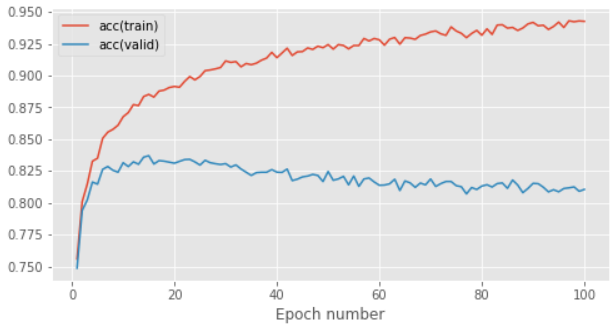
MLP Coursework 1

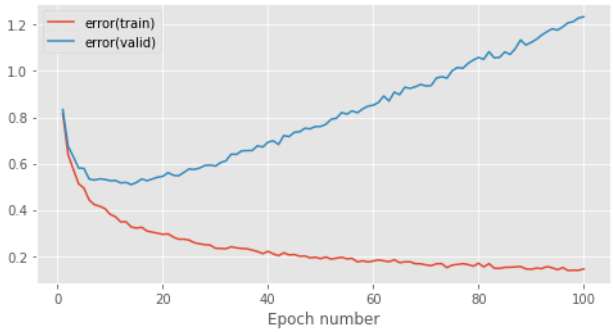
s2213361

Abstract

In this report we study the problem of overfitting, which is **[the concept where the model fits too closely to the training data, resulting in the loss of ability to make reliable predictions]** . 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 **[Question 2 - Summarise the effect increasing width and depth of the architecture had on overfitting]** . 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 **[Question 3 - Summarise what your results show you about the effect of the tested approaches on overfitting and the performance of the trained model]** . Finally, we briefly review another method, Maxout, discuss its strengths and weaknesses, and conclude the report with our observations and future work. Our main findings indicate that **[Question 4 - Give your overall conclusions]** .



(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 concept where the model fits too closely to the training data, resulting in the loss of ability to make reliable predictions]** . 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 each of them and their various combinations to a three hidden layer neural network, train it on the EMNIST dataset, which contains 131,600 images of characters and digits, each of size 28x28 which are split into 47 classes, grouping together some difficult to distinguish characters. 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 **[Question 3 - Summarise what your results show you about the effect of the tested approaches on overfitting and the performance of the trained model]** . In Section 5, we discuss a related work on Maxout Networks and highlight its pros and cons.[1] Finally, we conclude our study in section 6, noting that **[Question 4 - Give your overall conclusions]** .

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] 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.	generalization gap
32	78.5%	0.152
64	80.6%	0.342
128	80.2%	0.840

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

overfitting when **[it contains more parameters than what the data should require, resulting in the model's sensitivity to the irrelevant details and failure to project the general trend]** .

[Overfitting occurs when the same model is trained to exclusively maximizing performance on the training data repeatedly, causing the model to essentially "remembering" the training data and unable to identify generalizations within. This can usually be identified by a low error on training data and a high error on unseen data. During training, it can also be identified by the increase in error on validation data] .

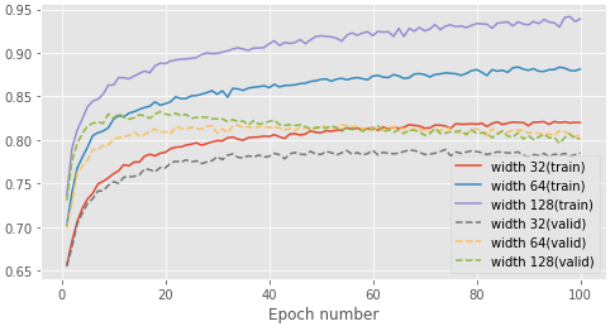
Fig. 1a and 1b show a prototypical example of overfitting. We see in Figure 1a that **[after around epoch 16, model's ability to predict training data continues to improve while accuracy for validation data gradually decreases. Figure 1b tells the same story from a different perspective—the error for training data decreases gradually throughout the process, but the error for unseen data rises drastically. These two graphs show that after around epoch 16, the model become too accustomed to the training data, losing the ability to identify common attributes within that would help it process new data]** .

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 a lot of varied training samples, or if our model is relatively shallow, it will in general be less prone to overfitting. Any form of regularisation will also limit the extent to which the model overfits.

