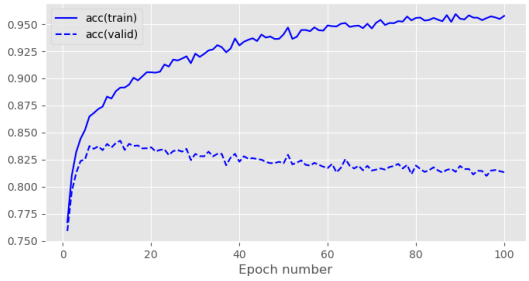
MLP Coursework 1

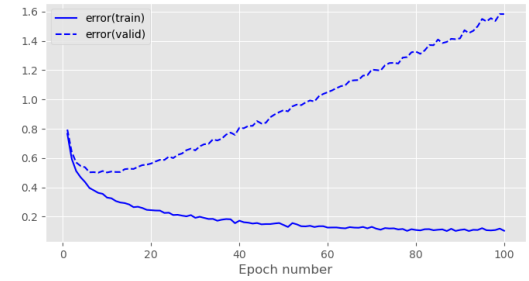
S2748897

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. 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 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. 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 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	78.7	0.558	0.703
64	81.3	0.340	0.662
128	80.5	0.163	0.924

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 that **[the baseline model reaches an almost not-improvable level of accuracy on the validation set after just 6 epochs. With an oscillating stepwise behaviour but clearly steady overall trend, this value (i.e., circa 0.835) starts then to decrease, scoring around 0.815 at the end of the training. The overfitting effects are evidently verified from epoch 13 on, as the accuracy on the train set keeps increasing at a slowing rate and the related error decreasing to a plateau of 0.1, reached only around epoch 80. Observing Figure 1b in particular, we recognize that the validation error's minima of 0.5 mirrors the highest-accuracy epochs' frame; the performance then degrades quite linearly and steeply, up to a generalization gap of 1.5]** .

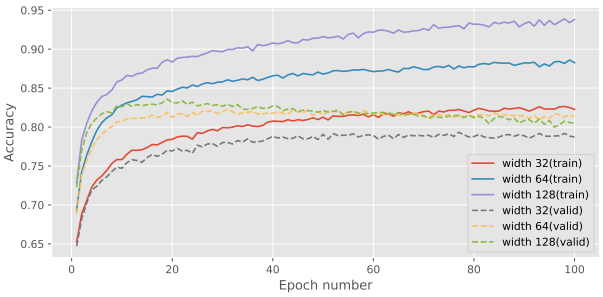
The extent to which our model overfits depends on many factors. For example, the quality and quantity of the training set and the complexity of the model. If we have sufficiently many diverse training samples, or if our model contains few hidden units, it will in general be less prone to overfitting.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

2.1. Network width

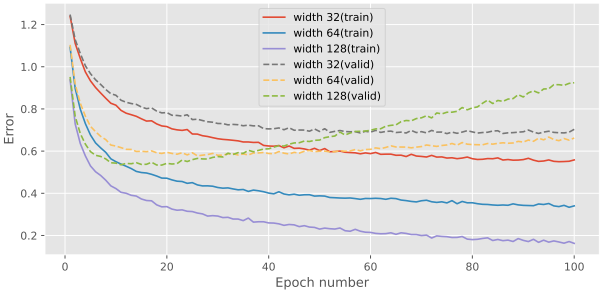
[Table 1 filled] [Figure 2 generated]

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



(a) accuracy by epoch



(b) error by epoch

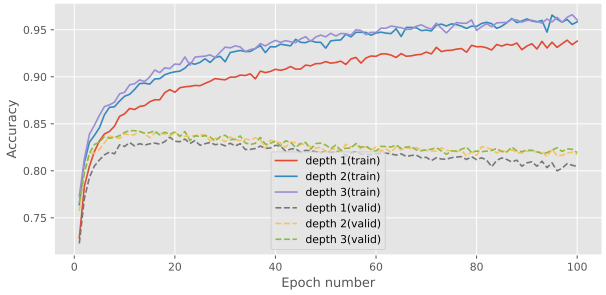
Figure 2. Training and validation curves in terms of classification accuracy (a) and cross-entropy error (b) on the EMNIST dataset for different network widths.

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 wider networks better perform on the test set, presenting constantly higher accuracy since the very first epoch and steeper curves in the following ones. This demonstrates a faster learning behaviour, which though leads these more complex models to rapidly enter the overtraining phase. Considering the validation scores, the 128-width network indeed reaches its best performance at epoch 19, with an error value (i.e., 0.53) lower than the ones ever reached through less hidden neurons. On the other hand, this same index starts then to increase quite linearly and very steeply, building-up a final generalization gap of circa 0.76. The 64-width training instead reaches the best validation-set performance around epoch 30 and loses accuracy at a considerably slower rate from there on. Finally, the thinner model does not even seem to start overfitting by the 100 epochs; its validation error presents a minimal variation along the last 40 epochs, highlighting how briefly the wider models' best-performance plateau phases last instead]** .

