International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

Studying and Analysing the Effect of Weight Norm Penalties and Dropout as Regularizers for Small Convolutional Neural Networks

Ujjwal Kumar
Student of Computer Science Engineering
SRM Institute of Science & Technology
Chennai, India

Anamitra Bhar
Student of Computer Science Engineering
SRM Institute of Science & Technology
Chennai, India

***Abstract* – A long-standing problem for both machine learning neophytes and researchers has been to create a learning model that performs well not just on seen data points (training data) but also on unseen data points (test data). It is usually the case that a deep model learns and co-adapts representations and features from the training data so well, that it fails to perform effectively on the test data at all. This is known as the problem of overfitting. A lot of research has been devoted to come up with solutions to the problem of overfitting. To address this complication, apprehending the concept of generalization becomes pivotal. Generalization is the ability for a model to perform well on unseen inputs and collectively, all the approaches that work upon decreasing the generalization error of a learning algorithm are called regularization techniques. One way to reduce variance and generalization error in a model is to introduce a penalty term (weight decay) in the cost function that restricts the model's parameters from increasing. Another recently accepted practice is to use dropout, wherein hidden units and visible units are randomly dropped to obtain a much simpler model that prevents it from overfitting. In this paper, an extensive analysis has been done between both these regularizers on the MNIST dataset for relatively small convolutional networks. The findings of this paper assert that with appropriate hyperparameter settings, dropout performed a better job in bringing down the training error and making the gap between training and test error (generalization gap) minimum.**

Keywords – Dropout; Weight Decay; Regularization; Neural Networks; Convolutional Neural Networks

I. INTRODUCTION

The onset of learning through labeled data has revolutionized several pattern recognition contests, especially in applications of computer vision. Modern Convolutional Neural Networks (CNNs) [1] trained using the backpropagation algorithm [2] are predominantly suited for this task and has achieved state-of-the-art results in several image datasets. These artificial neural networks give unrivalled results on data that can be structured to have a grid-like topology. This is the reason why convolutional networks are chosen for computer vision applications as the feature images can be reduced to form either a 2D or 3D grid of pixels. The success of these modern gradient-based networks is associated with the fact that this type of network architecture makes use of sparse interactions wherein, each convolution operation is carried by a kernel with a size smaller than the input data which enables the network to store fewer parameters and hence reduces memory requirements and enhances efficiency.

Over the years, the task for machines to learn and generalize has become more complex as the data associated with them became more versatile and big; to overcome this problem, the architecture of models underwent changes to become deeper in order to account for the complex task which made them susceptible to variance and overfitting. The success of designing any learning algorithm is based on making the training error smaller and reducing the generalization gap [3]. A model that fails to provide a low training error suffers from underfitting, and a model that results in a large generalization gap suffers from overfitting. The model reaches its optimum performance when the complexity of the architecture is appropriate for the task it has been assigned (Fig. 1).

Factoring the dual challenge of bringing down the training and test error for a model, regularization comes into play, which facilitates in achieving a low generalization error on the data. Creating any machine learning algorithm involves the inclusion of a regularization algorithm in it, as it is the most cardinal aspect after optimization. In our study, we analyse two standard regularization methods used in deep learning. The L2 regularization or weight decay (often referred to as ridge regularization or Tikhonov's regularization) [4] that penalizes the parameters of the neural network which makes the weights have smaller squared L2 norm. Another regularization method proposed by Srivastava et al. [5], [6] that employs the bootstrap aggregating (bagging) [7] algorithm is called dropout, in which training is done on several bagged ensemble of networks, where all the networks share the parameters. This paper compares the effect of both of these regularizers on convolutional models by varying the hyperparameters associated with the respective regularization methods. The comparison is done by plotting the learning curves obtained after incorporating regularization methods on the network. The conclusions of this paper are based on favoring the regularizer that accomplishes the task of reducing the training error and making the generalization gap smaller.

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

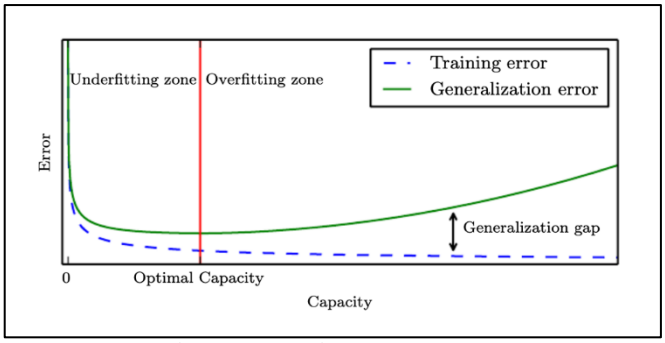

Fig. 1. Plot of the loss trend of a model against its capacity [3].

