MLP Coursework 1

s2153223

Abstract

In this report we study the problem of overfitting, which is **[Question 1 - the problem that the model too accurately captures all the characteristics of the data, resulting in the inability to accurately predict the unknown data, making the model perform well in the training set and verification set, but poorly in 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 feed-forward architecture and observe that increasing width and depth **[Question 2 - can make the model prone to overfitting because they all make the model more complex and capture all the features of the data too accurately]** . 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 - when using the dropout method, as the hyperparameters increase, the validation accuracy of the model increases and the gap increases slightly. For the L1 regularization, the most suitable hyperparameter value is 1e-4. For the L2 regularization, the more suitable hyperparameter value is 1e-3. In combined models, we can find that the combination of dropout and weight penalty can achieve better results. When the hyperparameter of dropout is 0.99, and when L1 penaly is 1e-4, the generalization performance of the model is the best]** . 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 - investigating dropout method and the L1 and L2 weight penalties can alleviate the overfitting problem. Maxout is also an effective method]** .

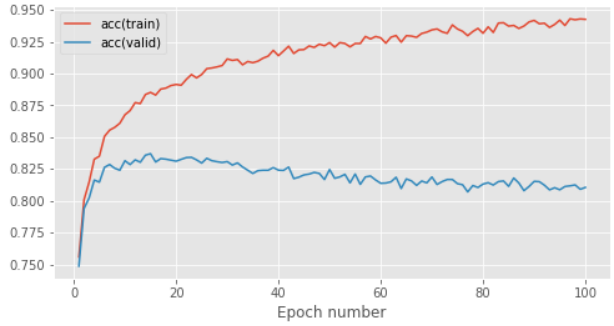
1. Introduction

In this report we focus on a common and important problem while training machine learning models known as overfitting, or overtraining, which is **[Question 1 - the problem that the model too accurately captures all the characteristics of the data, resulting in the inability to accurately predict the unknown data, making the model perform well in the training set and verification set, but poorly in the test set.]** . 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 - when using the dropout method, as the hyperparameters increase, the validation accuracy of the model increases and the gap increases slightly. For the L1 regularization, the most suitable hyperparameter value is 1e-4. For the L2 regularization, the more suitable hyperparameter value is 1e-3. In combined models, we can find that the combination of dropout and weight penalty can achieve better results. When the hyperparameter of dropout is 0.99, and when L1 penaly is 1e-4, the generalization performance of the model is the best]** . 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 - investigating dropout method and the L1 and L2 weight penalties can alleviate the overfitting problem. Maxout is also an effective method]** .

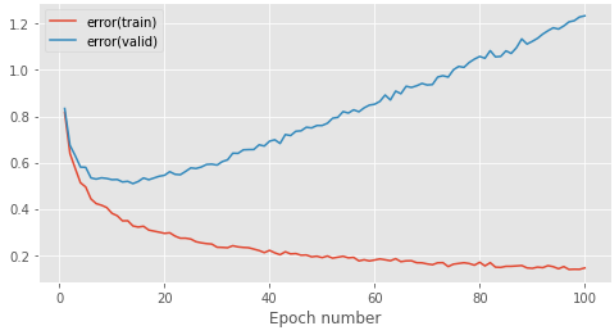
2. Problem identification

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 **[Question 5 - the error of the model is small in the training set and verification set, but large**

[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.



(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.

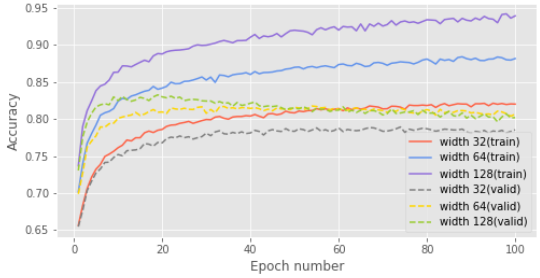
in the test set, and the generalization gap of the model is large] .

[Question 6 - When the model is too complex, that is, when the width and depth of the model are too large, over fitting will occur. This is because the model captures all the features of the data too accurately, which leads to the use of too many features of the original data when using the model to predict the unknown data, resulting in the increase of test error. When the training error of a model becomes smaller and the test error becomes larger, the gap between training error and test error becomes larger and larger. We can judge that the model is over fitted] .

