CS985 Machine Learning for Data Analytics
Deep Learning Assignment
Neural Networks for the Twenty Newsgroups and MNIST Digits Datasets
Team: **Prometheus**

Mihail Mihaylov, MSc Advanced Computer Science,
pjb18184@uni.strath.ac.uk, ID: 201888925
Linden Smith, MSc Advanced Computer Science with Big Data,
jrb18149@uni.strath.ac.uk, ID: 201876888
Jack Nielsen, MSc Advanced Computer Science,
wdb18167@uni.strath.ac.uk, ID: 201870058

9^{th} April 2019

1 MNIST Digits

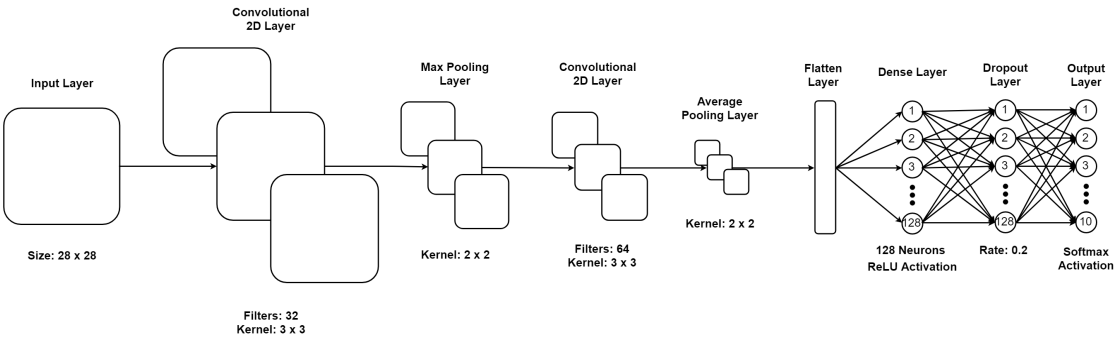

Figure 1: Network 1: MNIST

1.1 Final Architecture

The best architecture found for the MNIST Images dataset was a Convolutional Neural Network with 2 convolutional (32 and 64 filters with a kernel of 3x3) and 2 pooling layers (2x2 kernel) and 1 fully connected layer with 128 neurons and a dropout rate of 0.2. The Activation function chosen for that architecture was a ReLU activation which performed best and most efficiently. Other activation functions tested were Leaky ReLU and elu. The neural network was built using the Keras library within Tensorflow using the sequential function.

For the learning rate and batch size, numerous combinations were explored, with learning rate between 0.2 and 0.0001 and the best results for the 1st combination were obtained with a learning rate of 0.005. Higher learning rates led to the algorithm struggling to converge to an optimal solution and with lower rates the training time took too long to converge. The batch sizes explored varied from 32 to 256 with a batch size of 64 proving to yield the best results. A higher batch size would lead to faster training time but worst performance and the lower batch size would make training a lot slower taking a lot more epochs to reach an optimal solution. Another hyperparameter explored was an optimizer for the neural network. After training the neural network it was decided that the best and fastest results were reached with an Adam optimizer. Other optimizers explored were AdaDelta and Gradient Descend which took a higher number of epochs to reach to a good convergence. For the final architecture the number of epochs that gave the highest testing accuracy was 10 which took around 8 minutes to train. For the loss function it was used a sparse categorical cross-entropy since that is the best loss function for a classification problem like that one.

1.2 Other Architectures

Other architectures explored involved a multi layered perceptron (MLP) with 2 hidden layers (512 and 256 neurons) and a ReLU activation function as well as a CNN like the final architecture but with an additional dense layer of 128 neurons. Other CNNs explored involved the final architecture with a leaky relu or elu activation function and different learning rates, batches and epochs, as well as 3 other MLP architectures like the one previously described but with a 3rd dense layer of 128, a sigmoid activation function and gradient descent optimizer or a tanh

activation with Adam optimizer. The final architecture was chosen as such because it provided the highest results on the testing data for an optimal amount of time.