II. WEIGHT DECAY AND DROPOUT

Maintaining the tradeoff between optimization and generalization is a focal task in deep learning. Generalization focuses on improvising the model's performance of data that it has never come across before, whereas optimization works on increasing the model's performance metrics on the training set. Improving generalization of a learning algorithm on the test set can be done by restricting the parameters of a network from taking large values as the weight distribution becomes more regular which results in a simpler model that is less prone to overfitting. This is done by adding a cost associated with having large weights to the overall loss function of the network and is the premise behind L2 regularization or weight decay. Introduction of a parameter norm penalty to the overall loss function assists in limiting the overall capacity of the model and is done by:

$$J_{reg}(W) = J(W) + \frac{\Lambda}{2} W^T W \quad (1)$$

In the above equation, W denotes the parameters/weights of the network and Λ denotes the regularization parameter. The regularization parameter can be thought of as a hyperparameter that needs to be tuned ahead of time. The value of Λ equals to zero means that there is no regularization on the model, similarly choosing a lofty value of Λ results in making the weights even smaller. The intuition of adding parameter penalties to the loss function of the network can be gained by the conduct of weights under gradient descent. During backpropagation, the gradients of the parameters are calculated by partially differentiating the loss function with respect to the weights of the network:

$$\nabla_W J_{reg}(W) = \nabla_W J(W) + \Lambda W \quad (2)$$

Now to adjust the model's weight after each gradient step, the weight update is done by subtracting the initial weight with the gradient calculated above:

$$W \leftarrow W - \Omega[\nabla_W J_{reg}(W)] \quad (3)$$

The term Ω denotes the learning rate of the network which determines the size of the step taken by the model to move closer to the optimum value.

$$W \leftarrow W - \Omega[\nabla_W J(W) + \Lambda W] \quad (4)$$

After rearranging the terms of the weight update, we get:

$$W \leftarrow (1 - \Omega\Lambda)W - \Omega\nabla_W J(W) \quad (5)$$

It is evident from (5) that the weight decreased by a fixed value $(1 - \Omega\Lambda)$ i.e. the weight is decayed after each step of gradient descent. The addition of norm penalties forces the weights closer to the origin and hence induces a regularizing effect to the model.

Reduction of variance in a model can also be achieved by employing ensemble methods in which instead of a single model, multiple models are trained and then the outputs of all the models are sampled over to predict the results on unseen data points. Dropout uses a similar notion. Dropout training involves dropping random units (hidden and visible units only) from the network architecture with a probability of p (Fig. 2). This random dropping of units creates an ensemble of several thinned networks which the model samples over to predict results. Combining predictions of several models is a computationally extensive job which requires training each model separately and finding the optimum hyperparameter for each model, however dropout solves this issue by sharing the network parameters among the models which in turn reduces the complexity. If a neural network is assumed to have l units, then it can approximately have 2^l number of thinned networks to sample over.

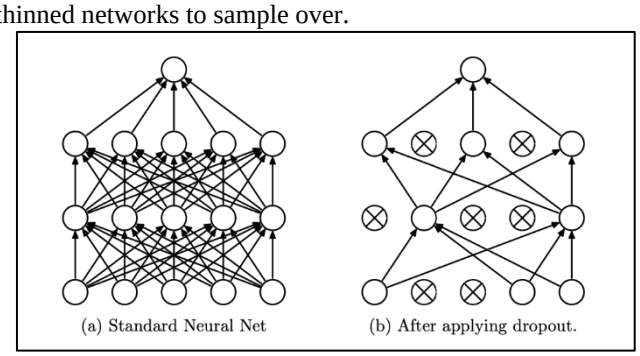

Fig. 2. Left: A dense network with all units preserved. Right: A thinned network produced from the parent network after randomly dropping several units [5].

III. EXPERIMENTATION AND EVALUATION

To scrutinize the effects of the aforementioned regularizers on a convolutional model, the MNIST [8] dataset was used for classifying handwritten digits into 10 classes (0-9). The dataset contains 60,000 training and 10,000 test images, all of which represents a 28 × 28 grayscale pixel image of a digit from any of the 10 classes (Fig. 3).

