MLP Coursework 1

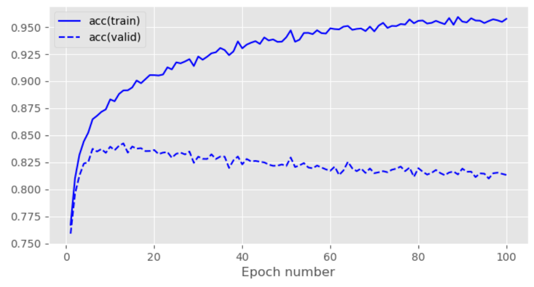
s1944632

Abstract

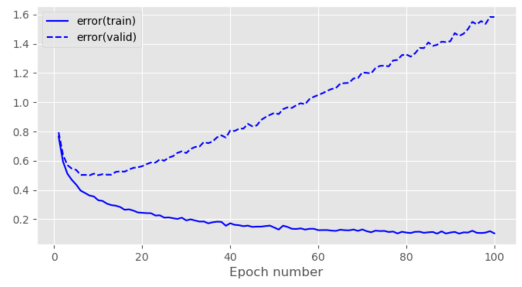
In this report we study the problem of overfitting, which is the training regime where performance increases on the training set but decrease on validation data. **[Overfitting trains the model parameters to be optimised only for the training test, i.e., it can accurately predict most of, if not all, of the training problem but performs very poorly on the test set.]** . 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 **[Increasing width reduces bias and increases variance, but after adding more than 64 units per layer, and increasing the number of layers (while keeping the number of units to 128), we keep overfitting the data.]** . Next we discuss why 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 [] . Finally, we briefly review a technique that studies use of weight decay in adaptive gradient algorithms, discuss its findings, and conclude the report with our observations and future work. Our main findings indicate that [Question 4] .

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 analyzing the given problem in Fig. 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. 2016) 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. 2016). We first explain them in detail and discuss why they are used for alleviating overfitting. In Section 4, we incorporate



(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.

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 [] . In Section 5, we study a related work that studies weight decay in adaptive gradient algorithms.[2] Finally, we conclude our study in section 6, noting that [Question 4] .

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

[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.

[2]Instructor note: Omitting this for this coursework, but normally you would be more specific and summarise your conclusions about that review here as well.

# Hidden Units	Val. Acc.	Train Error	Val. Error
32	78.5	0.5495	0.7149
64	79.7	0.3381	0.7541
128	80.0	0.1579	1.3245

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

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 is a problem as the model 'remembers' the results of the train data, ie, the weights are varied to the training set. As a result, the model performs very poorly on the test set.] .

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.

[Capacity of a network refers to the mappings a neural net can learn, when a neural net is unable to generalise and memorises the training data, it is said to have increased capacity.] .

Fig. 1a and 1b show a prototypical example of overfitting. We see in Fig. 1a that **[As we span through the epochs, at the 15th epoch, validation error is low and accuracy is high. However beyond this, as the neural net tries to learn and adjust its weights, the errors on the validation set keep increasing and on the training set it goes down. In the end, we get a model that is very accurate on the training set (nearly 95%) but performs poorly on the validation set.]** .

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

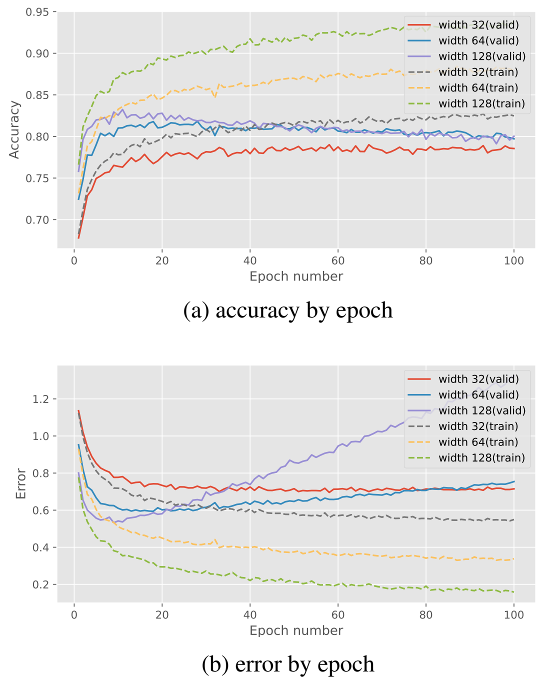


Figure 2. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network widths.