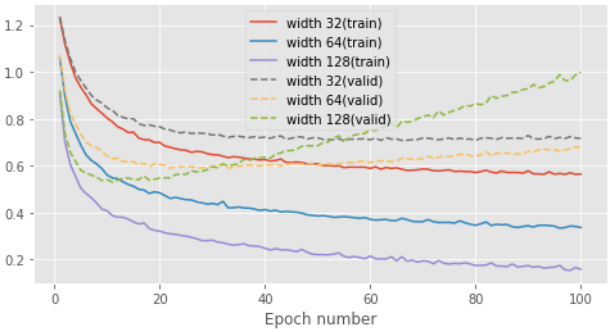
2.1. Network width

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 10^{-3} 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 and generalization gap. We observe that **[validation data accuracy rises consistently across all 100 epoches for the model with 32 neurons in the hidden layer, while the same attribute**



(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.

for the models with 64 and 128 neurons increases much faster in the beginning but dipped after epoch 31 and 16, respectively. Figure 1b tells the same story from a different perspective—the error for training data decreases gradually throughout the process, but the error for unseen data rises slightly for the model with 64 neurons and drastically for that with 128. These two graphs show that the model with 32 neurons seem to be on a good trajectory to becoming a successful model, while the one with 64 neurons is on the edge of overtraining. For the model with 128 neurons, the drastic increase in error after epoch 16 indicates that the model is undeniably overcomplicated, resulting in the model's excessive affinity towards the training data, losing the ability to make useful predictions] .

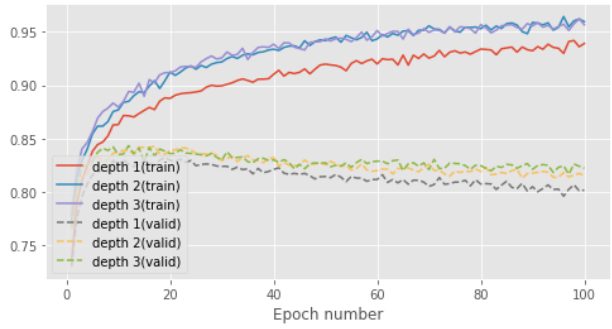
[Figure 2 and Table 1 shows that while increasing the hidden layer width can increase model performance during initial training, evidented by the much faster fitting, it can also hinder the resulting model due to overfitting. This result is expected because a more complicated network is more prone to overfitting] .

2.2. Network depth

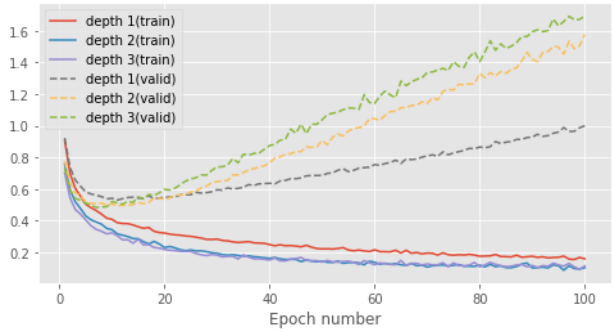
Next we investigate the effect of varying the number of hidden layers in the network. Table 2 and Fig. 3 shows the results from training three models with one, two and three hidden layers respectively, each with 128 ReLU hidden units. As with previous experiments, they are trained with

# hidden layers	val. acc.	generalization gap
1	80.2%	0.840
2	81.5%	1.478
3	82.2%	1.578

Table 2. Validation accuracy (%) and generalization gap (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.

the Adam optimizer with a learning rate of 10^{-3} and a batch size of 100.

We observe that **[Question 10 - Explain your network depth experiment results by using the relevant figure and table]** .

[Question 11 - Discuss whether varying depth affects the results in a consistent way, and whether the results are expected and match well with the prior knowledge (by which we mean your expectations as are formed from the relevant Theory and literature)] .

[Question 12 - Compare and discuss how varying width and height changes the performance and overfitting in your experiments] .

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 dropout rate. Dropout is commonly represented by an additional layer inserted between the linear layer and activation function. Its forward propagation during training is defined as follows:

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

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

where $y, y' \in \mathbb{R}^d$ 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 parameter of 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 rate p:

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

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:

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

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 features between layers so that the neurons of the next layer will not specifically depend on some features from of the previous layer. Instead, it force the network to 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. **[Question 13 - Explain L1/L2 weight penalties first in words and then with formulas. Explain how they are incorporated to training and what hyperparameter(s) they require]** .

[Question 14 - Discuss how/why the weight penalties may address overfitting, discuss how L1 and L2 regularization differ and support your claims with references where possible] .

4. Balanced EMNIST Experiments

[Question Table 3 - Fill in Table 3 with the results from your experiments varying the hyperparameter values for each of L1 regularisation, L2 regularisation, and Dropout (use the values shown on the table) as well as the results for your experiments combining L1/L2 and Dropout (you will have to pick what combinations of hyperparameter values to test for the combined experiments; each of the combined experiments will need to use Dropout and either L1 or L2 regularisation; run an experiment for each of 8 different combinations). Use *italics* to print the best result per criterion for each set of experiments, and bold for the overall best result per criterion.

]

[Question Figure 4 - Replace these images with figures depicting the Validation Accuracy and Generalisation Gap for each of your experiments varying the Dropout rate, L1/L2 weight penalty, and for the 8 combined experiments (you will have to find a way to best display this information in one subfigure).

