MLP Coursework 1

S2569758

Abstract

In this report we study the problem of overfitting, which is the training regime where performance increases on the training set but decrease on validation data. **[In overfitting, the model learns too much about the training data and is unable to generalise its learnings for unseen data which leads to poor performance on new data points.]** . We first analyse the given example and discuss the probable causes of the underlying problem. Then we investigate how the depth and width of a neural network can affect overfitting in a feedforward architecture and observe that increasing width and depth **[Increaseing the depth and width slightly increases the model performance overall but if the either or both the width and depth are increased by a lot, the model overfits the training data and returns low accuracy scores for any unseen data.]** . Next we discuss how two standard methods, Dropout and Weight Penalty, can mitigate overfitting, then describe their implementation and use them in our experiments to reduce the overfitting on the EMNIST dataset. Based on our results, we ultimately find that **[Q3]** . Finally, we conclude the report with our observations and related work. Our main findings indicate that **[complex models can easily overfit the data resulting in low accuracy scores on unseen data points. We can use various techniques like Dropout & L1/L2 regularisations to ensure that the model doesn't memorise the data and is instead able to generalise the learnings from it.]** .

1. Introduction

In this report we focus on a common and important problem while training machine learning models known as overfitting, or overtraining, which is the training regime where performances increase on the training set but decrease on unseen data. We first start with analysing the given problem in Figure 1, study it in different architectures and then investigate different strategies to mitigate the problem. In particular, Section 2 identifies and discusses the given problem, and investigates the effect of network width and depth in terms of generalisation gap (see Ch. 5 in Goodfellow et al. 2016) and generalisation performance. Section 3 introduces two regularisation techniques to alleviate overfitting: Dropout (Srivastava et al., 2014) and L1/L2 Weight Penalties (see Section 7.1 in Goodfellow et al. 2016). We first explain them in detail and discuss why they are used for alleviating overfitting. In Section 4, we incorporate each of them and their various combinations to a three hidden layer[1] neural network, train it on the EMNIST dataset, which contains 131,600 images of characters and digits, each of size 28x28, from 47 classes. We evaluate them in terms of generalisation gap and performance, and discuss the results and effectiveness of the tested regularisation strategies. Our results show that **[Q3]** .

Finally, we conclude our study in Section 5, noting that **[complex models can easily overfit the data resulting in low accuracy scores on unseen data points. We can use various techniques like Dropout & L1/L2 regularisations to ensure that the model doesn't memorise the data and is instead able to generalise the learnings from it.]** .

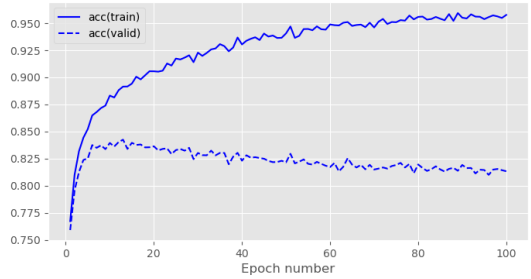
2. Problem identification

Overfitting to training data is a very common and important issue that needs to be dealt with when training neural networks or other machine learning models in general (see Ch. 5 in Goodfellow et al. 2016). A model is said to be overfitting when as the training progresses, its performance on the training data keeps improving, while its is degrading on validation data. Effectively, the model stops learning related patterns for the task and instead starts to memorize specificities of each training sample that are irrelevant to new samples.

