Introduction

The MNIST hand-written digits classification dataset has become the "Hello, World!" of computer vision. As research in the field has progressed, classification of the dataset has reached near-human performance levels. Because the performance of machine learning on the original dataset has improved to the point where humans do not achieve a much better score than modern algorithms, a new image classification dataset was created to challenge new research: the fashion MNIST dataset.

The fashion MNIST dataset includes images and labels for 60,000 training images and 10,000 testing images in the same 28x28 greyscale image format in which the original MNIST hand-written dataset was provided. Like the original dataset, there are ten different classes each of the images fall under, except each of the class labels are a different article of clothing.

Using the fashion MNIST dataset, I will attempt to train four different classifiers to classify the different articles of clothing in the dataset. The algorithms will be evaluated for accuracy, and compute performance. After all the models have been evaluated, the tradeoffs of the classification methods will be discussed.

Data

The dataset contains 60,000 labeled training images, and 10,000 labeled testing images, each of which are greyscale, and have an image size of 28x28. The labels for the dataset are provided as a NumPy array with a length of 10. The ten different indices corresponded with the ten different classes the image could be labeled in. The indices of the label and their corresponding class name is provided in Table 1.

Table 1 Classes for Fashion MNIST

Index	Class Name
0	T-shirt/Top
1	Trouser
2	Pullover
3	Dress
4	Coat
5	Sandal
6	Shirt
7	Sneaker
8	Bag
9	Ankle Boot

From the labels, I could tell that the classification models could have some trouble distinguishing between a few of the classes. For example, the pullover and coat classes could have many similar features. The dataset also has three different classes for three different kinds of shoes, which could cause some confusion between the three classes. Finally, the difference

between a t-shirt/top and a shirt could be minimal and the classifier may have trouble misclassifying between those two classes.

With the classes I figured to be difficult to classify in mind, I wanted to see what the images looked like to the classifier. To do this, I reshaped a 1x784 NumPy array into a 28x28 NumPy array and plotted the image with the matplotlib function imshow. The resulting images for each of the classes are provided in Figure 1.

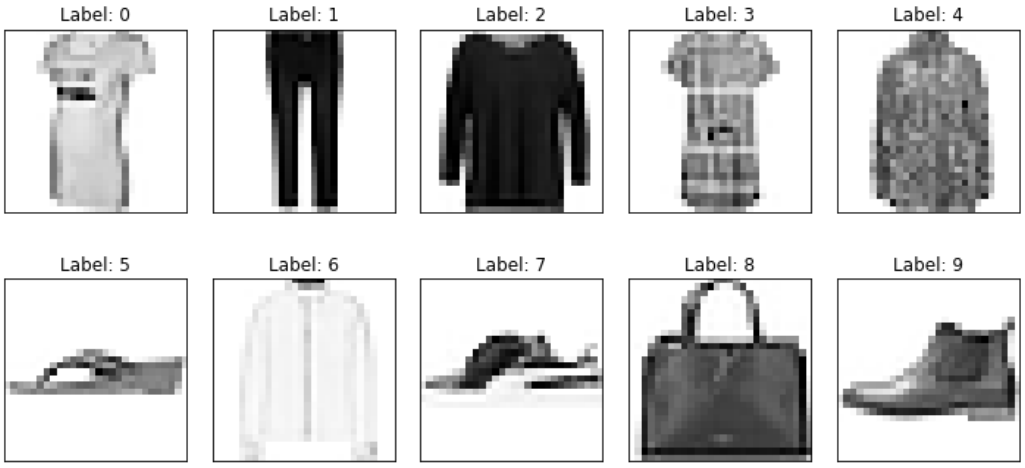

Figure 1 Labeled Examples for Each Fashion Class

The images provided in figure 1 provide some insight to the potential performance of the different algorithms on the training and test data. The first issue I noted was how blurry the images are, especially compared to the images of the hand-written digits. This is likely due to the reduction in detail that occurs when converting a full-sized image to an image of size 28x28. I also noted how similar the shirt (label 6) classification is to the pullover (label 2). While the differences may be more apparent with more detail in the original images, the reduced quality in the provided images make this classification task difficult for the human eye, which would also make it more difficult for a computer to classify.

Modeling

Four different models were trained on the images in the training set to classify the different images in the test set. The models were a random forest classifier, a single layer neural network, a deep neural network, and a convolutional neural network. The random forest classifier was trained using the algorithm provided in the sklearn package, and the neural networks were all created with TensorFlow.

