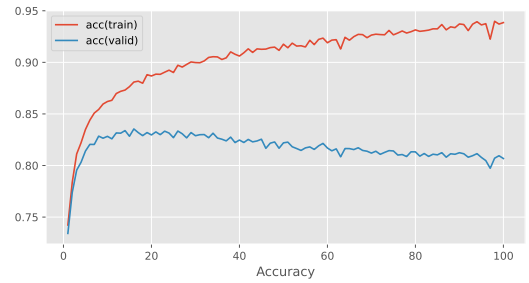
MLP Coursework 1

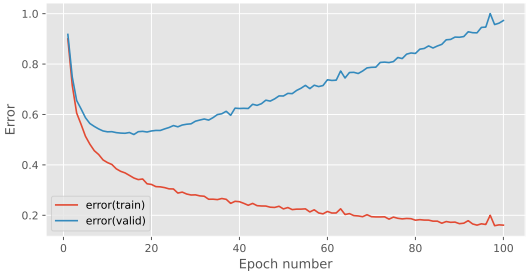
s1720422

Abstract

In this report we study the problem of overfitting, which is **[when a network function is too closely fit to a training set, and results in an increasing validaiton set error despite a monotonically decreasing training error(an illustration of this concept is shown in figure 1b).]** . 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 **[can lead to greater perfomrance on validation data, however also can result in poorer model gernealization due to an increased likelihood and rate of overfitting when the model is too flexible.]** . 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 **[when a network function is too closely fit to a training set, and results in an increasing validaiton set error despite a monotonically decreasing training error(an illustration of this concept is shown in figure 1b).]** . 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 overfitting when **[it's validation error begins to increase despite it's training error decreasing monotonically. This leads to an increasing generealization gap as the model is trained over a number of epochs. It inidcates that the model is too flexible and doesn't fit to unseen data very well.]** .

[Overfitting occurs due to a high variance being present within the trained model. This is due to the model becoming overly complex and fitting its degrees of freedom in such a way which closely resembles data points present in the training data. The parameters of the model learn the detail and noise in the training data which results in a negative performance when it comes to the gnerealization of the model and its ability to closely predict outputs on unseen data. Overfitting can occur for a number of reasons. If the learing rate of a model is increased too much, then it is likely that the parameters of the model will convege and resemble training data points very closely in a fewer number of epochs respectively. A clear way to identify if a model is overfit is by visualising the error of the model on a training and vaildation set across a fixed number of epochs. If the validation error begins to increase despite a monotonically decreasing training error, the model is said to be overfitted. To generalize, if the Model performs well on a training set, but poorly on unssen data, it is overfitted.] .

Fig. 1a and 1b show a prototypical example of overfitting. We see in Figure 1a that **[it highlights the ratio of total EMNIST samples which were classified with their target labels correctly over a period of 100 epochs. the red and blue lines rpresent the accuracy obtained by the baseline model across the training and test set respectively. It cna be observed that the accuracy of predictions steeply increased up until epoch 5. Beyond which training accuracy continued to increase steeply in contrast to the validation accuracy which increased at a slower rate and began to level off. At around epoch 16, it is clear that the model was no longer improving in validation accuracy due to a flat gradient in the validation accuracy curve. This indicates a maximum optimum accuracy of 82.5% was achieved for the model at epoch 16 before it's performance began to decline. Figure 1b similarly illustrates epoch 16 as the point where the cross-entropy error measured on the validation set was at it's minimum of 0.5, before it begian to increase. It can be deduced that beyond epoch 16, training accuracy continued to increase monotonically, whereas validation accuracy began to decline. A similar pattern can be observed for figure 1b where the error on the training set conintured to decrease beyond epoch 16, whereas it increased for the validatoin set. This led to an expanidng generalization gap. Hence, the baseline model became overfitted beyond epoch 16. By the end of epoch 100, it can be seeen that the accuracy across the validation set had fallen from 82.5% to 80.5%. Furthermore, the error across the validation set increased from a minimum of 0.5 to approximately 0.98. In contrast, by epoch 100, the training set was clearly fitted to the model due to the high training accuracy of 93.5% and low error of 0.17.]** .

[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	0.15
64	80.9	0.32
128	80.3	0.83

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

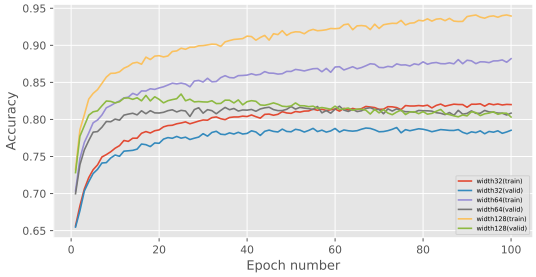
The extent to which our model overfits depends on many factors. For example, the quality and quantity of the training set and the complexity of the model. If we have a lot of varied training samples, or if our model is relatively shallow, it will in general be less prone to overfitting. Any form of regularisation will also limit the extent to which the model overfits.