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. Fig. 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 **[As network width increases (number of units per layer), we see that the model improves between 32 and 64 units, but beyond this we see the accuracy on the validation set peak at 0.83 and then reduce to 0.82. However we can observe that the model achieves high accuracy in fewer epochs, this is because with more units the model has a higher capacity and reduced bias.]** .

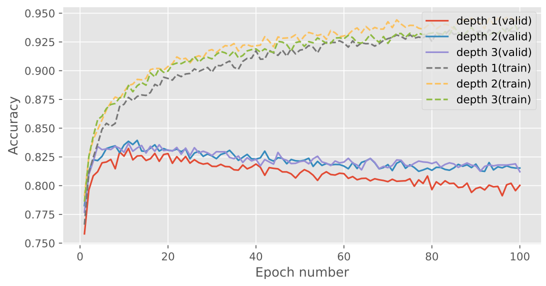
[] .

2.2. Network depth

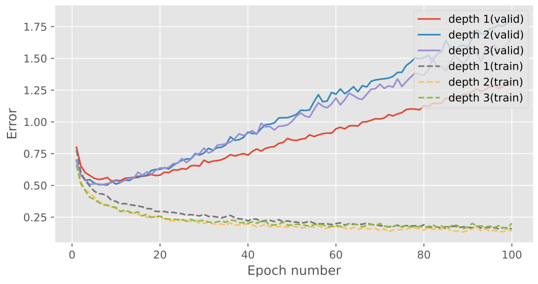
[Question Table 2 - Fill in Table 2 with the results from your experiments varying the number of hidden layers.] [Question Figure 3 - Replace these images with figures depicting the accuracy and error, training

# Hidden Layers	Val. Acc.	Train Error	Val. Error
1	80.0	0.1579	1.3245
2	81.4	0.1497	1.7849
3	81.9	0.1746	1.6380

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.

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 Fig. 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 **[The primary effect is the model takes too long to complete computation to achieve largely the same accuracy and high error. However the model overfits much more and its more accurate on the training set. As increasing network depth involves more sequential calculations, increasing depth has a greater impact on runtime than width.]** .

[Varying depth increases the dimensionality and hence a more non-linear decision boundary can be produced. However this increases overfitting and also takes more time to run, since a more sequential set of operations are performed. Magnitude of error for the training set reduce whereas it increases for the validation set. Accuracy increases only slightly.] .

[Overall varying width, performs faster than varying height. However, varying width produces a significant difference in accuracy for both the training set and validation set, increasing the number of layers doesn't. This is because depth increases variablity and increases the number of decision boundaries that can be produced, hence it] .

3. Dropout and Weight Penalty

In this section, we investigate three regularization 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:

$$\text{mask} \sim \text{bernoulli}(p) \quad (1)$$
$$\boldsymbol{y}' = \text{mask} \odot \boldsymbol{y} \quad (2)$$

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:

$$\frac{\partial \boldsymbol{y}'}{\partial \boldsymbol{y}} = mask \quad (4)$$

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:

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

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

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 regularization.

Weight penalty works by constraining the scale of parameters and preventing them to grow too much, avoiding overly sensitive behavior 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.

[A combination of L1 and L2 is called a an elastic net approach. L1 is used for data that has high number of features and it focuses on the medium of the data. L2 meanwhile can be used for complex data with high correlation, a combination would allow for the model to better set the weights, one that is immune to both outliers and still avoids being biased. We could make such a combination by first using L1 to perform feature selection, and also at the two ends of the network and using L2 to learn complex patterns in the middle of the network.] .

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 Figure 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.

Then, we apply the L1 or L2 regularization with dropout to our baseline and search for good hyperparameters on the validation set. We summarize 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.

[Question 14 - Explain the experimental details (e.g. hyperparameters), discuss the results in terms of their generalization performance and overfitting. Select and test the best performing model as part of this analysis] .

[Question 15 - Assume you were able to run 8 further experiments where you could combine Dropout and L1, and/or Dropout and L2 regularization. Which 8 runs would you pick and what question(s) would you aim to answer? Make sure you define the experiment setup, including any relevant hyperparameters] .

5. Literature Review: Decoupled Weight Decay Regularization

Summary In this section, we briefly study a method (Loshchilov & Hutter, 2019) that decouples the weight penalty from the weight update. The authors shed light onto the relation between weight decay and L2 weight penalty for SGD and Adam optimizers, pointing out that weight decay and L2 weight penalty are not equivalent when used with the Adam optimizer.