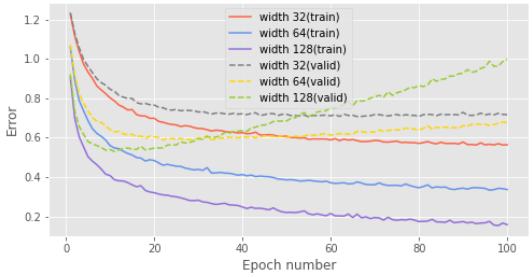
Fig. 1a and 1b show a prototypical example of overfitting. We see in Figure 1a that **[Question 7 - the accuracy of the training set increases gradually and the error decreases gradually, but the accuracy of the valid set increases in the first 10 epochs and then decreases gradually, and the error decreases in the first 10 epochs and then increases continuously. Since the accuracy curve of the valid set reaches the maximum at about the 10th epoch and the error curve reaches the minimum at about the 10th epoch, and then the error becomes larger, it is speculated that the model has been over fitted after about 10 epochs are trained. With the increase of epoches, the accuracy of the valid set decreases significantly and the error increases significantly, so the over fitting trend is more and more obvious]** .

# hidden units	val. acc.	generalization gap
32	78.561%	0.152
64	80.622%	0.342
128	80.245%	0.842

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



(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 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

[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 10^{-3} and a batch

# hidden layers	val. acc.	generalization gap
1	80.378%	0.827
2	81.244%	1.516
3	82.353%	1.502

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

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 **[Question 8 - when the model has only one hidden layer, as the hidden units increase from 32 to 64 and then to 128, the valid accuracy of the model first increases and then decreases, and the generalization gap of the model first decreases and then increases. We can see from Figure 2 that for the model with 32 hidden units, the valid accuracy curve increases with the increase of epoches, and the valid error curve decreases with the increase of epoches. There is no overfitting problem. However, for models with 64 hidden units and 128 hidden units, their valid accuracy first increases and then decreases, and valid error first decreases and then increases, indicating that both models have overfitting problems]** .

[Question 9 - We can find from *Figure 2* that models with different widths do not consistently affect the results. The model with 64 hidden units has the smallest value of valid accuracy and valid error. The model with 32 hidden units is not as wide as the model with 64 hidden units, so the valid accuracy is lower and the valid error is higher. Although the model with 128 hidden units has the largest width, due to over fitting problems, the value accuracy decreases and the value error increases, resulting in poor performance. The results meet our expectations and match well with the prior knowledge] .

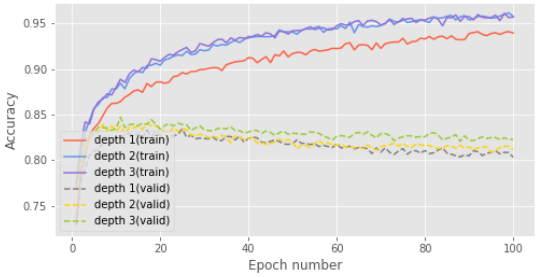
2.2. Network depth

[Question Table 2 - Fill in Table 2 with the results from your experiments varying the number of hidden layers.

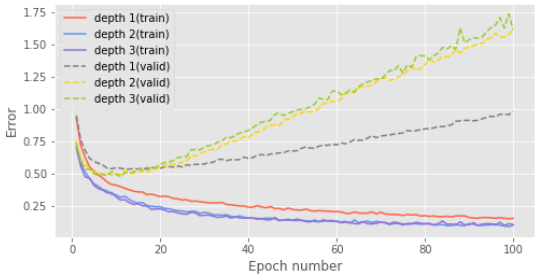
] [Question Figure 3 - Replace these images with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden layers.

]

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



(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.

units. As with previous experiments, they are trained with the Adam optimizer with a learning rate of 10^{-3} and a batch size of 100.

We observe that **[Question 10 - When hidden units is 128, models with different depths will have over fitting problems. Among them, the model with one hidden layer has the lowest valid accuracy, the generation gap is the smallest, the model with two hidden layers has the medium valid accuracy, the generation gap is the medium, and the model with three hidden layers has the highest valid accuracy and the generation gap. The generalization gap of the model with more hidden layers is significantly higher than that of the model with less hidden layers, and the over fitting problem is more obvious]** .