1.3 Results Table

Example	Comb	Param & Config	Train Acc	Val Acc	Test Acc	Train Time
a	**1**	**LR:0.005 Epochs:10 Batches: 64**	**99.43%**	**98.95%**	**99.21%**	**8m 4s**
b	2	LR:0.001 Epochs:50 Batches:64	99.85%	98.38%	98.25%	7m 46s
c	3	LR:0.003 Epochs:5 Batches:64 1 with additonal hidden layer	99.13%	99.30%	99.2%	4m 5s
d	4	LR:0.001 Epocs:20 Batches: 32 comb 2 with additional hidden layer 128 neurons	99.52%	98.32%	98.17%	7m 2s
e	-	LR:0.002 Epochs:6 Batches:128 Comb 1 with leaky ReLU	99.04%	98.98%	98.89%	6m 1s
f	-	LR:0.003 Epochs:6 Batches: 128 comb 1 with elu act	98.88%	98.65%	98.62%	5m 20s
g	-	LR:0.2 Epochs: 40 Batches: 128 comb 2 with sig and gradient descent optimiser	96.81%	97.27%	96.57%	6m 18s
h	-	LR:0.001 Epochs:30 batches:128 comb 2 with tanh act	99.67%	98.12%	97.93%	4m 31s

2 News

2.1 Fully Connected Feed Forward Network Architecture

The final architecture was a fully connected network with an **input layer** of *dict_size* neurons, the number of terms in our term frequency inverse document frequency vocabulary, which was settled to 12500 through experimentation; **one hidden layer** consisting of 256 neurons with **ReLU activation**, **no dropout**, **batch normalisation** performed after activation, connected to a **softmax** output layer with 20 neurons: one for each category we are predicting. Categorical cross-entropy is employed with Adam optimisation for our loss function, with weight optimisation occuring per batch. The weights are initialised according to Xavier Initialisation and the biases are initialised to zero. We believe this network strikes the best balance between accuracy, training times and complexity.

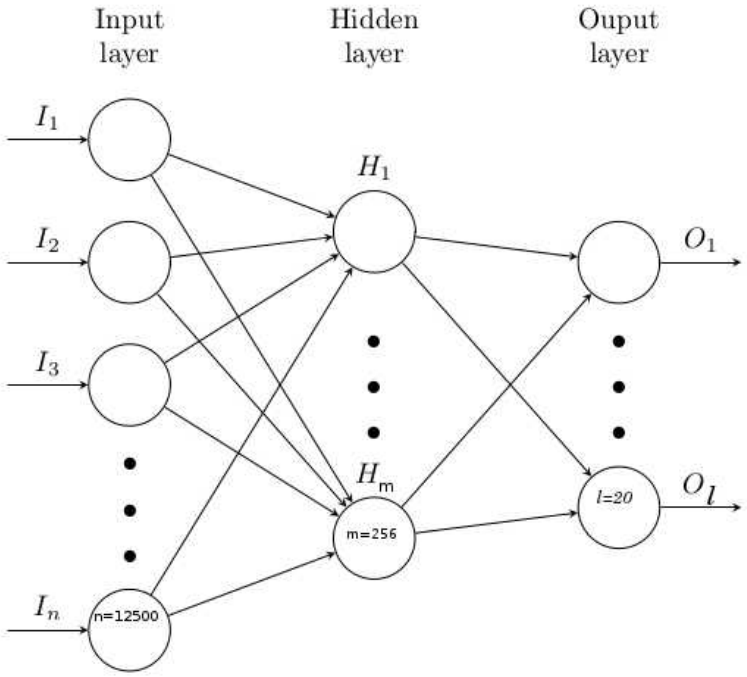

Figure 2: Network 1: 20 NewsGroups

2.2 Results Table

In optimising the parameters, controlled search, where other parameters are kept the same, was employed; above is only a selection.

A training, validation and test split of 0.7/0.1/0.2 provided better results than the pre-built sci-kit learn split two thirds train, a third test — likely due to increased training samples. For our best combination, similar results were found between 3 and 4 epochs, with 3 more consistently reaching 91.4%.

Networks with sigmoid and tan hyperbolic activations struggled in training, sometimes needing two to three times the epochs of a Rectified Linear Unit to reach convergence. However, when doing so, results were typically similar or marginally worse. Learning rate and size of network were related; with a larger network, smaller learning rates seemed to improve results, likely by slowing the process of overfitting, allowing us to stop at the appropriate epoch rather than wait for one iteration of n batches through the data, overfitting between epochs. Learning rate also affects the time and ability to converge, with the learning rate of 0.001 proving to be most efficient across a few configurations. Initialising the network weights with Xavier Initialisation improved time to convergence significantly, as did the zero initialiser on the biases. Its possible that, with random initialisation, the model was stuck in local optima.

Unless otherwise stated, the parameters are as in the final configuration (1) or (2).