Random Forest

The random forest classifier was first tuned with different hyperparameters on an 80-20% split of the training data. After multiple models were trained, it was determined that the best parameter setting had 100 estimators, a max tree depth of 100, and used entropy as the criterion to split on. The model took four minutes and twenty seconds in total CPU time (23.2s

wall time) to train the models, and resulted in an accuracy of 0.8128 (in 2.73s CPU time, 553 ms wall time) on the final test set. To get a better idea of the performance of the model on the test set, I displayed the first ten misclassification in the test set. The plot created is provided in Figure 2.

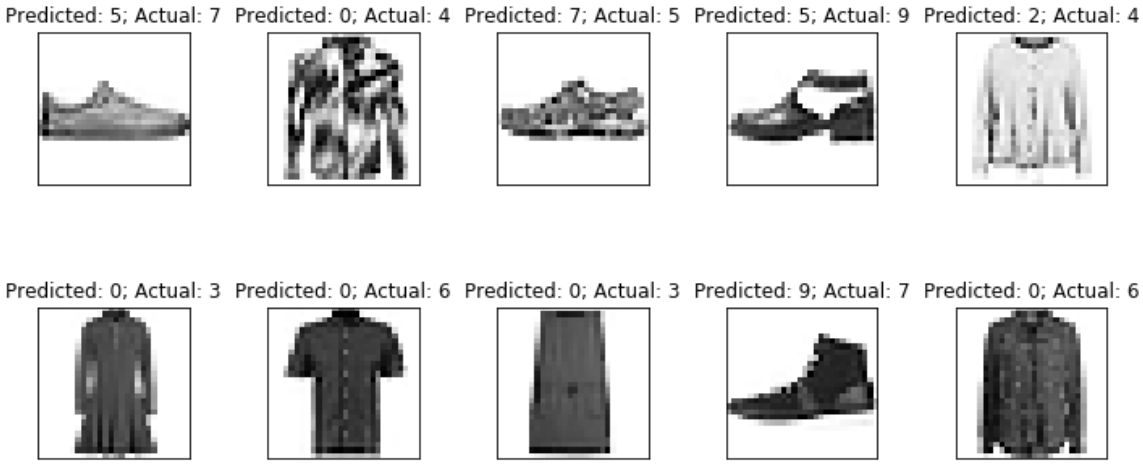

Figure 2 Misclassifications of Random Forest Classifier

Of the first ten misclassifications there were only three misclassifications made by the classifier that I personally would have been able to classify relatively easily: The first two shoe misclassifications, and the misclassification of a dress as a t-shirt/top. To see if there were any trends in the misclassification of certain classes with others, I created a confusion matrix showing how the actual class vs. predicted class worked within the random forest classifier. The confusion matrix I created is provided in Table 2.

Table 2 Random Forest Confusion Matrix

		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	960	0	1	12	1	0	22	0	4	0
	1	43	943	0	11	2	0	0	0	1	0
	2	245	0	674	6	64	0	11	0	0	0
	3	147	1	0	840	12	0	0	0	0	0
	4	203	0	49	17	705	0	26	0	0	0
	5	31	0	0	0	0	937	0	25	0	7
	6	452	0	60	15	42	0	422	0	9	0
	7	31	0	0	0	0	4	0	937	0	28
	8	43	0	0	1	1	0	0	3	952	0
	9	30	0	0	0	0	3	0	33	0	934

The confusion matrix provided in Table 2 shows that if the random forest classifier misclassified a fashion item, it likely classified it as a t-shirt/top instead of the actual class of the item. This was especially the case when an image was actually depicting a shirt. This made sense to me as the difference between a shirt and a t-shirt/top can be almost arbitrary. However, this "feature" of the model created provided little certainty that an image was a t-shirt/top if the model predicted that the image was a t-shirt/top, as class 0 had a precision of 0.439.

Single Layer Neural Network

The next classification algorithm I created was a neural network with a single hidden layer made with TensorFlow. The neural network had one layer with ten nodes, which used the softmax activation function to predict which class an image belonged to. To train the neural network, I created a cost function of cross entropy, and used the Adam Optimizer (with a learning rate of 0.0001) to minimize the cross entropy of the batches passed to the neural network. I trained the neural network for 20,000 epochs and a batch size of 50 and scored an accuracy of 0.834 on the test set. The single layer neural network took 16 seconds of CPU time to train, and 108 CPU ms to predict the test set.