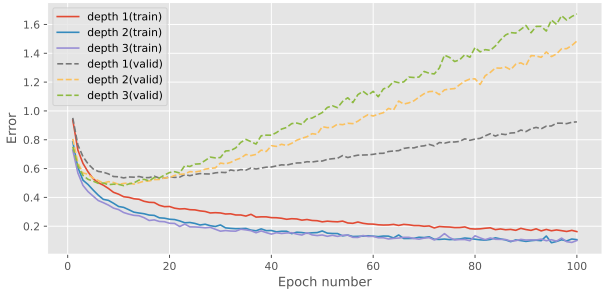
[Summarizing the above results, we are observing a totally expectable situation: more complex networks are capable of record accuracies on both the sets, but more prone to overfit data. In the attempt of always better perform on the train set, each model risks indeed to stop learning widely applicable patterns and memorize specific instances' characteristics instead; the

# Hidden Layers	Val. Acc.	Train Error	Val. Error
1	80.5	0.163	0.924
2	81.7	0.107	1.485
3	81.9	0.101	1.673

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.

switch between the first virtuous behaviour and the generalization-penalizing second one is simply happening faster and from earlier epochs for networks employing more parameters, due to a higher flexibility] .

2.2. Network depth

[Table 2 filled] [Figure 3 generated]

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 **[the results are totally comparable to the ones already discussed for the width-tuning experiment. In fact, models presenting more hidden layers – thus a higher number of fixable parameters at each iteration – outperform since the first epoch on the test set, but tend to overtrain earlier. Considering the validation scores in particular, the 3-depth network reaches its best performance at epoch 11, then the error starts to increase quite linearly and steeply to a final generalization gap of almost 1.6. The second deeper model instead starts overtraining around epoch 13, and still loses accuracy at a considerably faster rate with respect to the already analysed single-hidden-layer network. Finally, both the multiple-layer networks present a briefer and more oscillating best-performance phase]** .

[These conditions are verified because increasing the number of hidden layers empower the model with greater flexibility, which – as already stated in the width-increment analysis – makes the patterns' recognition faster and provokes earlier overfitting. However, in this case the latter dynamic is even better represented along the training iterations by the complete reversion in ranking of the best validation-performing models: the wider the network, the higher the record accuracy is and the faster that value is reached; but also the quicker is the generalization gap to grow, bringing the related error from the lowest to the highest value among the other less complex models' one] .

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:

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

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

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 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 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

[Table 3 filled]

[Figure 4 generated]

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.

[We can observe that by quite high inclusion rates the Dropout method leads to better performances on the validation set, avoiding overtraining and keeping the generalization gap low. Hyperparameter values beneath 80% – which strongly reduce inter-layers dependencies by more frequent neuron deactivations – cause the error on both the data sets to increase, due to a considerably hampered learning ability. Up to a certain level, dropping out units pushes instead the network to conceive more global patterns and give them a sensible importance. In particular, with the 0.97 inclusion rate we obtain a record validation accuracy of 85.4%, combined with a still very low final generalization gap which highlights a functional overfitting avoidance. Applying this model on the EMNIST test-set, we obtain comparably low errors and an accuracy of 84.5%. Moreover, it is important to notice that in this case label smoothing allowed comparably satisfying results through a definitively less computation-expensive method; here the outstanding train error reported is obviously caused by the intrinsic labels' distortion procedure.

Penalty-based strategies led instead to considerably different best-performances depending on the formulation applied: L2 reached an accuracy of 85.1%, while L1 did not overstepped the baseline's 0.8-level with the hyperparameter settings considered. However, both the penalty methods operated better regularizations with

Model	Hyperparameter value(s)	Validation accuracy	Train Error	Validation Error
Baseline	-	83.7	0.241	0.533
Dropout	0.6	80.7	0.549	0.593
	0.7	83.2	0.446	0.506
	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	74.8	0.843	0.849
	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	0.366	0.457
	5e-3	81.3	0.586	0.607
	5e-2	39.2	2.258	2.256
Label smoothing	0.1	*85.2*	*1.013*	*0.552*