[Question 11 - We can find from *Figure 3* that models with different depths consistently affect the results. The model with one hidden layer has the minimum value of valid accuracy and generalization gap. For the model with two hidden layers, the value of valid accuracy and generalization gap are improved. For the model with three hidden layers, the value of valid accuracy and generalization gap are the highest among the three models. With the increase of model depth, the over fitting problem becomes more significant. The results meet our expectations and match well with the prior knowledge] .

[Question 12 - When the model has only one hidden layer and the width is 32 hidden units, the model is not fitted. When the width becomes 64 hidden units, the model is slightly over fitted, and when 128 hidden units

are included, the model is obviously over fitted. This shows that the wider the model width is, the easier the model is to have over fitting problems, and the more obvious the over fitting is with the increase of the width. When the model width is 128 hidden units, the models with different hidden layer layers are over fitted, and the over fitting performance becomes more and more obvious with the increase of model depth. Among all models, the model with the largest depth and width has the most serious over fitting problem] .

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 by the inverse of the dropout 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 - L1 regularization is also known as Lasso regression, the regularization term $\Omega(\theta) = \|w\|_1 = \sum_i |w_i|$ (Goodfellow et al., 2013). The regularized objective function $\widetilde{J}(\omega; \underline{\mathbf{X}}, y)$ is given by**

$$\widetilde{J}(\omega; \underline{\mathbf{X}}, y) = \alpha\|w\|_1 + J(\omega; \underline{\mathbf{X}}, y), \tag{5}$$

with the corresponding gradient

$$\nabla_w \widetilde{J}(\omega; \underline{\mathbf{X}}, y) = \alpha sign(\omega) + \nabla_w J(\omega; \underline{\mathbf{X}}, y). \tag{6}$$

L1 regularization will make the elements of the original optimal solution offset by different amounts, and make some elements 0, resulting in sparsity.

L2 regularization is also known as Ridge regression,the regularization term $\Omega(\theta) = \frac{1}{2}\|w\|_2^2$. The regularized objective function $\widetilde{J}(\omega; \underline{\mathbf{X}}, y)$ is given by

$$\widetilde{J}(\omega; \underline{\mathbf{X}}, y) = \frac{\alpha}{2}\omega^\top\omega + J(\omega; \underline{\mathbf{X}}, y), \tag{7}$$

with the corresponding parameter gradient

$$\nabla_w \widetilde{J}(\omega; \underline{\mathbf{X}}, y) = \alpha\omega + \nabla_w J(\omega; \underline{\mathbf{X}}, y). \tag{8}$$

The effect of L2 regularization is to enlarge and shrink each element of the original optimal solution in different proportions.

hyperparameters includes learning rate and penalty coefficient for the Weight Penalty functions. Penalty coefficient is a manually selected float value, usually in the range of 0.1-0.00001] .

[Question 14 - The weight penalities can address overfitting is because they add a regularization term after the loss function, which is equivalent to adding constraints on the basis of the original objective function and limiting the parameters of the model. The less the parameters of the model, the simpler the model is. Therefore, the weight penalties can solve the over fitting problem.

L1 regularization and L2 regularization are mainly different in the following points:

1. L1 regularization adds the sum of absolute values of various parameters of the model after loss function. L2 regularization adds the square value of the square sum of each parameter of the model after the loss function.

2. L1 tends to produce a small number of features, while other features are 0. Because the optimal parameter value is very likely to appear on the coordinate axis, it will lead to a one-dimensional weight of 0, resulting in a sparse weight matrix, which is easy to obtain a sparse solution. L2 will select more features, which will be close to 0. The optimal parameter value has a small probability of appearing on the coordinate axis, so the parameter of each dimension will not be 0. When $||w||$ is minimized, each term will approach 0, and the solution will be smoother.