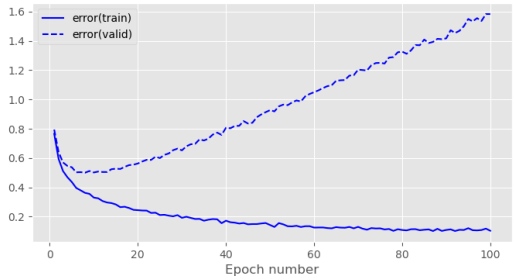
[Overfitting causes a model to start recognising random deviances and noise present in the training data as a regularity rather than learning the underlying patterns. This poor generalisation leads to the model struggling to capture the trends on unseen data points and hence performing poorly on them. This makes the model less reliable to be used in real world applications. Another direct consequence of overfitting is that it skews the relationship between variables affecting the interpretability of the model and making it difficult to fix the model. Overfitting a model also wastes a lot of resources as it would require a lot of computational power to create a model which won't be able to handle new data points very well.] .

Although it eventually happens to all gradient-based train-

[1] We denote all layers as hidden except the final (output) one. This means that depth of a network is equal to the number of its hidden layers + 1.



(a) accuracy by epoch



(b) error by epoch

Figure 1. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for the baseline model.

ing, it is most often caused by models that are too large with respect to the amount and diversity of training data. The more free parameters the model has, the easier it will be to memorise complex data patterns that only apply to a restricted amount of samples. A prominent symptom of overfitting is the generalisation gap, defined as the difference between the validation and training error. A steady increase in this quantity is usually interpreted as the model entering the overfitting regime.

[Network capacity refers to the ability of the model to learn the general characteristics of the data. When a model is more complex, it starts memorising the data rather than learning the trends from it. This usually happens when the model has more room than what is needed for capturing the patterns in the data. If the dataset is small or simple, the basic essense of it can be captured with a small model itself. If in scenarios like this, a complex model is used, the model has the capacity to learn more than the general features of the data and will also start to fit the outliers in the data leading to overfitting on the training set. Thus it is very important to choose an appropriate model for the data given to us otherwise choosing a complex model with more network capacity will result in overfitting of the data.] .

Figure 1a and 1b show a prototypical example of overfitting. We see in Figure 1a that **[the training accuracy of the model increases rapidly in the first few epochs and then gradually slows down as the epochs continue to run. Even though the validation accuracy also increases at a good rate in the first few epochs, it starts flattening out at a point and then slowly starts to decline. This is the point where overfitting has started, stating that the model has learned the training data and is unable to generalise it to the validation data. This is further validated by the graph in Figure 1b which plots the error of the model on the training and validation dataset. We can see that the training error reduces rapidly at first and then eventually tends to 0. After a rapid initial decline in the error of the validation set, the error starts increasing at the same time when the accuracy of the validation set starts decreasing. This is due to the fact the model has now overfit and is struggling to process unseen data points.]** .

# Hidden Units	Val. Acc.	Train Error	Val. Error
32	78.9	0.563	0.705
64	80.9	0.345	0.809
128	80.5	0.163	0.805

Table 1. Validation accuracy (%) and training/validation error (in terms of cross-entropy error) for varying network widths on the EMNIST dataset.

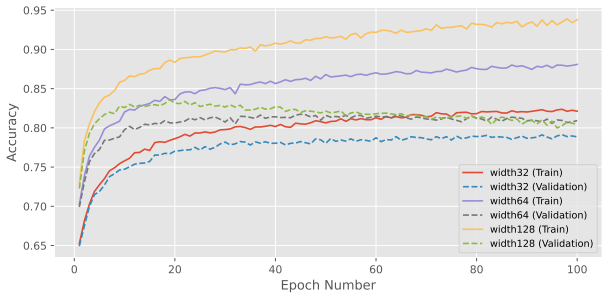
The extent to which our model overfits depends on many factors. For example, the quality and quantity of the training set and the complexity of the model. If we have sufficiently many diverse training samples, or if our model contains few hidden units, it will in general be less prone to overfitting. Any form of regularisation will also limit the extent to which the model overfits.

