Examining the Impact of Dropout on Overfitting in Deep Learning Architectures

Mohammed Ali Maghmoum
Department of Artificial Intelligence
Bahcesehir University
mohamedali.maghmoum@bahcesehir.edu.tr

Hashem Ali Alshami
Department of Artificial Intelligence
Bahcesehir University
ali.alshami1@bahcesehir.edu.tr

36th Conference on Neural Information Processing Systems (NeurIPS 2022).

Abstract

Despite having powerful computers, deep learning networks frequently deal with overfitting. Dropout is a technique created to address this problem by occasionally deactivating some network units during training, preventing them from becoming overly dependent on one another. When tested with a single, unthinned network, this procedure generates a number of simpler "thinned" networks, whose predictions are averaged. This method significantly improves on other regularization algorithms and successfully reduces overfitting when applied to different model architectures, making it useful for supervised learning applications including speech recognition, document categorization, and computational biology.

Keywords: regularization, dropout, deep learning, neural networks, model combination, dropout rate.

1 Introduction

Deep neural networks learn complex relationships through their nonlinear hidden layers, but overfitting can be a challenge, particularly with limited training data. Strategies like early training cessation, weight penalties, and L1/L2 regularization have been used, but but some of these become computationally expensive. Dropout offers a solution by training multiple thinned networks with shared parameters, randomly removing units during training to prevent overfitting. During testing, a single unthinned network with reduced weights simulates the averaged outputs of the thinned networks. Dropout has shown significant improvements in generalization across various classification problems and can efficiently combine outputs from separately trained models.

At each update during training, dropout randomly sets a portion of the input units to 0, the probability of dropping out a neuron or unit during training is referred to as the dropout rate, dropout's only tunable hyperparameter, the value of dropout rate can be from 0 to 1, a dropout rate of 0.2 or 0.5 are often thought to be optimal.

2 Related Works

Dropout is a regularization technique for neural networks that introduces noise to hidden units. It is an extension of the concept used in Denoising Autoencoders (DAEs) to add noise to hidden layers, which helps in model averaging. Previous studies have examined deterministic regularizers for exponential-family noise distributions, including dropout. However, they mainly focus on input noise and models without hidden layers. In this approach, we incorporate stochastic loss minimization under a noise distribution, while other works consider fixed maximum unit dropout and do not explore hidden units.

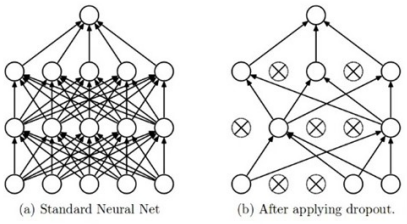

Figure 1: Before And after Dropout

3 Experimenting with Dropout Models

To test out the effectiveness of dropout models in preventing overfitting, a number of datapoints belonging to two different classes were defined and plotted on a plane, two models were used to apply classification in order to separate the two different classes, the models contain two hidden layers,

each having 128 units. Model 1 does not apply dropout to any of its layers. While model 2 applies dropout with rate 0.5 to each hidden layers.

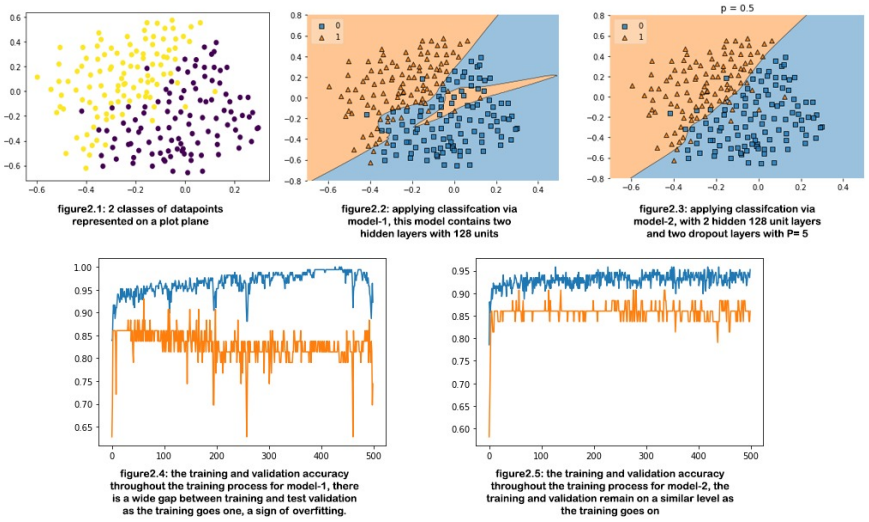

Figure 2: Classification task with and without dropout: model 1 overfitted the data, as the training continued, the validation accuracy dipped below the training accuracy a significant amount. Model 2 on the other hand was able to keep its training and validation accuracies consistent.