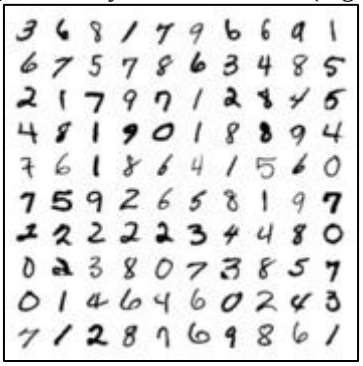

Fig. 3. Size normalized images of handwritten digits from MNIST [8] dataset. showing all 10 classes (0-9).

The convolutional model architecture used for this study consists of 3 convolutional layers each of which is followed by a max pooling [9] layer. Implementing the max pool operation after each convolution summarizes the maximum output within the rectangular grid of handwritten pixel images. Pooling operation improves the computational efficiency of the model as it downsizes the image which makes it easier for the next layer to learn visual

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

representations of the features. The output from the last pooling layer is then flattened and fed into a dense layer. All the layers employ the ReLU activation except the output layer which uses Softmax activation to predict the 10 classes of digits. Refer Fig. 4 for the exact dimensions of the layers. The 28 × 28 × 1 image is processed by a series of convolutional and max-pooling layer. The kernel size of each convolution was set to be 3 × 3 and the corresponding pool size to be 2 × 2. From the last pooling layer, the image is flattened and is finally processed by a series of dense layers. The model was compiled using Adam [10], a gradient-based optimizer. The learning rate parameter for the network was set to be the same when the model used either of the regularizers - L2 or dropout – with a value of $\Omega = 1 \times 10^{-3}$ for impartial analysis. The principal amount of effort made in this study was in varying the regularization parameter Λ (associated with L2) and the probability p of dropping neurons (associated with dropout) to identify the performance of the network on either of the regularizers. For simplicity, the architecture of the convolutional network was sustained throughout our experiment.

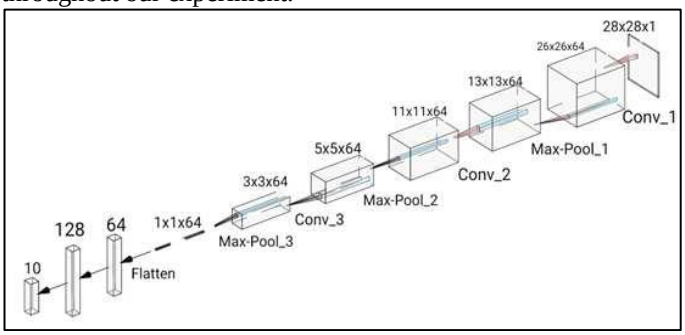

Fig. 4. The convolutional model architecture used in this experiment is displayed here.

In the first run, the convolutional model was trained using only dropout as the regularizer where the probability p of randomly dropping units varied from p = 0.05, 0.1, 0.2 to 0.3, the relative observations were recorded in Table I. Fig. 5.1 - 5.4 display corresponding learning curves showing the training and validation loss obtained from the model trained on dropout. The results of the model trained using L2 regularization with $\Lambda = 1 \times 10^{-3}, 1 \times 10^{-4}, 1 \times 10^{-5}, 1 \times 10^{-6}$ are shown in Table.2 and the corresponding learning curves in Fig. 6.1 - 6.4. The values of the hyperparameters (p and Λ) were chosen because they gave a noticeable close performance in terms of the accuracy and loss value, however, the purpose of this experiment is to identify amongst these combinations of hyperparameters, which regularizer outperforms the other. The loss plots obtained from the model trained with L2 regularization (Fig. 6.1 - 6.4) showed substantial amount of irregularities. The learning curves from the model trained using dropout portrayed smooth loss plots with low generalization gap between them as compared to L2 regularization. Fig. 5.3 and 5.4 demonstrate that the model trained using dropout with p = 0.2 and 0.3 respectively produced test loss lower than training loss The reason behind this behavior is based on the fact that the model architecture used in this study is small with only 3 convolutional layers and dropping 20% or 30% of the features makes the training set relatively easier to train. However, at test time, dropout is not applied and all the features are retained which makes the model robust and susceptible to higher loss.

It is evident from Table I and Table II that the model trained with dropout produced the lowest difference between training loss and test loss with average difference across the four values of p being 0.021 which is 0.017 less that the average difference across the four values of Λ from the model performance obtained using weight decay.