]

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 in EMNIST as shown in section 2. We follow the previous training settings where we deliberately let the baseline overfit on the training set as in previous experiments. These settings ensure the fairness of the evaluation of three methods to alleviate overfitting. 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 15 - Explain the experimental details (e.g. hyperparameters), discuss the results in terms of their generalization performance and overfitting] .

5. Literature Review: Maxout Networks

Summary of Maxout Networks In this section, we briefly discuss another generalization method: Maxout networks (Goodfellow et al., 2013). This paper further explores the dropout method and proposes a new "maxout" layer which can complement dropout. The authors evaluate the performance of Maxout Networks in four standard datasets, namely MNIST, CIFAR-10 and 100, and SVHN. They point out that although dropout has been widely applied in deep models, **[it remains unproven whether dropout performs model average in these cases, and it may be worthwhile to design a model specific fordropout instead of applying it on arbitrary models to maximize performance]** . Following this motivation, they propose the Maxout activation layers. These can be considered learnable activations that work as a universal convex function approximator. The Maxout layer first maps the hidden space to k subspaces through independent affine transformations, then, for each element in output vectors, it takes the maximum value across all subspaces.

[Maxout is fully compatible with dropout. The dropout mask can be applied directly to the layer input just as it is done on conventional models. As such, the dropout mask dictates whether a neuron is included or excluded across all subspaces, and selecting the maximum across all subspaces still preserves the input's dropout status. Plus, because Maxout takes the maximum of value across the subspaces, an input can be mapped to different piecewise functions by changing the dropout mask, isolating the input from local noises] .

Strengths and limitations The author proposed a novel neural activation unit that further exploits the dropout technique. **[Question 18 - Give an overview of the experiment setup in (Goodfellow et al., 2013) and analyse it from the point of view of how convincing their conclusions are]** .

Although the Maxout activation units can maximize the averaging effect of dropout in a deep architecture, we can argue that the Maxout computation is expensive. The advantage of dropout lies in its high scalability and computational advantages. It can be arbitrarily applied to various network structures, and the calculation speed is fast, which is very suitable for heavy computing algorithms such as training and inference of neural networks. In comparison, the design of the Maxout network needs to project the hidden vector into k subspaces. Both the forward algorithm and the backward algorithm of dropout can be calculated in $O(D)$ complexity, but the complexity of Maxout is $O(kD)$. This can lead to increasing the number of training epochs needed to reach convergence. Furthermore, the universal approximation property of Maxout seems powerful, but it would be interesting to verify that it is useful in practice. Specifically, we can design an experiment where we increase the number of subspaces k and see where performances stop improving. In extreme cases, it is even possible that the function learned is too specific to the training data, effectively causing overfitting.

6. Conclusion

[Question 19 - 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] .

Model	Hyperparameter value(s)	Validation accuracy	Generalization gap
Baseline	-		
Dropout	0.1		
	0.3		
	0.5		
	0.7		
	0.9		
L1 penalty	1e-5		
	1e-4		
	1e-3		
	1e-2		
	1e-1		
L2 penalty	1e-5		
	1e-4		
	1e-3		
	1e-2		
	1e-1		
Combined	for example 0.95, L1 1e-6		
	?, ?		
	?, ?		
	?, ?		
	?, ?		
	?, ?		
	?, ?		
	?, ?		

Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series and **bold** indicate the best overall

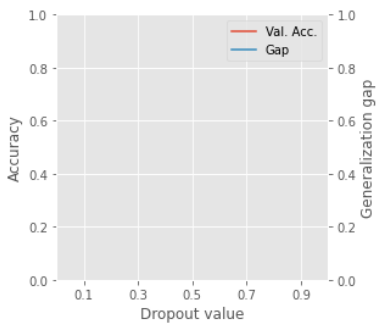
References

Goodfellow, Ian, Warde-Farley, David, Mirza, Mehdi, Courville, Aaron, and Bengio, Yoshua. Maxout networks. In *International conference on machine learning*, pp. 1319–1327. PMLR, 2013.

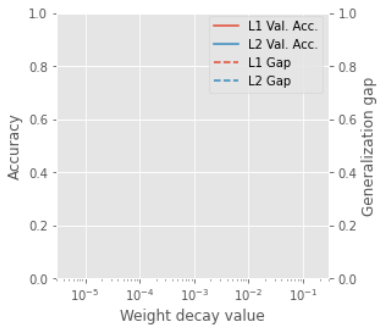
Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

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.

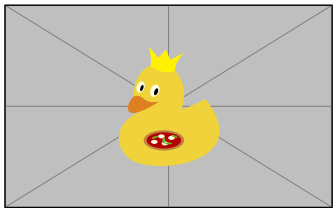
Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, Sutskever, Ilya, and Salakhutdinov, Ruslan. Dropout: a simple way to prevent neural networks from overfitting. *The journal of machine learning research*, 15(1): 1929–1958, 2014.



(a) Metrics by dropout rate



(b) Metrics by weight penalty



(c) Extra experiments

Figure 4. Hyperparameter search for every method and combinations