In particular, the authors claim that SGD with L2 weight penalty and weight decay are equivalent when their coefficients have the relation $\lambda' = \frac{\lambda}{2}$ (see their **Proposal 1**), which they prove with Equations (5) and (6) (proof is in their Appendix A). In addition, they claim that such a simple scalar relation does not hold in Adam optimization (see **Proposal 2**) and show, in their Appendix A, that the necessary relation for equivalence requires use of a preconditioner matrix $\boldsymbol{M}_t$. This requirement can be explained with the fact that **[Question 16 - Explain, in your own words, why the equivalence between L2 weight penalty and weight decay in Adam optimizer requires a more complex relation; a preconditioner matrix M_t rather than a scalar relation]** .

[Question 17 - Based on Algorithm 2 in (Loshchilov & Hutter, 2019), discuss whether your code with L2

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	82.9	0.450	0.509
	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	75.4	0.775	0.787
	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	83.8	0.418	0.470
	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 Regularization, L2 Regularization) and **bold** indicates the best overall.

Weight Penalty in the MLP library implements "Adam with L2 regularization", "Adam with decoupled weight decay (AdamW)" or another variation. Refer to the MLP code, pointing to the specific files and functions to justify your answer] .

Finally the authors validate their findings and proposed decoupled optimization strategies in various learning rate schedules, initial learning rates, network architectures and datasets. [3]

6. Conclusion

[Question 18 - Briefly draw your conclusions based on the results from the previous sections (what are the take-away messages?) and conclude your report with a recommendation for future directions] .

References

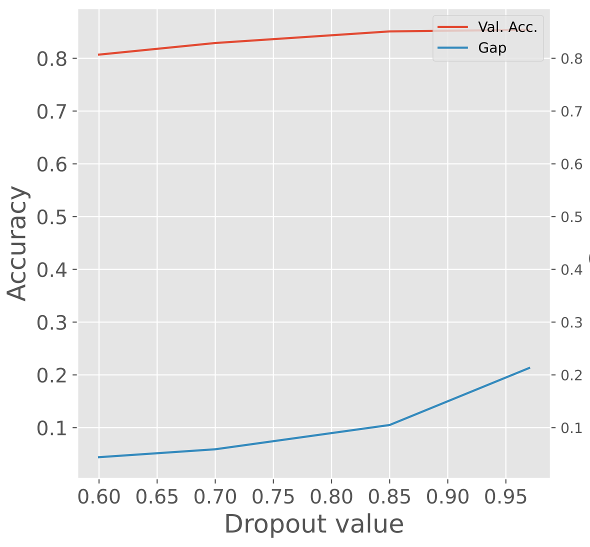
Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Loshchilov, Ilya and Hutter, Frank. Decoupled weight decay regularization. In *International Conference on Learning Representations (ICLR)*, 2019.

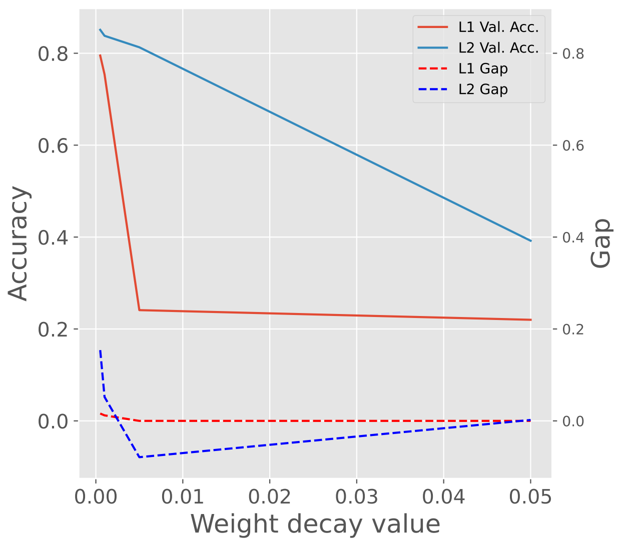
Ng, Andrew Y. Feature selection, l1 vs. l2 regularization, and rotational invariance. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 78, 2004.

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.

[3]Instructor note: Omitting this for this coursework, but normally you would give an evaluation of the experiment setup in (Loshchilov & Hutter, 2019) here.



(a) Accuracy and error by inclusion probability.



(b) Accuracy and error by weight penalty.

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