TABLE I. MODEL PERFORMANCE OBTAINED USING DROPOUT

Dropout Rate (p)	Training Accuracy TrA (%)	Test Accuracy TeA (%)	|TrA – TeA|	Training Loss TrL	Test Loss TeL	|TrL – TeL|
0.05	99.54	98.92	0.62	0.013	0.051	0.038
0.1	99.53	99.15	0.18	0.021	0.038	0.017
0.2	98.80	99.02	0.22	0.038	0.037	0.001
0.3	98.10	99.00	0.90	0.063	0.034	0.029
			Average = 0.480			Average = 0.021

TABLE II. MODEL PERFORMANCE OBTAINED USING WEIGHT DECAY

Regularization Parameter (Λ)	Training Accuracy TrA (%)	Test Accuracy TeA (%)	|TrA – TeA|	Training Loss TrL	Test Loss TeL	|TrL – TeL|
1×10^{-3}	98.64	98.45	0.19	0.102	0.108	0.006
1×10^{-4}	99.48	98.47	1.01	0.045	0.086	0.041
1×10^{-5}	99.71	98.87	0.84	0.021	0.064	0.043
1×10^{-6}	99.73	98.71	1.02	0.010	0.072	0.062
			Average = 0.765			Average = 0.038

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

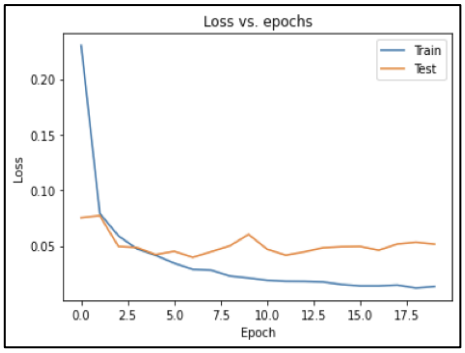

Fig. 5.1. Dropout, p = 0.05 (5%)

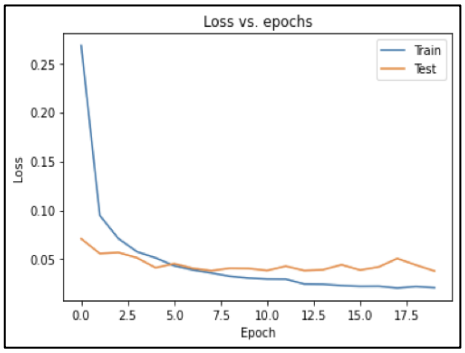

Fig. 5.2. Dropout, p = 0.1 (10%)

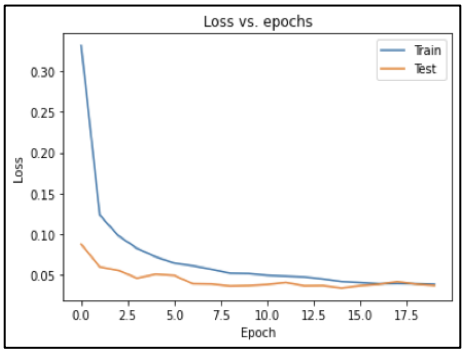

Fig. 5.3. Dropout, p = 0.2 (20%)

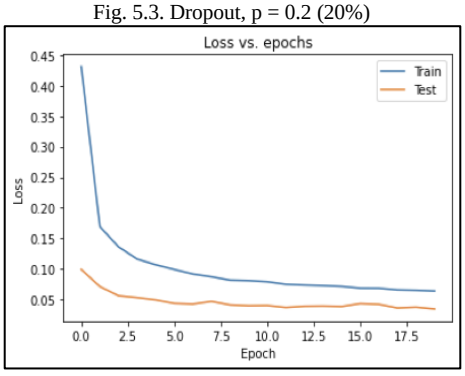

Fig. 5.4. Dropout, p = 0.3 (30%)