2.1. Network width

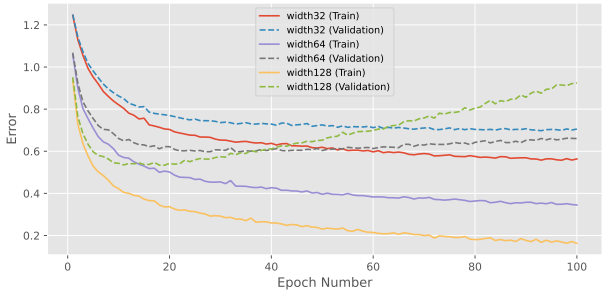
[Question Table 1 - Fill in Table 1 with the results from your experiments varying the number of hidden units.] [Question Figure 2 - Replace the images in Figure 2 with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden units.]

First we investigate the effect of increasing the number of hidden units in a single hidden layer network when training on the EMNIST dataset. The network is trained using the Adam optimiser with a learning rate of 9×10^{-4} and a batch size of 100, for a total of 100 epochs.

The input layer is of size 784, and output layer consists of 47 units. Three different models were trained, with a single hidden layer of 32, 64 and 128 ReLU hidden units respectively. Figure 2 depicts the error and accuracy curves over 100 epochs for the model with varying number of hidden units. Table 1 reports the final accuracy, training error, and validation error. We observe that **[the final validation accuracy of all the models is very close (Figure 2a) indicating that the models performance very similarly. In Figure 2b, if we look at the train and the validation error, we can see the difference in the performance of the models. The training error of the model with 32 hidden units is the highest and the one with 128 hidden units is**



(a) accuracy by epoch



(b) error by epoch

Figure 2. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network widths.

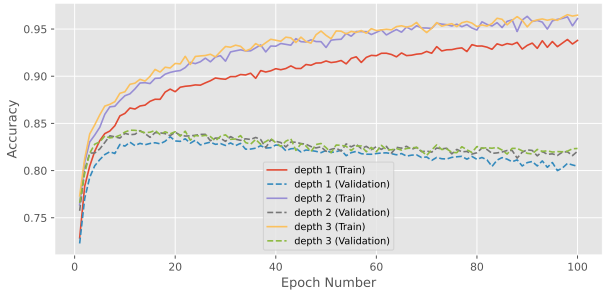
the lowest. This would ideally mean that the model with 128 neurons would be perform the best. But if we look at the validation error, we can see the 128 units model has the highest validation error. This means that model has overfit the training data and is performing relatively worse on data it has not seen before when compared to the 32 units model.

From Table 2, we can see that the model with 64 hidden units has the highest validation accuracy and the lowest validation error signifying that the model has learned from the training data and is able to generalise the best on unseen data out of all the 3 models. This model is relatively not too complex and is performing better than a model with more width.] .

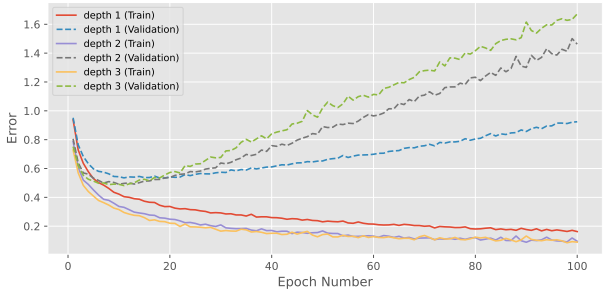
[Changing the width of the model gave us a variety of results. We saw that a model with more width (more complex model) is not necessarily the best model for most scenarios. Even though the model might slightly perform better on the training data, it gives a higher error and not a significant increase on the accuracy for new data points. The results we've achieved also prove the point about the network capacity, where a higher width model has the more than the required capacity to learn the features of the data and thus it starts fitting the outliers to the data leading to overfitting. This won't always be the case as this test was conducted on a relatively simple dataset. If the data to be processed had more features, we would require a model with more width to learn the underlying patterns in the data. A complex model in this scenario won't overfit because

# Hidden Layers	Val. Acc.	Train Error	Val. Error
1	80.5	0.163	0.924
2	82.1	0.095	1.463
3	82.3	0.089	1.672

Table 2. Validation accuracy (%) and training/validation error (in terms of cross-entropy error) for varying network depths on the EMNIST dataset.



