassifier. I started with a small number of estimators and worked my way up to n_estimators = 80.

This model took just under one minute to run. The output of running this on the training data was 99.99% accuracy. The random forest was 87.81% accurate on the testing data.

SVM

I decided to use Principal Component Analysis in combination with SVM and KNN classifiers. I did this to reduce the dimensionality of the image data and, hopefully, speed up the computational time without diminishing the accuracy of the models.

I plotted the explained variance ratio to determine how many components to use. Using the graph below, I set the number of components equal to 400 to preserve close to 100% of the explained variance in the data.



After fitting the model on the training data, I transformed the training and testing images with the pca.transform(). The result of running the trained SVM classifier with svm.SVC() on the testing data was 88.73% accuracy. This took 7.3536 minutes to complete.

KNN

I also took advantage of the PCA transformation to run a KNN classifier. I utilized the KNeighborsClassifier() function with the number of neighbors set to 5. This took just over 13 minutes to run. The accuracy of the model run on the testing data was 85.54%.

Keras

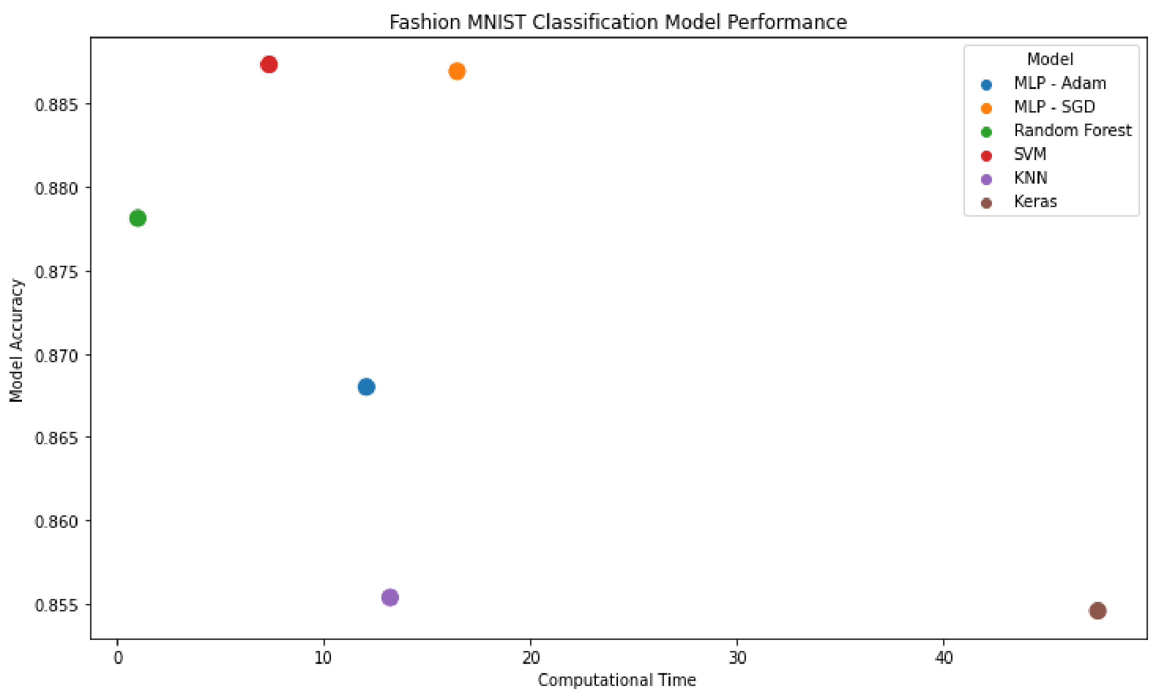
Finally, I attempted to create a TensorFlow Keras model function. I was having some difficulty getting this to run until I set the loss = “sparse_categorical_crossentropy”. Again, I used SGD as the optimizer. This classifier took quite a long time to run at a whopping 47 minutes! The accuracy was not particularly high, scoring 85.46%.

<u>Results and Conclusion</u>

One trade-off for MLP and Keras methods is that there are a lot of parameters that can affect the performance of these models. They each take minutes to run and tuning the parameters to achieve the greatest accuracy possible is time consuming.

In contrast, the lack of parameters in the Random Forest, KNN, and SVM models make them easy to implement, adjust, and re-run.

Below is a scatterplot of each model created, displaying accuracy compared to computational time.



I expect there are additional parameters that could be added to the TF Keras model that would significantly improve the performance, however, it takes a significant amount of time for each iteration. Random Forest, on the other hand, was very speedy and highly accurate. SVM was also relatively quick to run and was the most accurate.

Out of all these methods, my recommendation would be to use the SVM model because it has the highest accuracy (88.73%) and only takes the second longest amount of time (less than 10 minutes) to run against the fashion image data.

While I would have liked to explore more variations of each classifier with adjusted parameters to gain more accuracy, I think these methods are acceptable for classifying the clothing images. Given more time and computational power, it would be interesting to see if it's possible to break 90% accuracy with more finely-tuned parameters.