2.1. Network width

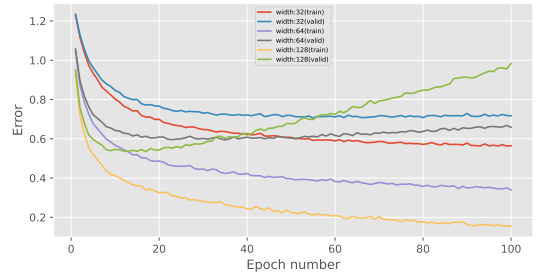
[Question Table 1 - Fill in Table 1 with the results from your experiments varying the number of hidden units.] [Question Figure 2 - Replace the images in Figure 2 with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden units.]

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 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 **[as the number of hidden units is increased, this does lead to an imprvement in validation accuracy, as shown in table 1. The final validation accuracy increased from 78.5% for 32 units to 80.3$ for 128 units. We observe that the highest final validation accuracy of 80.9% was achieved using 64 hideen units. The reason why the 128 uint model was unable to outperform the 64 units model is due To overfititing. Fig-**



(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.

ure 2a and 2b shows that overfitting of the 128 model is indicated by the rise in validation error(green line) beytond epoch 18, despite its's corresponding training error decreasing(yellow line). This led to the decrease in validation accuracy as shown in figure 2a for acccuraccies recorded beyond epoch 18. By the end of epoch 100, we can see that the 64 units mode perfromed almost identically to the 128 ubits model on the validation set. The 128 units model had an advantage when it comes to how quickly the model converged and obtained lower validation error and higher validation accuracies. It acheived a maximum validation accuracy of approximately 82.5% after epoch 18. It was also the fastest model to see a drastic improvment in validation accuracy. However due to overfitting, this afvantage didn't last too long. The 32 layer model perfomred the worst on the validation set and obtained the lwoest accuracy across the board. This is due to the model being underfit and not flexible enough to learn the parameters needed to fit the training data well. Despite this, it did obtain the lowest generalixation error of 0.15 so is the least likely model to result in overfiting. In contrast, the 128 unit model is the most likely to overfit due to a high geneeralization gap of 0.83.] .

[Varying the width affects the results in a consistent way as expected. We expect the model to start off underfitted to the training data when less units are used. This is because the model is less flexible due to having less free parameaters which can fit to to the data. This is represented well with the lowest validation accutacy obtained in table 1 for the 32 units model. As we increase

# hidden layers	val. acc.	generalization gap
1	81.2	0.78
2	81.5	1.46
3	82.0	1.58

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

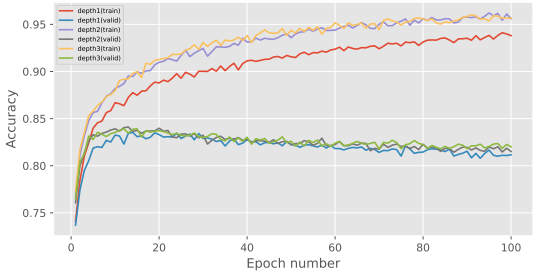
the number of units. we expect the model to fit the inpuit data better due to the model having increased flexibiliy. This is demonstrated successfully due to the 64 units model having obtained the best final vaildation accuracy of all the models. Furthermore, we hypthesised that increasing the Model flexibility too much and having a number of free paramters close to the number of samples being trained on would result in overfitting due to each free parameters closely fitting each training point, hence creating noise. This is clerarly demonstrated with the 128 units model in figure 2a where it's validation Accuracy began to decrease despiute the trianing accuracy increasing. It is also clear the model is overfit due to the increasing error generalization gap which had formed.] .

2.2. Network depth

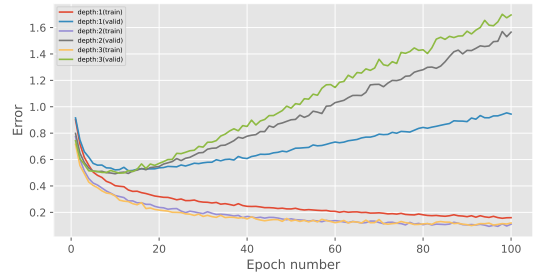
[Question Table 2 - Fill in Table 2 with the results from your experiments varying the number of hidden layers.] [Question Figure 3 - Replace these images with figures depicting the accuracy and error, training and validation curves for your experiments varying the number of hidden layers.]