3. L1 regularization will make the elements of the original optimal solution offset by different amounts, and make some elements 0, resulting in sparsity. The effect of L2 regularization is to enlarge and shrink each element of the original optimal solution in different proportions

4. Because of the "sparse solution" characteristic of L1 regularization, L1 is more suitable for feature selection to find out the more key features and set some less important features to zero. L2 regularization can produce many models with small parameter values, that is, these models have strong anti-interference ability and can adapt to different data sets and extreme conditions] .

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 - In the experiment, the validation accuracy of baseline is 82.315 %, and the generalization gap is 1.555. The model is over fitted. After using the dropout method, the generation gap is significantly smaller. When different hyperparameters of 0.1-0.9 are selected, the generation gap is less than 0.2. When the hyperparameter is 0.1, the generalization gap is only 0.002. Therefore, we can judge that the dropout method can solve the over fitting problem. From the perspective of generalization performance, the validation accuracy of the model increases with the increase of hyperparameters and gap becomes slightly larger. When the hyperparameter is 0.7, the accuracy exceeds the accuracy of baseline, which is 83.303 %. When the hyperparameter is 0.9, the validation accuracy is the highest in the dropout method, reaching 85.778 %.

In addition, after using the weight penalty method, the generalization gap of the model is significantly smaller, the minimum gap is less than 0.001 and the maximum gap is 1.203. As can be seen from table 3, the accuracy of the model using L1 penalty first increases and then decreases with the increase of hyperparameters. When the hyperparameters are 1e-5 or 1e-4, the validation accuracy is higher. When the hyperparameters are 1e-4, the model has the highest accuracy, which is 84.854%. However, when the hyperparameters are 1e-5, the gap is large. Therefore, for the L1 penaly method, the most suitable hyperparameter value is 1e-4. For L2 penalty, we can see from table 3 that with the increase of L2 hyperparameters, the accuracy of the model first increases and then decreases sharply, while the generation gap decreases continuously. When the hyperparameters are 1e-5 and 1e-4, the gap of the model is high, so the over fitting phenomenon is not well improved. However, when the hyperparameters are 1e-3, the accuracy of the model reaches the highest of the five groups of results, and the gap is only 0.070, indicating that the over fitting problem can be solved. Therefore, using the L2 penalty method, we can preliminarily determine that the most suitable hyperparameter value is 1e-3.

In the combined model, we can further confirm the speculation about the selection of hyperparameters. When

Model	Hyperparameter value(s)	Validation accuracy	Generalization gap
Baseline	-	82.315%	1.555
Dropout	0.1	4.854%	*0.002*
	0.3	62.139%	0.006
	0.5	77.544%	0.017
	0.7	83.303%	0.055
	0.9	*85.778%*	0.191
L1 penalty	1e-5	83.329%	0.794
	1e-4	*84.854%*	0.098
	1e-3	76.171%	0.011
	1e-2	2.013%	*0.001*
	1e-1	1.981%	*0.001*
L2 penalty	1e-5	82.782%	1.203
	1e-4	83.487%	0.518
	1e-3	*84.620%*	0.070
	1e-2	76.101%	0.009
	1e-1	2.044%	**0.001**
Combined	for example 0.95, L1 1e-6	85.696%	0.306
	0.95, L1 1e-4	85.810%	0.046
	0.95, L2 1e-3	85.133%	0.044
	0.90, L1 1e-4	84.519%	0.031
	0.99, L1 1e-4	**85.867%**	0.072
	0.99, L1 1e-6	83.108%	0.724
	0.99, L2 1e-3	85.532%	0.058
	0.90, L2 1e-3	84.152%	*0.028*

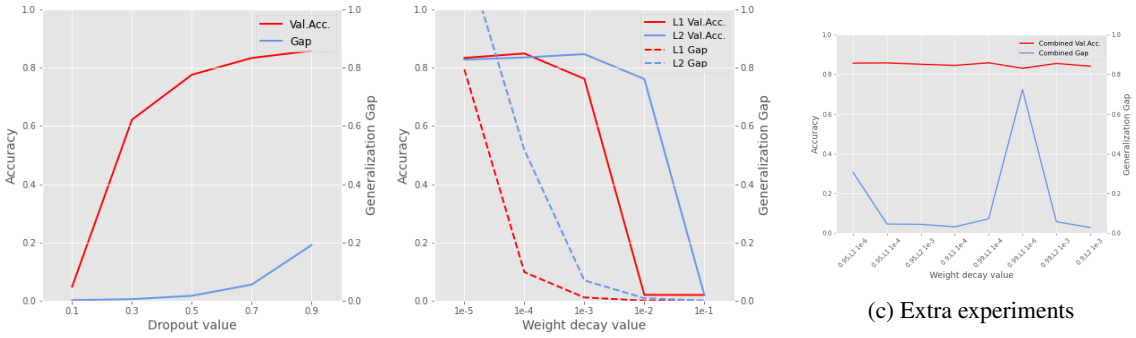
Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series and **bold** indicate the best overall