Additionally, a learning curve was created for the training of the single layer neural network to monitor convergence of the cost function. The learning curve for the single layer neural network is provided in Figure 3. The learning curve shows that the neural network converged around epoch 10,000 and that the validation curve remains pretty steady with the training curve. This means that the neural network did a good job of not overfitting to the training data, and provided an appropriate representation of all the data. The variation on the training curve is due to the smaller batch size during training (50), compared to the batch size when evaluating the validation curve (10,000).

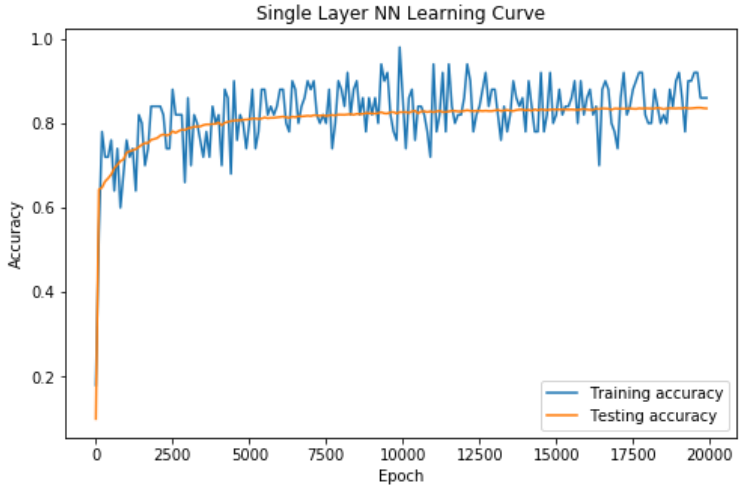

Figure 3 Learning Curve for Single Layer Neural Network

To get an idea of which articles of clothing the neural network was not able to classify, I once again looked at the first ten misclassified images in the test set. The misclassified images, the prediction, and the actual class of the image is provided in Figure 4.

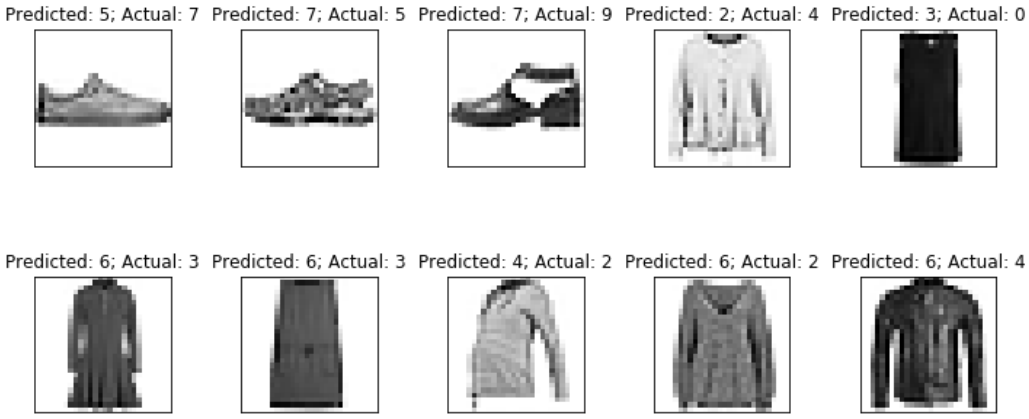

Figure 4 Misclassifications of Single Layer Neural Network

The first ten misclassifications of the neural network showed I would have had difficulty classifying some of the articles of clothing myself. The image shown in the fifth frame (predicted dress, actual t-shirt/top) also shows the difficulty of differentiating between the white background and white on the article of clothing (specifically the sleeves). To see where the common misclassifications occurred, I created a confusion matrix for the single layer neural network, which is provided in Table 3.

Table 3 Single Layer Neural Network Confusion Matrix

		Prediction									
		0	1	2	3	4	5	6	7	8	9
Actual	0	825	5	14	48	7	0	84	0	17	0
	1	5	945	10	31	5	0	2	0	2	0
	2	20	4	734	8	156	1	69	0	8	0
	3	30	12	14	867	36	0	37	0	4	0
	4	0	1	116	39	775	0	63	0	6	0
	5	1	0	0	1	0	889	0	64	4	41
	6	155	3	140	45	127	0	505	0	25	0
	7	0	0	0	0	0	32	0	933	0	35
	8	2	1	7	10	3	7	21	6	943	0
	9	0	0	0	0	0	12	1	46	1	940

The confusion matrix for the single layer neural network not only shows a more accurate classifier, but also a less biased classifier by not typically choosing class 0 (t-shirt/top) as the class, but the classifier generally misclassifies fairly evenly around the classes. It is definitely worth noting that the classifier is still not very good at correctly classifying shirts as shirts.