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 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 **[across all 3 depths tested, the final validation accuracy of each model increased as the model depth increased. This is primarily due to the model converging quicker within a fixed amount of epochs as there are more parameters being fitted to the training data. The consequence of increaisng depth is an increased generalizxation error gap. This can be seen from the spread between the orange and green lines in figure 3a for the model of depth 3. It means although the final validation accuracy increases as depth increases, the amount the model is overift by increases too hence causes the model to generalize worse on unseen data. From figure 3b, we can see that the model of depth 1 has the smallest generalization gap, whereas the model of depth 3 has the largest. Furthermore, we can observe the effect of increasing depth to be marignal when it comes to validation accuracy. This can**



(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.

be explained by the model of depth 1 having a similar final validation accurtacy of 81.2% compared to the model of depth 2 and 3 which had accuraccies of 81.5% and 82.0% respectively. Therefore, from figure 3a and 3b we can conclude that increasing model complexity by increasing the depth is counter-intuitive to imporving the generalization and validation accuracy of the model. The model genrealizes better with a depth of 1, comparted to depth of 2 and 3. The validation acuttacy gain of using greater depths are marginal and makes for a worse generalized model.] .

[Vartying depth affects the reuslts in a consistent way as it increases the amount the model is overfited by. From the previous experiment, we couild see that using 128 hidden units for a single layer model already caused the model to become overfitted. By using 128 hidden units per layer, the model had twice and thrice the amount of parameters to fit to the input data for depths of 2 and 3 respectively. Therefore, the generalization gap increased in proportion to this. The results reflect what is expected as there are a lot more parameters being fitted to the training data, hence there will be more noise in the resulting function of the model. This is due to the number of paramters fitted far exceeding the number of samples being used to train the model on.)] .

[From the conducted experiments it can be observed that increaisng the width and depth of the model both reuslted in improved performance. The final accuracy on unseen data improved when the model complexity was increased by varying the width and depth. However, it can also be observed that increases in these parameters can increase the likelihood and rate of how quickly the model overfits to the training data. This is primarily due To the increase in model complexity causing the model to overfit to noise in the training data. As observed, there is a point where increasing model depth results in marginal gains in perfomrance on unssen data, but reults in poorer model generalization. Thus, to prevent overiftting, a compromise between width and depth needs to be reached to acheive an ideal balance between model performance and generalization.

] .

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. 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. **[L1/l2 regularisation are methods used to penalise the flexibility a model has by scaling the models parameters such that it becomes less flecible.**
L1 regualarization achieves this by adding a penalty term parameter to the error function being minimised when training a model. The penalty term added is dependent on the sum of the magnitude of weight parameters present within the model. Adding this penalty term reults in the model reducing small weights to 0 and retaining larger, more important wieghts that are fundamnetal for the model to generalize well, essentailly acting as a form of feature selection. L1 regularization can be expklained by the following equations:

$$E^n = E_0 + \beta \sum_w |w_i| \tag{5}$$

$$\frac{\partial E^n}{\partial w_i} = \frac{\partial E_0}{\partial w_i} + \beta sgn(w_i) \tag{6}$$

E^0 represents the unregularized error function which is added to the sum of the magnitude of all weights present in the model, scaled by a fixed coeefficent β. The scaling factor is chosen manually beforehand and is resposnible for controlling the magnitude of regularization to the weights. The seocnd equation shows that the updated gradient of the regularized wrror function with respect to weights now includes a temr dependent on the direction of the magnitude of weights. The new updated weight for the model will take this into account and move in a psoitve or negative direction proportional to the value of β.

L2 regularization (commonly known as weight decay) works in a similar way To L1 execpet that the penalty term added is dependent on the sum of the squared weights as opposed to the sum of the magnitude weights in it's L1 couunterpart. This reusxlts in the updated wights being dependent on the size of the weights. Therefore, the gradients shrink at a rate proportional to this. Thus, this means weightd are pusehjd to be as small as possible as long as the regularized cost function continues to decrease. L2 regularization can be desicrbed by the following equations:

$$E^n = E_0 + \frac{\beta}{2} \sum_w {w_i}^2 \tag{7}$$

$$\frac{\partial E^n}{\partial w_i} = \frac{\partial E_0}{\partial w_i} + \beta w_i \tag{8}$$

From the seocnd equation, we can see the updated wieght will now be dpendent on an anditional term which has a value propoortional to the sixe of the current weight The regularization techinques are incorporated in trianing by ussing a chosen hyperparamter /beta which will be incoroporated into the fomrulas above. The corresponding regularization fomula will then be computed on the ranodmly initialised weights for each chosen layer in the model. In order to update the gradient, backpropogation will be used in the same way as it is used to caluclate te gradient of the cost function with respect to the weights. The only differnce is that the corrsponing penalty gradient will be added on during the packrpopogation phase of training. In l2 regularization, a bias penalty is often not used due to large bieas values being potentially useful for the model as it allows for additional flexibility (Nielson, 2015). Furthermore, having large biases does not make the neurons inputs sensitice in the same way weights do.] .

