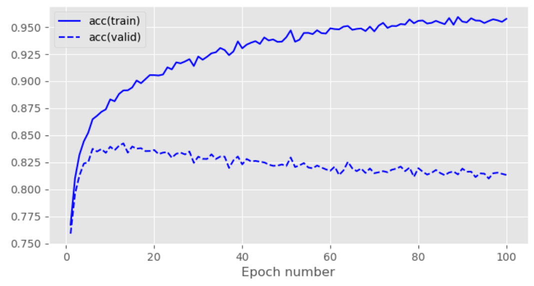
MLP Coursework 1

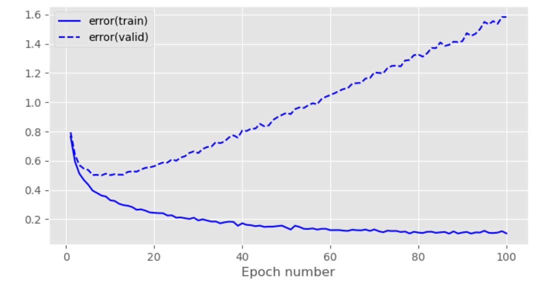
S2713719

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. Overfitting prevents our trained model from generalizing well to unseen data. 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 tend to enable further overfitting. 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 both dropout and weight penalty are able to mitigate overfitting. Finally, we conclude the report with our observations and related work. Our main findings indicate that preventing overfitting is achievable through regularization, although combining different methods together is not straightforward.



(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.

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 generalization gap (see Ch. 5 in Goodfellow et al. 2016a) and generalization performance. Section 3 introduces two regularization techniques to alleviate overfitting: Dropout (Srivastava et al., 2014) and L1/L2 Weight Penalties (see Section 7.1 in Goodfellow et al. 2016a). 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 generalization gap and performance, and discuss the results and effectiveness of the tested regularization strategies. Our results show that both dropout and weight penalty are able to mitigate overfitting.

Finally, we conclude our study in Section 5, noting that preventing overfitting is achievable through regularization, although combining different methods together is not straightforward.

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. 2016a). 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 leads to bad generalization per-

[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.

# Hidden Units	Val. Acc.	Train Error	Val. Error
32	78.3	0.558	0.706
64	80.6	0.346	0.661
128	81.0	0.158	0.902

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

formance in unseen data, as performance on validation data is indicative of performance on test data and (to an extent) during deployment.

Although it eventually happens to all gradient-based training, 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 memorize complex data patterns that only apply to a restricted amount of samples. A prominent symptom of overfitting is the generalization 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.

Figure 1a and 1b show a prototypical example of overfitting. We see in Figure 1a, **[Question 1 - the training accuracy is increasing rapidly across the number of epochs, and after a point, it continues to show a steady rise, eventually exceeding the value of 95.0%. In contrast, the validation accuracy rises rapidly at first, peaks at 0.84, and then decreases slightly, approaching 0.816. Figure 1b, training error consistently decreases to a minimum of 0.0997, but the validation error initially drops and then starts to rise, eventually reaching a high of 1.59. The gap that appears early in the number of epochs between the training and validation metrics indicates that the model begins to overfit.]**

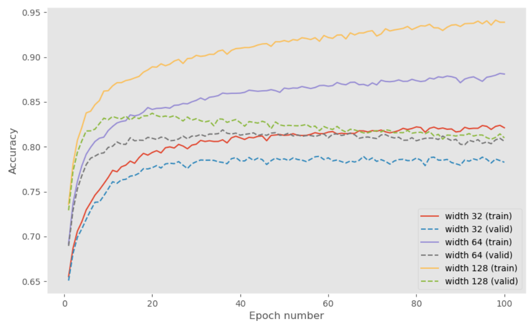
.

2.1. Network width

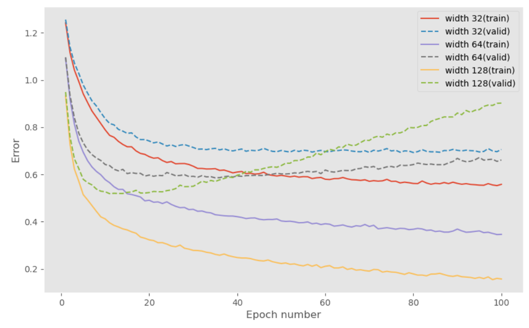
[Question Table 1 - Fill in Table 1 with the results from your experiments varying the number of hidden units.] [Question Figure 2 - Replace the images in Figure 2 with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden units.]

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 optimizer 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 **[Question 2 - the training accuracy improves as the number of hidden units increases, with the largest model (128 units) achieving 93.9%. However, the validation accuracy initially increases but then decreases slightly for each of the three models, reaching the values of 78.3%, 80.6%, and 81%, respectively, indicating overfitting. While training error continues to drop reaching the lowest value of 0.902 for the model of 128 hidden units, validation error for the models with 32 and 64 units initially decreases and then remains almost stable with some small fluctuations reaching the final values of 0.706 and 0.661. In contrast, the model with 128 units shows a nearly linear rise in validation error, creating a noticeable gap between training and validation performance.]** .