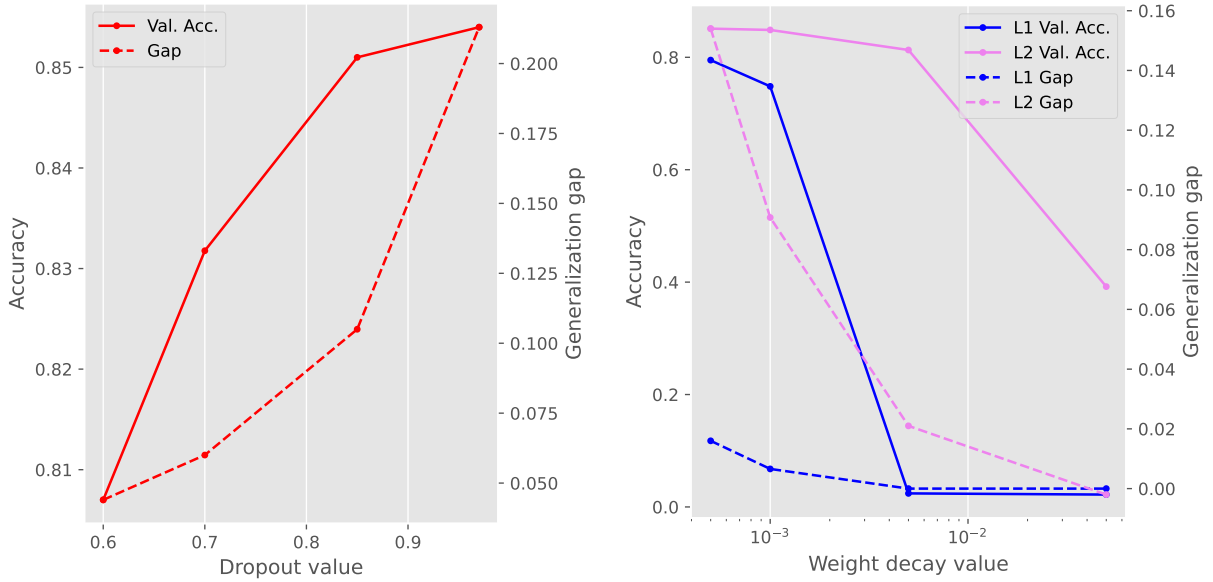
Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series (Dropout, L1 Regularisation, L2 Regularisation, Label smoothing) and **bold** indicates the best overall.

lower λ-values and produced instable training curves (i.e., presenting chaotically oscillating error and accuracy trends on both the data sets) for hyperparameter settings above 1e-3 for L1 and 5e-3 for L2. This performances align in general with the better adaptability of L2 – uniformly scaling the weights – to tasks with an high number of relevant features and training examples, in opposition to the more simplistic L1 (Ng, 2004)] .

[The Dropout method is often used together with weight penalties to obtain better regularization effects; it would thus be interesting to run further experiments combining these two strategies and tuning again their hyperparameters. For this aim, it is reasonable to keep the same baseline (i.e., 784-128-128-128-47 ReLU-activation network with learning rate of 1e-4 and 100-epochs training) in order to more directly compare the results with the previous analysis. The inclusion rate would then be set alternatively at 0.97 and 0.85: the two best performing values, which would not be sensible to increase beyond when implementing another regularization operation in parallel. With both these dropout hyperparameters, L1 could be applied with a λ of 5e-4 and with a lower one (e.g., 1e-4), respectively to test this penalty strategy's compatibility with Dropout – searching for a validation accuracy higher than both the baseline's and the relevant Dropout instance's ones – and the eventuality of a substantial further performance improvement at lower λ values. For similar reasons, L2 could instead be combined to Dropout by using its current best hyperparameter of 5e-4 and a lower explorative one of 1e-4. Slightly better results are expected in particular for the latter case, given the Dropout-L2 combined strategy's outcomes (i.e., test classification error around 23% lower than penalty-only regularization's one) with bigger ReLU architectures on the MNIST dataset (Srivastava et al., 2014)] .

5. Conclusion

[The above analysis explored the problem of overfitting in small-sized neural networks and drew a direct relation between the appearance of this phenomenon and the models' complexity level, here represented by the number and width of the hidden layers. However, Dropout and two penalty-based strategies are found functional to preserve networks' generalization ability; as a direct trade-off, these methods increase training time at a widely studied rate (Srivastava et al., 2014)(Ng, 2004), making label smoothing a valid and quick alternative in some cases. Moreover, combining multiple regularization methods is often a solution to even better preserve the general-patterns recognition behaviour: due to its uniform-penalty rationale, L2 is especially used together with Dropout for complex classification tasks. Other methods not discussed are the record-performing Dropout and Max-norm co-regularization (Srivastava et al., 2014) and some hybrid algorithms particularly useful for training convolutional networks (e.g. the one based exactly on Dropout and L2-Penalty which is applied on MNIST by Xie et al. 2022). Particularly in the latter field – employing very complex models and currently spreading across various fields where accuracy is of main importance – an interesting direction for future work would be to refine and speed up the combined regularization strategies, considering each method adaptability to different tasks and model architecture to research a common state-of-art framework applicable to any specific case] .



(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).

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.

Xie, Xiaoyun, Xie, Ming, Moshayedi, Ata Jahangir, and Skandari, M. H. A hybrid improved neural networks algorithm based on l2 and dropout regularization. *Mathematical Problems in Engineering*, 2022:1–19, 11 2022. doi: 10.1155/2022/8220453.