# Algorithm Selection

**Random Forest (RF)**

One of the main issues of decision trees is the problem of overfitting which can be solved by introducing the ensemble method employed by random forests. By taking the mode decision over many trees provides a more accurate, powerful classification with less variance in results.

The hyperparameters I will be exploring are:

* n\_estimators: The number of decision trees the model is made up of. This is probably the most important feature in determining the accuracy of the results. However, while increasing this number, the time of execution would also increase.
* max\_features: Here we look at how we can reduce the number of features we consider at each split of a tree (and hence reduce runtime) while still maintaining strong accuracy.
* bootstrap: A ‘true or false’ as to whether a sample of the input data is used when creating each tree. The test here will be to see if not bootstrapping can improve results without overfitting (i.e. repeat successful results when presented with new data).

**Stochastic Gradient Descent (SGD)**

Gradient descent can be seen as the foundation for many machine learning algorithms, the ‘stochastic’ variant greatly improves accuracy when handling a larger dataset, as posed in Fashion MNIST. This is done by only following the gradient of one point at a time rather than looking through the whole dataset.

Hyperparameters:

* loss: The loss function controls the fit of the model. I will be testing the application of the SGD to be a Support Vector Machine (SVM)/hinge, or Logistic regression/log .
* penalty: A regularisation term added to the loss function. Here we will look at which of ‘L2’, ‘L1’ or ‘Elastic Net’ lead to optimal performance.

**Convoluted Neural Network (CNN)**

A CNN is a fully connected feed forward neural network. They are most commonly applied to image recognition because they are effective in reducing the number of parameters without losing quality on the models – hence it is a good application for our Fashion MNIST dataset.

Hyperparameters:

* Number of layers: The CNN is made up of input, convolution, pooling, fully connected, softmax and output layers. I will be seeing how changing the number of max pooling layers and the addition of dropout layers affects the results.
* optimizer: The methods that change the attributes of the network in order to reduce loss. We will be looking at the ‘adagrad’, ‘adam’ and ‘adadelta’ optimizers.
* activation: The activation function defines the output of the nodes in a layer given an input. The performance of the activation function is vital in the performance of the network since it is the basis of the networks response to behaviour at each layer. I will be looking at the Relu and sigmoid activation functions.

**Methodology**

Data Preprocessing

The data is loaded as 60,000 training images and labels, and 10,000 testing images and labels. For RF and SGD each 28x28 image was reshaped into one list, this was not necessary for CNN as this network can process 2D images data. For SGD and CNN the data needed to be inputted as a normalised input by removing the mean and scaling to a unit vector.

During training, the data was randomly split by 5-fold cross-validation. This allows the models to learn over a variety of inputs in order to avoid overfitting.

Optimisation

The optimization was done using ‘Scikit learn’s GridSearchCV whereby, given a dictionary of required input hyperparams, an exhaustive method is performed to repeat the given model over all necessary params and return the optimal combinations.

I will be measuring algorithm performance on the accuracy of predictions (percentage of predictions correct) and the time they take to execute.

**Results**

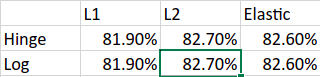
Random Forest

The optimal parameters returned from the grid search for random forest was {Bootstrap = False, max\_features = sqrt, n\_estimators = 300}. Suggesting that by not bootsrapping the data, we can achieve more accurate results without overfitting to the data. It also offered the largest number of estimators, increasing the number of estimators further may continue to improve the results but would not be worth the additional computational time.

The final results on this combination were: **accuracy** = 88.3% in **time**, 327 seconds.

SGD

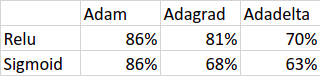
The results of SGD can be found in the table:

 Here you can see that ‘Hinge’ and ‘Log’ haven’t affected the accuracy of the results, however, time on ‘Log’ is drastically increased making ‘Hinge’ the more efficient

The optimal SGD model used the hinge loss function (acting as an SVM) and the L2 penalty leading to the results: **accuracy** = 82.7% in **time**, 254 seconds.

CNN

Below are the results for CNN:

 Here it is clear that ‘Adam’ is the best optimiser and ‘Relu’ is the best activation function. Although ‘Sigmoid’ returned the same accuracy for ‘Adam’, it also had more variance across each epoch which is an issue that would be solved by implementing the network over many more epochs. In this test I only varied between 10 and 15 epochs due to computational and time constraints.

The overall optimal model for the CNN was {Optimiser = Adam, Activation = Relu} leading to the results: **accuracy** = 86% in **time**, 30 minutes (on 10 epochs).

**Conclusion**

To conclude from my results, it has been shown that the random forest algorithm provided the most accurate model, with SGD being the least accurate. The simplified CNN was significantly faster than the other two (at 14 seconds), although this comes with greater variance in results.

It is also worth noting that, for the CNN, when adding 3 ‘convoluted’ layers and a ‘max pool’ layer to the original basic model we improved the model (which was used for the results above). This process took by far the longest. The length of time could be reduced by including dropout layers after each convoluted layer, but losing accuracy as a result.

I found the use of L1 penalty in SGD to be the most sensitive parameter, taking 750 seconds to complete the algorithm with less accuracy than it’s L2 counterpart.