(a) accuracy by epoch



(b) error by epoch

Figure 3. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network depths.

its capacity is being completely utilised in learning the features of the data.] .

2.2. Network depth

[Question Table 2 - Fill in Table 2 with the results from your experiments varying the number of hidden layers.] [Question Figure 3 - Replace these images with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden layers.]

Next we investigate the effect of varying the number of hidden layers in the network. Table 2 and Figure 3 depict results from training three models with one, two and three hidden layers respectively, each with 128 ReLU hidden units. As with previous experiments, they are trained with the Adam optimiser with a learning rate of 9×10^{-4} and a batch size of 100.

We observe that **[models of different depth are all capable of classifying classes of EMNIST dataset. From Figure 3a, providing the models with more depth has en-**

abled them to extract features from the data efficiently and give excellent accuracy scores. Irrespective of their depth, in case of validation data, all of them are performing similarly and giving accuracy of around 80% (Table 2). The rise of accuracy is very steep in the first 10 epochs, and then it is slow gradual improvement in training accuracy and barely any change in the validation accuracy.

The error behaves very differently and gives a good understanding of the models. In Figure 3b, we can see that the error of all the models on the training set approaches 0, but on the validation set starts increasing as the epochs increase. This is the point of overfitting; and the magnitude of the error indicates how much the model has learned the training data. The model with the highest depth has the highest capacity and hence has been able to learn the train set the best resulting in it having the highest error on the validation set (from Table 2).] .

[Increasing the depth of the model can be helpful in processing large or complex datasets. But in other situations it can turn out to be counter productive and as the model will overfit the data. This is because certain capability is required for the model to process the data, but if the model has more room to capture features, it'll start fitting the noise to the dataset. As the depth of the model increases, there are more neurons available which allow the model to extract more features from the dataset allowing it to overfit the training data and result in poor accuracy scores for unseen data. It also becomes computationally expensive to train the model and the resources will be wasted if the model overfits on the training data. We can tackle the problem of overfitting by using methods like Dropout and Regularisation which penalise the model for generalising too much to the training set preventing overfitting.] .

3. Dropout and Weight Penalty

In this section, we investigate three regularisation methods to alleviate the overfitting problem, specifically dropout layers and the L1 and L2 weight penalties.

3.1. Dropout

Dropout (Srivastava et al., 2014) is a stochastic method that randomly inactivates neurons in a neural network according to an hyperparameter, the inclusion rate (*i.e.* the rate that an unit is included). Dropout is commonly represented by an additional layer inserted between the linear layer and activation function. Its forward pass during training is defined as follows:

$$\text{mask} \sim \text{bernoulli}(p) \tag{1}$$

$$\boldsymbol{y}' = \text{mask} \odot \boldsymbol{y} \tag{2}$$

where $\boldsymbol{y}, \boldsymbol{y}' \in \mathbb{R}^d$ are the output of the linear layer before and after applying dropout, respectively. mask $\in \mathbb{R}^d$ is a mask vector randomly sampled from the Bernoulli distribution with inclusion probability p, and $\odot$ denotes the element-wise multiplication.

At inference time, stochasticity is not desired, so no neurons are dropped. To account for the change in expectations of the output values, we scale them down by the inclusion probability p:

$$\boldsymbol{y}' = \boldsymbol{y} * p \tag{3}$$

As there is no nonlinear calculation involved, the backward propagation is just the element-wise product of the gradients with respect to the layer outputs and mask created in the forward calculation. The backward propagation for dropout is therefore formulated as follows:

$$\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{y}} = mask \tag{4}$$

Dropout is an easy to implement and highly scalable method. It can be implemented as a layer-based calculation unit, and be placed on any layer of the neural network at will. Dropout can reduce the dependence of hidden units between layers so that the neurons of the next layer will not rely on only few features from of the previous layer. Instead, it forces the network to extract diverse features and evenly distribute information among all features. By randomly dropping some neurons in training, dropout makes use of a subset of the whole architecture, so it can also be viewed as bagging different sub networks and averaging their outputs.