Deep Neural Network

To see how much of difference depth in a neural network makes in terms of accuracy, I created a deep neural network with four hidden layers. The layers of the deep neural network had node sizes of 1024, 512, 128, and 10. I set the activation function of the first three layers to be the ReLU function, with the fourth layer having a softmax activation function. The cost function was once again set to be cross entropy, and the cost was minimized with the Adam optimizer. The model was trained for 20,000 epochs, with a batch size of 50; it also took fifteen minutes and 19 seconds in total CPU time to train the model, and 1.32 CPU seconds to predict the test set, in which it scored an accuracy of 0.891.

Another learning curve was created for the deep neural network, and is provided in Figure 5. The learning curve showed some expected attributes of neural networks that the single layer neural network did not show, such as the higher performance in the training set compared to the validation set. The elbow of the validation curve for the deep neural network was also much sharper than that of the single layer neural network, meaning that the accuracy of the deep neural network improved much quicker than that of the single layer neural network. Finally, the learning curve shows that the neural network did converge, but gradual increases to the model's performance continued until approximately epoch 13,000.

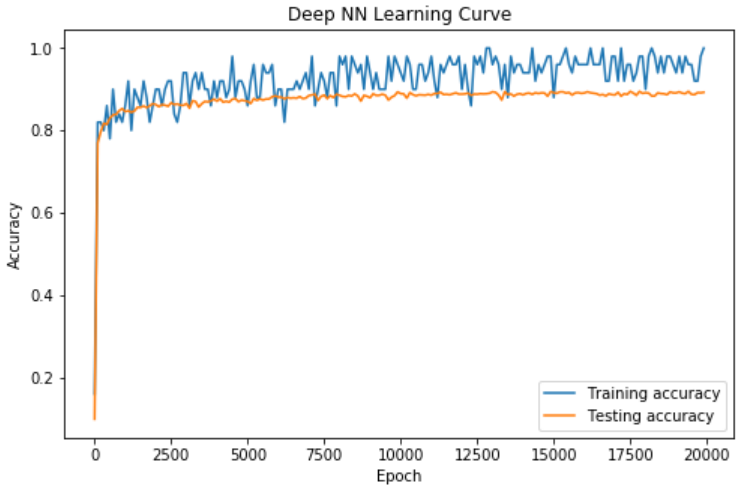

Figure 5 Learning Curve for Deep Neural Network

As I had done with the previous classifiers, I plotted the first ten misclassifications of the test set to see any obvious errors, and to compare which articles of clothing the deep neural network was able to classify that the single layer neural network was not able to classify. The first ten misclassifications are provided in Figure 6.

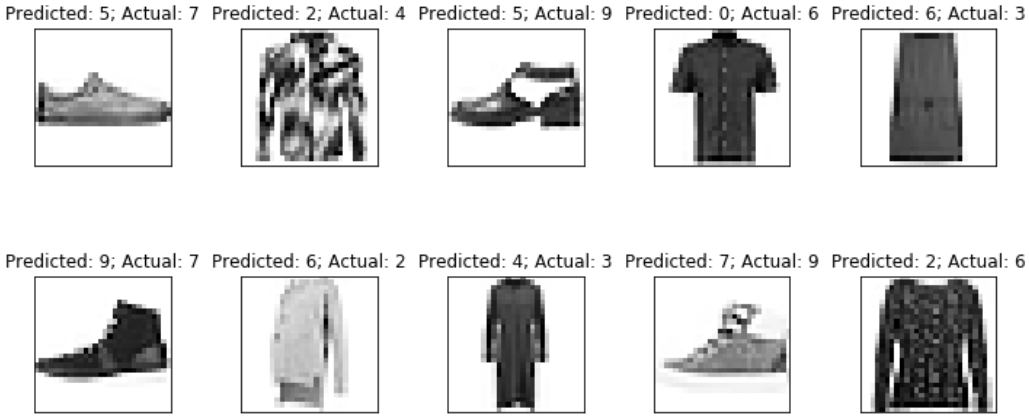

Figure 6 Misclassifications of Deep Neural Network

The deep neural network's first ten misclassifications had five of the same misclassifications that the first 10 misclassifications of the single layer neural network. The brief look at the misclassifications shows that the deep neural network is good at determining that a shoe is a shoe, but not necessarily great at choosing what kind of shoe an image is. A confusion matrix of the deep neural networks classifications was also created, and is provided in Table 4.