[The weigth pentaltes help to address overfitting as it reduces the flexibility and sensitivity of the model by making the weights smaller. This in turn means The model is less likelly to fit to noise within the training data. Large values will be multiplied by small weight vectors so the output reuslt is less likely to deviate far from the target label. If no regularization was used, ther weights are unbounded and are free to diverge and become very large which leads to The free parameters being prone to fitting to noise. Rgularization almost acts as a form of smoothing for the trained parameters and prevents them from becoming too large which would amke the model sensitive to small changes in the inpuit. Regualrization helps to keep the model simpler and gices it a chance to learn patterns commonly present within the trianing data, hence leads to better generalization.

The main differnce between L1 and L2 are higihlighted by thier gradient terms. In L1 regualrization it can be seen that weights will always be updated at a constant rate with the only vairation being the direction in which the gradient is updated. With L2 regularization, the weights will be updated at changing rates due to the updated gradient being reliant on thr size of the current weights. As mentioned in (Nielson, 2015), the main differneces bettwen both types of regularizations are the weights that are targetted. For a model with large weights, L11 will shrink the weights slower than L2 due to L2's shrink factor being larger than L1's. For a model containing smaller weights, L2 will shirnk a lot

slower than L1. With L1, small weights are shrunk to 0, whilst a small number of high importance weight neurons are retained.] .

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 inclusion 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 start by lowering the learning rate to 10^{-4}, as the previous runs were overfitting in only a handful of epochs. Results for the new baseline (c.f. Table 3) confirm that lowering 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 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, **[Question 16 - Explain the motivation behind Maxout Networks as presented in (Goodfellow et al., 2013)]** . 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 - State whether Dropout is compatible (can be used together) with Maxout and explain why] .

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] .

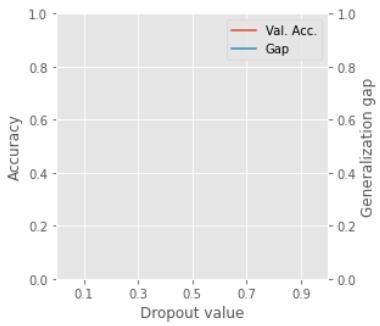
References

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.

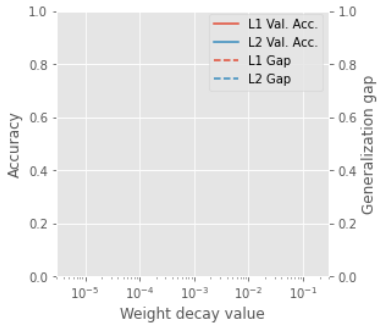
Goodfellow, Ian, Bengio, Yoshua, and Courville, Aaron.

Model	Hyperparameter value(s)	Validation accuracy	Generalization gap
Baseline	-	0.836	0.290
Dropout	0.7	0.817	0.030
	0.9	0.856	0.090
	0.95	0.861	0.140
L1 penalty	1e-4	0.847	0.08
	1e-3	0.751	0.01
	1e-1	0.0246	0.00
L2 penalty	1e-4	0.845	0.24
	1e-3	0.849	0.09
	1e-1	0.0198	0.00
Combined	0.95, L2: 1e-4		
	0.90, L1: 1e-4		
	0.95, L1: 1e-4		
	0.95, L2: 1e-3		
	0.90, L2: 1e-3		
	0.90, L2: 1e-4		

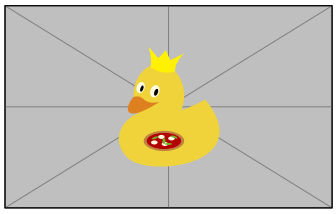
Table 3. Results of all hyperparameter search experiments. *italics* indicate the best results per series and **bold** indicate the best overall



(a) Metrics by inclusion rate



(b) Metrics by weight penalty



(c) Extra experiments

Figure 4. Hyperparameter search for every method and combinations

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.

Nielson, Michael A. *Neural Networks snd Deep Learning*. Determination Press, 2015. http://neuralnetworksanddeeplearning.com/.

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.