(a) accuracy by epoch



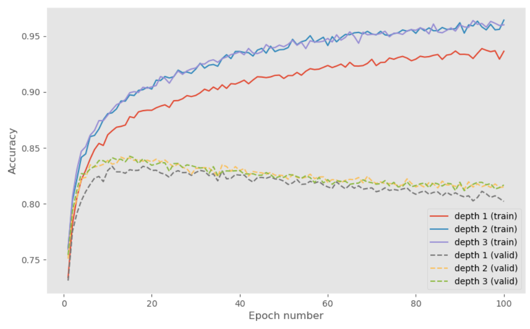
(b) error by epoch

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.

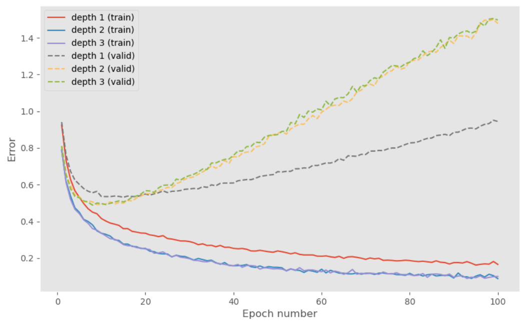
[Question 3 - Research on neural network architecture has shown that increasing the width of a neural network can lead to improved performance on training data by capturing more complex patterns. However, making the model too wide can lead to overfitting, indicating that it memorizes the training data (etc. noise) resulting in reduced generalization to unseen data (see Ch. 5 in Goodfellow et al. 2016a). Thus, the research findings align with the experimental results, demonstrating that increasing the width of the network negatively impacts generalization, creating a gap between training and val-

# Hidden Layers	Val. Acc.	Train Error	Val. Error
1	80.2	0.166	0.907
2	81.5	0.088	1.478
3	81.7	0.100	1.497

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.

idation performance.].

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 optimizer with a learning rate of 9×10^{-4} and a batch size of 100.

We observe that **[Question 4 - As the depth of a neural network increases (e.g., from 1 to 3 layers), a significant gap emerges between training and validation performance. For each network depth, the training accuracy continually improves, often exceeding 95%. In contrast, in the early epochs, the validation accuracy appears to stabilize and then gradually decreases, approaching the levels of 80.2%, 81.5%, and 81.7% for each of the models, respectively. Similarly, the training error decreases to the corresponding low values of 0.166, 0.088, and 0.100. However, after an initial drop, the validation error begins to rise sharply reaching higher levels of 0.907, 1.478, and 1.497 respectively. This discrepancy indicates overfitting, where deeper networks perform well in training data but do not respond effectively to unseen data.]** .

[Question 5 - We observe that as the number of layers increases in a neural network, the model tends to overfit, leading to a big gap between training and validation performance. Research confirms that deeper networks have a higher capacity, enabling them to learn more complex patterns in the training data. This can introduce noise that prevents the model from performing well in the unseen data. (see Ch. 5 in (Goodfellow et al., 2016b, p. 110)] .

3. Regularization

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

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:

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

$$\boldsymbol{y}' = \text{mask} \odot \boldsymbol{y} \quad (2)$$

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:

$$\boldsymbol{y}' = \boldsymbol{y} * p \quad (3)$$

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 regularization (Ng, 2004) are simple but effective methods to mitigate overfitting to training data. The application of L1 and L2 regularization 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 regularization method is to penalize 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 optimization problem takes a different form:

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

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

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 regularization.

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 regularization are similar to each other in calculation, they have different effects. Gradient magnitude in L1 regularization 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 regularization tends to shrink the weights to a smaller scale uniformly.

3.3. Label smoothing

Label smoothing regularizes a model based on a softmax with K output values by replacing the hard target 0 labels and 1 labels with $\frac{\alpha}{K-1}$ and $1-\alpha$ respectively. α is typically set to a small number such as 0.1.

$$\begin{cases} \frac{\alpha}{K-1}, & \text{if} \quad t_k = 0 \\ 1-\alpha, & \text{if} \quad t_k = 1 \end{cases} \tag{7}$$

The standard cross-entropy error is typically used with these *soft* targets to train the neural network. Hence, implementing label smoothing requires only modifying the targets of training set. This strategy may prevent a neural network to obtain very large weights by discouraging very high output values.

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, Dropout and label smoothing (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 regularization 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.