Table 4 Confusion Matrix for Deep Neural Network

		Predictions									
		0	1	2	3	4	5	6	7	8	9
Actual	0	889	1	16	13	4	2	70	0	5	0
	1	4	976	1	12	4	0	2	0	1	0
	2	18	1	830	12	75	0	60	1	3	0
	3	23	7	16	885	40	0	24	0	5	0
	4	1	0	90	25	844	0	39	0	1	0
	5	0	0	0	1	0	945	0	41	1	12
	6	151	2	89	21	78	0	653	0	6	0
	7	0	0	0	0	0	5	0	979	0	16
	8	6	0	6	3	0	4	4	5	972	0
	9	0	0	0	0	0	8	1	50	0	941

The confusion matrix showed that the improvements of the classifier across the board, as every class had more correct predictions in the deep neural network than the single layer neural network. The confusion matrix also showed that the classifier is still not great at correctly predicting articles of clothing in the "shirts" category as a shirt.

Convolutional Neural Network

The final method I used to classify articles of clothing in the fashion MNIST dataset was with a convolutional neural network (CNN). The convolutional neural network first goes through a convolutional layer with 32, 5x5 convolutional kernels and a 1x1 stride, followed by a ReLU activation function and a max pooling of size 2x2 with a 2x2 stride. This convolution, activation, max pooling process layer was applied again, but with 64 convolutional kernels. Then the output of the two convolutions were flattened and fully connected to three hidden layers with 1024, 512, and 10 nodes. The hidden layers with the 1024 and 512 nodes had the ReLU activation function applied to them, and the final layer was activated by the softmax function.

After the architecture of the CNN was made, the cost function was once again made to be the cross entropy, and the cross entropy was minimized using the Adam optimizer with a learning rate of 0.0001. The training of the CNN took place over 20,000 epochs with batches of 50 training images. The CNN took two hours and forty minutes of CPU time to train, and 32.2 seconds of CPU time to predict on the test set. The final accuracy of the CNN on the test set was evaluated to be 0.911.

Again, a learning curve was created during the training of the CNN model, and is provided in Figure 7. The learning curve had a more gradual elbow than that of the deep NN, but was sharper than the single layer. The validation learning curve also appeared to perform closer to the training learning curve, likely due to the dropout function implemented during training to prevent over reliance on certain nodes in different layers. Also, the gradual improvement on the learning curve appeared to stop around epoch 15,000 indicating convergence.

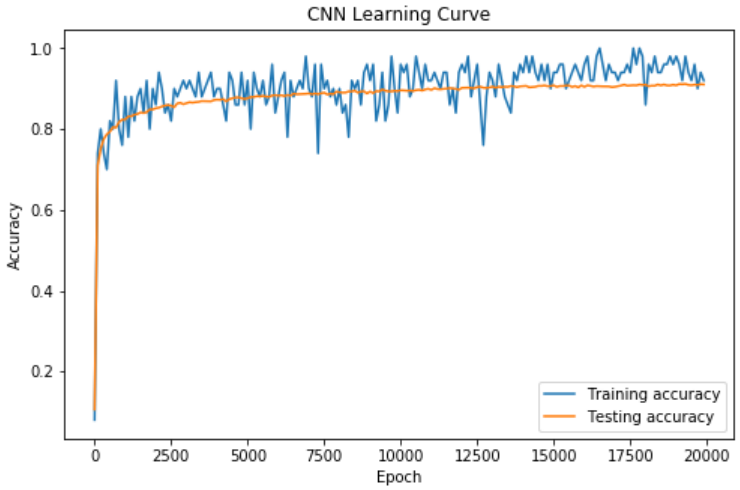

Figure 7 Learning Curve for CNN

For the CNN, I plotted the first ten misclassification of the test set to compare with the other classification methods. The plot of the misclassified images showed quite a bit of improvement in some of the obvious misclassifications, and once again showed the effects of having low quality images and having a white background with clothes that have white. The misclassified images are provided in Figure 8.