the hyperparameter of dropout is 0.99, and when L1 penaly is 1e-4, the generalization performance of the model is the best, the validation accuracy reaches 85.867 %, and the generalization gap of the model is 0.072 without over fitting. In addition, when the hyperparameter of dropout is 0.95 and L1 penaly is 1e-6, the model is slightly over fitted. When the hyperparameter of dropout is 0.99 and the L1 penalty is 1e-6, the model is over fitted. Therefore, the hyperparameters of the fifth group of experiments are the best] .

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, **[Question 16 - it has not previously been demonstrated to actually perform model averaging for deep architectures. They argue that rather than using dropout as a slight performance enhancement applied to arbitrary models, the best performance may be obtained by directly designing a model that enhances dropout's abilities as a model averaging technique. The model is designed to both facilitate optimization by dropout and improve the accuracy of dropout's fast approximate model averaging technique]** . 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.

[Question 17 - Dropout is compatible with Maxout because Dropout is a network regularization technique, which is equivalent to training many different network structures, but the parameters of different structures are shared, because in fact, only one network exists. Maxout is a kind of neural network activation function. There is no nonlinear activation function in the affine transformation of Maxout model. Since there is only one network in dropout and the weight of the model is multiplied by the Dropout ratio p, this method performs fairly well in the linear activation function, so we can also introduce the dropout technique into this transformation. Moreover, the experimental results in the paper show that Maxout and Dropout have a good combination effect] .



(a) Metrics by dropout rate

(b) Metrics by weight penalty

(c) Extra experiments

Figure 4. Hyperparameter search for every method and combinations

Strengths and limitations The author proposed a novel neural activation unit that further exploits the dropout technique. **[Question 18 - They conducted Maxout experiments using MNIST, CIFAR-10, CIFAR-100, and SVHN data sets. On the permutation variant MNIST dataset, only methods that are regulated by modeling the input distribution output perform the Maxout MLP. In all experiments, the test error using Maxout + Dropout method is the lowest. At the same time, it can show that Dropout is crucial to Maxout and Dropout attaches a good approval to model averaging in deep models.]** .

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 - We studied the influencing factors and solutions of over fitting through EMNIST dataset. Over fitting is the problem that the model too accurately captures all the characteristics of the data, resulting in the inability to accurately predict the unknown data, making the model perform well in the training set and verification set, but poorly in the test set. The error of the model is small in the training set and verification set, but large in the test set, and the generalization gap of the model is large. The wider the model width and the deeper the depth, the easier it is to over fit the model.

After that, we successively investigate the drop out method and the L1 and L2 weight penalties to alleviate the overfitting problem. After using these methods. The generalization gap of the model is obviously smaller, and the validation accuracy is also improved. When using the dropout method, as the hyperparameters increase, the validation accuracy of the model increases and the gap increases slightly. For the L1 penalty method, the most suitable hyperparameter value is 1e-4. For L2 penalty method, the more suitable hyperparameter value is 1e-3. In combined models, we can find that the combination of dropout and weight penalty can achieve better results. When the hyperparameter of dropout is 0.99, and when L1 penaly is 1e-4, the generalization performance of the model is the best, the validation accuracy reaches 85.867 %, and the generalization gap of the model is 0.072 without over fitting.

Finally, we explore another method to alleviate over fitting, Maxout, and analyze the advantages and disadvantages of Maxout. Next, we can design experiments to explore the practicability of Maxout. When we increase the number of subspaces k and see where performances stop improving. We can also start from dropout and combine other methods to study the allow the overfitting problem.] .

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.