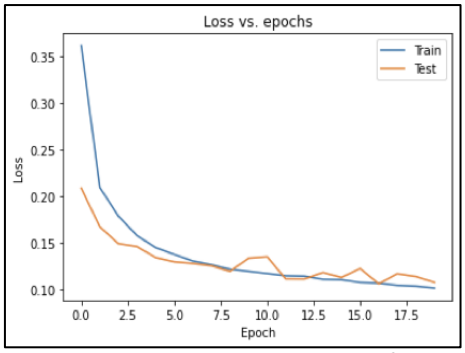

Fig. 6.1. Weight Decay, $\Lambda = 1 \times 10^{-3}$

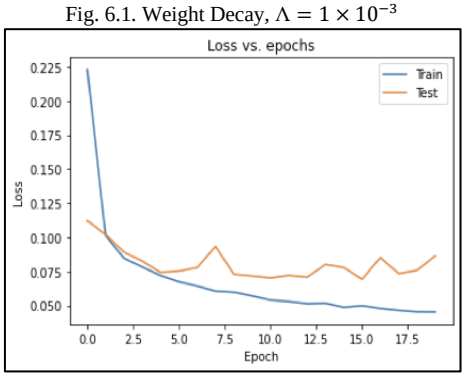

Fig. 6.2. Weight Decay, $\Lambda = 1 \times 10^{-4}$

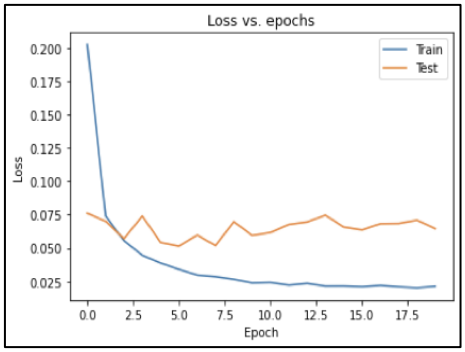

Fig. 6.3. Weight Decay, $\Lambda = 1 \times 10^{-5}$

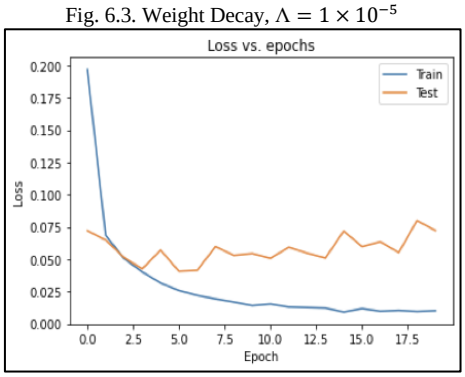

Fig. 6.4. Weight Decay, $\Lambda = 1 \times 10^{-6}$

IV. CONCLUSION

The practice of regularizing neural networks has come a long way with the numerous advances to enhance the generalization of the networks. For a learning machine to give its most advantageous performance, a perfect balance needs to exist between the complexity of the model and the amount of information about the training set. Weight decay is a type of regularization where the model complexity is limited by preventing the weights of the network from growing too large. Using dropout is also a well-received

International Journal of Engineering Research & Technology (IJERT)
ISSN: 2278-0181
Vol. 10 Issue 01, January-2021

strategy to limit the model complexity as it involves sampling over several thinned networks which all share parameters amongst themselves making it less expensive than other ensemble methods. The experiments of this study demonstrate that employing dropout enhances the generalization ability of the model mainly by lowering the generalization gap ad minimizing the training error of the model when compared with regularizers that penalizes weights from growing (weight decay). The scope of dropout is not limited to just convolutional models, but can also address overfitting issues in Multiplayer Perceptrons or even Restricted Boltzmann Machines.

V. REFERENCES

[1] Y. LeCun, "Generalization and Network Design Strategies", Technical Report, CRG-TR-89-4, Department of Computer Science, University of Toronto, 1989.
[2] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, "Learning Internal Representations by Error Propagation", *Parallel Distributed Processing: Explorations in the Microstructure of Cognition: Foundations.* MIT Press, pp. 318-362, 1986.
[3] I. Goodfellow, Y. Bengio, and A. Courville, "*Deep Learnning"*. MIT Press, 2016.
[4] A. N. Tikhonov, "On the Stability of Inverse Problems", *Doklady Akademii Nauk SSSR,* 39(5): pp. 195-198, 1943.
[5] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A Simple Way to Prevent Neural Networks from Overfitting", *Journal of Machine Learning Research,* 15(1): pp. 1929-1958, 2014.
[6] N. Srivastava, "Improving Neural Networks with Dropout", Master's thesis, Graduate Department of Computer Science, University of Toronto, 2013.
[7] L. Breiman, "Bagging Predictors", *Machine Learning*, 24(2): pp. 123-140, 1994.
[8] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition", *Proceedings of the IEEE,* Vol. 86, No. 11, pp. 2278-2324, 1998.
[9] Y. Zhou and R. Chellappa, "Computation of optical flow using neural network", in *IEEE 1988 International Conference on Neural Networks,* Vol. 2, pp. 71-78, 1988.
[10] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines", in *27th International Conference on Machine Learning*, pp. 807-814, 2010.
[11] D. P. Kingma and J. Ba, "Adam: a method for stochastic optimization", in *3rd International Conference on Learning Representations*, 2015.