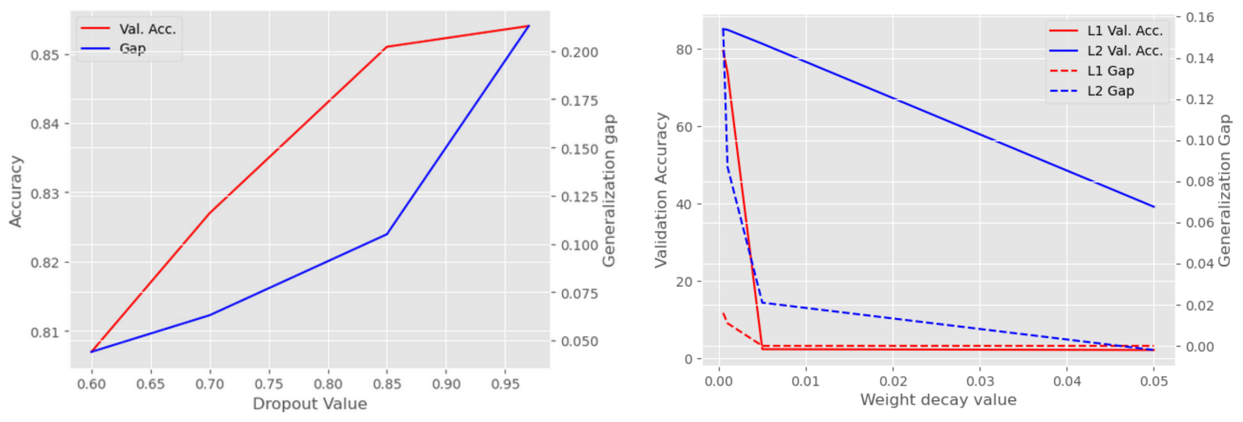
Here, we apply the L1 or L2 regularization with dropout to our baseline and search for good hyperparameters on the validation set. Then, we apply the label smoothing with $\alpha = 0.1$ to our baseline. We summarize all the experimental results in Table 3. For each method except the label smoothing, we plot the relationship between generalization 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.

[Question 6 - Based on the experimental results Table 3, we observed notable insights from the regularization techniques. First, we applied Dropout and ran several experiments with different inclusion probabilities. As the inclusion probability increased, validation accuracy improved, peaking at 85.4% with an inclusion probability of 0.97. However, higher inclusion probabilities reduced regularization, leading to overfitting and a larger generalization gap. The best model performance occurred with an inclusion probability of 0.85, which achieved high validation accuracy while maintaining a smaller generalization gap compared to 0.95, where the gap reached 0.200. In the case of regularization of the L1 penalty, the validation accuracy reached its highest value of 79.5% when small hyperparameters, such as 0.0005, were applied. As lambda values increased, the validation accuracy dropped dramatically to 2.20%,

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	82.7	0.451	0.514
	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	73.9	0.870	0.881
	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	84.9	0.362	0.449
	5e-3	81.3	0.586	0.607
	5e-2	39.2	2.258	2.256
Label smoothing	0.1	85.2	1.026	0.567

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



(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).

while the generalization gap was eliminated. This indicated that increasing the penalty value causes L1 to shrink less important features, leading to underfitting. The L2 penalty, unlike L1, penalizes large weights while allowing less important features to influence the performance of the network. As shown in Table 3, increasing the coefficient results in a significant decrease in validation accuracy, from 85. 1% to a low of 39.2%. At the same time, the generalization gap rapidly decreases and approaches zero. Taking into account both the accuracy of the validation and the generalization gap, the experiment with a coefficient of 0.001 yielded the best results. At last label smoothing with alpha equal to 0.1, the validation accuracy was 85.2%, while the training error was 1.026, which is higher than the validation error, leading to the model generalizing better. In summary, Dropout with an inclusion probability of 0.85 achieved the best balance between high validation accuracy (85.1%) and a small generalization gap (0.105). L2 Penalty (1e-3) also performed well with a validation accuracy of 84.9% and a generalization gap of 0.087.] .

[Question 7 - Combining some of the regularization methods mentioned above can present interesting insights and positively impact the reduction of overfitting. Specifically, some experiments were conducted with the application of Dropout and L2 Penalty, Dropout and L1, L1 and L2, and Dropout with L1 and L2 for a set of hyperparameters each time (inclusion probability, coefficient of L1 and L2 Penalty). As expected, given the individual performance of each model, the combination of Dropout(inclusion probability = 0.85) and L2 Penalty (coefficient = 1e-3) regularization, resulted in better generalization, achieving a balance between the generalization gap (0.042) and validation accuracy(84.1%). This outcome can be attributed to the fact that when ap-

plied together, they complement each other, by reducing the model's reliance on some neurons and encouraging more generalized weights, leading to more controlled and balanced learning, enhancing the model's ability to generalize better to new data.

In summary, the experiments demonstrate that combining regularization techniques can improve generalization performance, supporting the hypothesis that such combinations can outperform individual techniques in reducing overfitting.] .

5. Conclusion

[Question 8 - The study aimed to explore the problem of overfitting by training different neural networks on the EMNIST dataset and analyzing various regularization methods to mitigate this issue. The findings indicate that increasing the complexity of a network can negatively impact its generalization performance on unseen data. Experiments with different regularization techniques were promising, with Dropout, L2 penalty, and their combination standing out.

Future research could explore broader regularization techniques such as early stopping and apply these in more complex network architectures. Overall, the study highlights the critical problem of overfitting and underscores the importance of regularization techniques in improving model performance.] .

References

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

Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016b. 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.