Example	Comb	Param & Config	Train Acc	Test Acc
a	**1**	**LR 0.001, 1 hidden layer of 256 neurons, 3 epochs, batch size of 64 and a ReLU activation**	**99.6%**	**91.36%**
b	-	1 with two hidden layers of 512 neurons and 0.5 dropout	92.6%	90.25%
c	-	1 with three hidden layers, two 512 neurons and one 256, 4 epochs, grad desc opt	99.6%	87.8%
d	1	only 128 neurons in the single hidden layer, 3 epochs	98.7%	91.15%
e	1	0.0001 LR, 12 epochs, 32 Batch, sig act	97.24%	89.83%
f	1	0.01 LR, 2 epoch, tanh act	99.65%	90.27%
g	1	1 with 256 batch size, 8 epochs	99.21%	91.12%
h	2	Single convolutional layer followed by global max pooling, then by 20 neuron softmax layer	0.99	0.87
i	2	Two convolutional layers, max pooling, then 20 neuron softmax layer	96.32%	80.23%
j	2	Single convolution layer, max pooling, two 32 neuron fully-connecter layers, then 20 neuron softmax layer	93.51%	81.13%

Dropout relates to the size of the network, having a more beneficial effect in larger networks with more neurons and layers as it reduces overfitting. In our case, with two layers of 512 neurons, dropout reduced overfitting, preventing the network from rushing to a full accuracy training fit, and instead teasing the underlying relation from the data; allowing us to use early stoppage when the validation accuracy stops improving. However, with a smaller network of 256 neurons, there was little need for dropout, having no noticeable effect on final accuracy, and the smaller network itself having the same accuracy as the one with 512 neurons in the only hidden layer. Early stoppage was essential until the network was fully optimised to complete training in two epochs. Following this, the drop in validation score often came as soon as the third or fourth epoch. Until optimisation, a limitation on early stoppage to trigger only if at-least n epochs had occurred, was used, and a decrease in validation accuracy between epochs was used as the trigger. Batch normalisation can aid in poor initialisations, but too much regularisation can lead to underfitting. Batch normalising proved to provide a better result in this case however. The combination of this and Xe initialisation leads to strong training speed and initialisation, quickly climbing in training accuracy. As far as the number of neurons, the more there are, the

slower training can be, and somewhat surprisingly can prevent convergence; it is likely that the strong computational power of a network with, for example, three layers of 512 neurons, leads to overfitting on the current batch and the pattern does not generalise. Our training method, that is, adam optimisation with mini-batches, optimises weights at the end of each batch. A nonrepresentative batch can lead to losses in accuracy. However, we found that larger batches performed slightly poorer. A batch size of 64 seems optimal, with similar performance from $n = 32$, $n = 16$ reaching a full fit of the training data between one and two epochs due to the increase frequency of weight optimisations. The size should perhaps be balanced with the desire to include mixed samples in each batch there are twenty classes in the dataset;however, the samples are not stratified, and the data as a whole has similar distributions of of each category. As far as data pre-processing, stemming was explored but found to reduce performance, likely due to filtered ascii strings and numerical features ending up as strong predictors inside the black box network. In training, decreasing the learning rate slowed time to convergence, however this can also allow for more careful optimisation: we can stop training at the ideal time. With smaller learning rates, of orders of magnitude, however, no improvements over 0.001 were discovered. The training accuracy and loss, as-well as for validation, are output to tensorboard with the time unit as mini-batch instead of epoch as the best configurations tended to have too few epochs for meaningful visualisation. Validation does not change per mini-batch, however, nor does test. In troubleshooting, summaries per epoch can be used, but the additional tensorflow operations required were found to slow down training significantly.

2.3 Convolutional Nerual Network Architecture

The advantage of the convolutional neural network architecture over the fully connected is we can input the data as simple tokenized strings rather than a large tfidf dictionary which requires sparse matrices for storage and therefore training in batches; this reduces the required memory for storage significantly, and larger batch sizes, even the entire data in one step, can be used. The features learned by the CNN are potentially words, then, or subsections of words, phrases, etc. Through experimentation, we found better results without trimming ascii characters and numerics, as with the feed forward network, which can still be learned by the CNN. CNNs also tend to have longer training times than Fully Connected Feed Forward networks, which is the case here also.

Adding more layers decreases accuracy and is analogous to overfitting for this problem. Similarly, increasing the number of feature maps leads to overfitting.

Increasing the number of feature maps on the convolutional layer yielded increased accuracy to a point, however this effect stopped at a feature size of 64. We theorise this is because the number of learnable features in the dataset is ¡= 64, so increasing the number of feature maps does not yield increased peformance.

The effect of the network hyperparameters and initialisation distribution were similar to that of the feed-forward network, and similar selections for both yielded good results.

The network was built using keras due to the ease of use, however many poweful features were used to improve network performance including spatial dropout on the convolutional layer. In addition, the time saved by using keras over tensorflow allowed us to search many combinations of hyperparameters using a grid search, which in turn allowed us to achieve higher accuracy rates across the board.