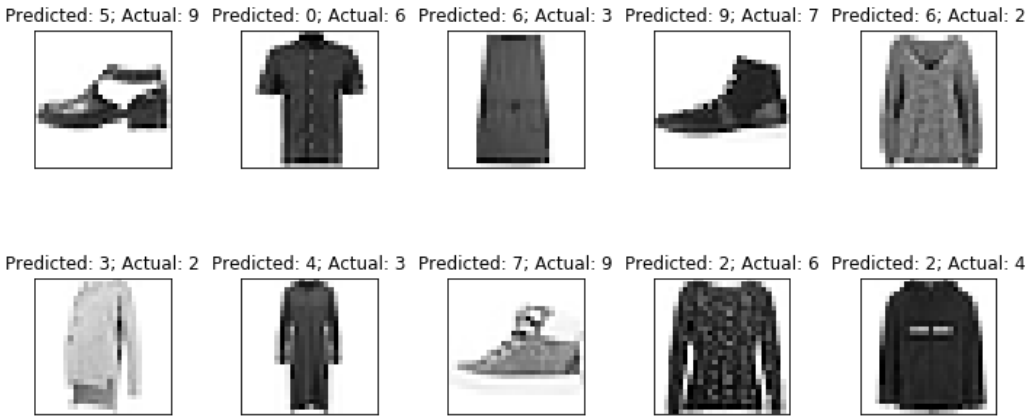

Figure 8 Misclassified Images of Convolutional Neural Network

Of the ten misclassified images, I would have misclassified six of them by looking at the image. I would even argue that some of the labels classified by the creators of the dataset are wrong (specifically frames 8 and 10). I also made a confusion matrix for the CNN, and provided it in Table 5.

Table 5 Confusion Matrix for Convolutional Neural Network

		Predicted									
		0	1	2	3	4	5	6	7	8	9
Actual	0	840	0	15	21	3	1	111	0	9	0
	1	2	978	0	13	2	0	3	0	2	0
	2	17	0	839	12	78	0	53	0	1	0
	3	8	2	5	934	26	0	22	0	3	0
	4	1	1	48	32	885	0	32	0	1	0
	5	0	0	0	0	0	976	0	18	0	6
	6	97	0	54	36	78	0	727	0	8	0
	7	0	0	0	0	0	3	0	978	0	19
	8	1	1	1	3	1	1	0	4	988	0
	9	1	0	0	0	0	4	0	29	0	966

The confusion matrix showed that there was a fairly significant improvement in the classification of shirts in the CNN compared to the classification of shirts in the Deep Neural Network. However, this improvement in the classification of shirts comes at a slight cost of a drop-off in performance in the classification of t-shirts/tops.

Conclusion

While the fashion MNIST dataset is much more complex and difficult to classify compared to the MNIST hand-written digits dataset, accuracy of the model can still score around 90% fairly easily. However, when choosing a model for a classification problem, it is important to consider the tradeoffs in performance versus the compute efficiency of the models. A summary table of the models used to classify the fashion MNIST dataset is provided in Table 6.

Table 6 Summary Data of Classification Algorithms

Model	Accuracy	Total CPU Train Time	Total CPU Prediction Time
Random Forest	0.813	4 mins 23 sec	2.73 sec
Single Layer Neural Network	0.834	16 sec	108 ms
Deep Neural Network	0.891	15 min 19 sec	1.32 sec
Convolutional Neural Network	0.911	2 hrs 40 min	32.2 sec

When choosing a model to use in production is important to consider the use case of the algorithm. As table 6 shows, the CNN provided the most accurate results when classifying the test set, but it also took more than 10 times longer to train the model, and almost 25 times longer to predict the results than the deep neural network. If the algorithm is going to be used in a medical setting for diagnosing patients with some disease, then the extra 2% increase in accuracy would likely be worth the longer training and predicting time (given that a patient has more than 30 seconds to live). However, if the neural network is going to be used in an online setting, the quick prediction time may be more valuable.

The single layer neural network did not perform as well as either the deep neural network or the CNN; however, the single layer neural network only needed 16 seconds of total CPU time to train a model, and 108 ms to predict the test images. This extremely quick training and

prediction time could also be very useful in an online setting, especially if there are a lot of visitors to the site, or if you are trying to save money on compute time.

For the task of classifying the images in the fashion MNIST dataset, I think the Convolutional Neural Network would be most appropriate for the task. The actual wall time (stopwatch time) for the model to train was only 20 minutes and 48 seconds, and the predictions only took 6.1 seconds of wall time. Classifying the fashion images is not a time restrictive, and the 2% increase in performance was enough to tolerate the extended time training time.