3.2. Weight penalty

L1 and L2 regularisation (Ng, 2004) are simple but effective methods to mitigate overfitting to training data. The application of L1 and L2 regularisation strategies could be formulated as adding penalty terms with L1 and L2 norm square of weights in the cost function without changing other formulations. The idea behind this regularisation method is to penalise the weights by adding a term to the cost function, and explicitly constrain the magnitude of the weights with either the L1 and L2 norms. The optimisation problem takes a different form:

$$\text{L1: } \min_{\boldsymbol{w}} E_{\text{data}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) + \lambda \|w\|_1 \tag{5}$$

$$\text{L2: } \min_{\boldsymbol{w}} E_{\text{data}}(\boldsymbol{X}, \boldsymbol{y}, \boldsymbol{w}) + \lambda \|w\|_2^2 \tag{6}$$

where E_{data} denotes the cross entropy error function, and $\{\boldsymbol{X}, \boldsymbol{y}\}$ denotes the input and target training pairs. λ controls the strength of regularisation.

Weight penalty works by constraining the scale of parameters and preventing them to grow too much, avoiding overly sensitive behaviour on unseen data. While L1 and L2 regularisation are similar to each other in calculation, they have different effects. Gradient magnitude in L1 regularisation does not depend on the weight value and tends to bring small weights to 0, which can be used as a form of feature selection, whereas L2 regularisation tends to shrink the weights to a smaller scale uniformly.

[The L1 penalty also known as Lasso regularisation and L2 penalty known as Ridge regularisation can be used together to tackle the overfitting that occurs during the training of the model. One of the approaches to combining the L1 and L2 regularisations is called the Elastic Net. The equantion for the Elastic Net is given by:

$$E = E_{\text{data}} + \lambda_1 \sum_{i=1}^{N} |w_i| + \lambda_2 \sum_{i=1}^{N} w_i^2 \tag{7}$$

where λ_1 and λ_2 are the hyperparameters for the L1 and L2 regularisation respectively.

The L1 regularisation is used to reduce the number of features by shrinking the coefficients of the less important features to 0. The L2 regularisation penalises the model for having large coefficients and shrinks them to a smaller scale. This helps in reducing the variance of the model and prevents overfitting. By adjusting the values of λ_1 and λ_2 appropriately, we can control if we want more of L1 or L2 penalties applied to the weights. This helps us in controlling the number of features and the size of the coefficients of the model.] .

4. Balanced EMNIST Experiments

[Question Table 3 - Fill in Table 3 with the results from your experiments for the missing hyperparameter values for each of L1 regularisation, L2 regularisation, and Dropout (use the values shown on the table).]

[Question Figure 4 - Replace these images with figures depicting the Validation Accuracy and Generalisation Gap (difference between validation and training error) for each of the experiment results varying the Dropout inclusion rate, and L1/L2 weight penalty depicted in Table 3 (including any results you have filled in).]

Here we evaluate the effectiveness of the given regularisation methods for reducing the overfitting on the EMNIST dataset. We build a baseline architecture with three hidden layers, each with 128 neurons, which suffers from overfitting as shown in section 2.

Here we train the network with a lower learning rate of 10^{-4}, as the previous runs were overfitting after only a handful of epochs. Results for the new baseline (c.f. Table 3) confirm that lower learning rate helps, so all further experiments are run using it.

Then, we apply the L1 or L2 regularisation with dropout to our baseline and search for good hyperparameters on the validation set. We summarise all the experimental results in Table 3. For each method, we plot the relationship between generalisation gap and validation accuracy in Figure 4.

First we analyze three methods separately, train each over a set of hyperparameters and compare their best performing results.

[Q13] .

[Question 14 -] .

5. Conclusion

[Question 15 - Briefly draw your conclusions based on the results from the previous sections (what are the take-away messages?), discussing them in the context of the overall literature, and conclude your report with a recommendation for future directions] .

Model	Hyperparameter value(s)	Validation accuracy	Train Error	Validation Error
Baseline	-	0.837	0.241	0.533
Dropout	0.6	80.7	0.549	0.593
	0.7	83.0	0.445	0.511
	0.85	85.1	0.329	0.434
	0.97	85.4	0.244	0.457
L1 penalty	5e-4	79.5	0.642	0.658
	1e-3	73.7	0.872	0.882
	5e-3	2.41	3.850	3.850
	5e-2	2.20	3.850	3.850
L2 penalty	5e-4	85.1	0.306	0.460
	1e-3	85.0	0.364	0.454
	5e-3	81.3	0.586	0.607
	5e-2	39.2	2.258	2.256

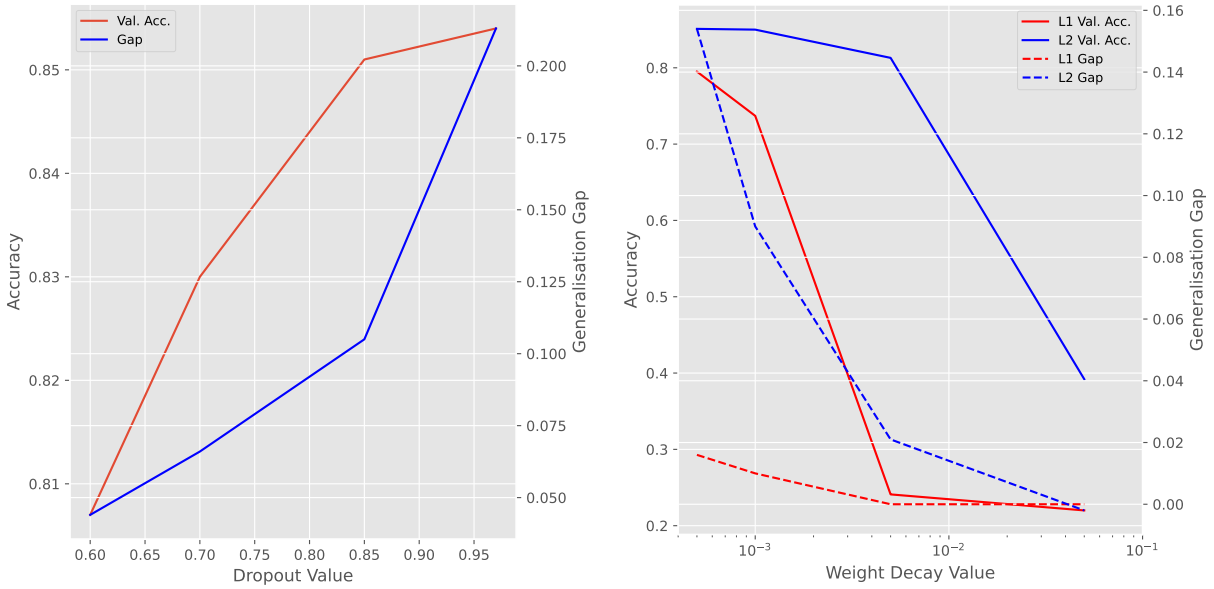
Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series (Dropout, L1 Regularisation, L2 Regularisation) and **bold** indicates the best overall.

References

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Ng, Andrew Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 78, 2004.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.



(a) Accuracy and error by inclusion probability.

(b) Accuracy and error by weight penalty.

Figure 4. Accuracy and error by regularisation strength of each method (Dropout and L1/L2 Regularisation).