A variety of architectures built with tensorflow keras were created and used to conduct machine learning tasks on real-world data to examine the effectiveness of using dropout to prevent overfitting and improve the performance of machine learning models.

Data Sets	Domain	Dimensionality	Training Set	Test Set
MINIST	Vision	784 (28 × 28 grayscale)	60K	10K
CIFAR-10	Vision	3072 (32 × 32 color)	60K	10K
CIFAR-100	Vision	3072 (32 × 32 color)	60K	10K

Table 1: Overview of datasets used in experimenting

3.1 MNIST Dataset Experiment

The MNIST data set comprises of handwritten digit pictures of 28x28 pixels. The aim is to categorize the photos into ten-digit classes. In this experiment, 7 different architectures were used to perform a classification task, dropout was then applied to these models, the evaluations of all models were then compared to observe the effects dropout had on each individual model, keeping all parameters unchanged.

The results shown in Figure 3 showcase that dropout generally improves the performance of models by decreasing the effects of overfitting, however, this effect was not observed on all tested models, models 3,6 and 7 showed a decline in performance after dropout was applied. Model 3 has the same architecture as model 2 but uses relu activation function, multiple results show that using relu with dropout generally causes underperformance. models 6 and 7 are both wide models that contain an increased amount of units, the results show that dropout heavily decreases their accuracy if the dropout rates are kept the same.

3.2 CIFAR-10 and CIFAR-100

The CIFAR-10 and CIFAR-100 data sets each contain 32×32 color images selected from ten and one hundred categories, respectively. To observe the effects of dropout, we used 2 different models and we tested them without and with dropout and calculated the error rate.

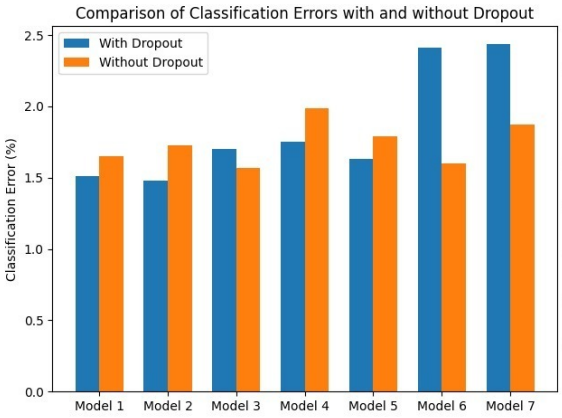

figure 3.1: the classification error of all the models before and after applying dropout.

model name	architecture	act-function
model1	2 hidden layers 800 units	Sigmod
model2	3 hidden layers 1024 units	Sigmod
model3	3 hidden layers 1024 units	re-LU
model4	3 hidden layers 1024 units with max-norm	Sigmod
model5	3 hidden layers 2048 units with max-norm	Sigmod
model6	2 hidden layers 4096 units with max-norm	Sigmod
model7	2 hidden layers 8192 units with max-norm	Sigmod

figure 3.2: the architecture and activation function of each model used.

Figure 3: comparing the results of dropout and standard models trained on MINST

Method	Cifar10	Cifar100
Conv Net + max pooling (hand tuned)	30.76	66.44
Conv Net + max pooling + dropout fully connected layers	27.87	62.04

Table 2: Error rate percentage on CIFAR10/100: As seen in the table, using the same model but adding dropout in fully connected layers decreases error rate on both datasets.

4 Dropout Neural Networks vs Bayesian Neural Networks

Dropout and Bayesian neural networks are both methods of model averaging, but they differ in their approach. Dropout performs equally-weighted averaging of models with shared weights, while Bayesian neural networks consider the prior and model-data fit to assign weights to each model. Bayesian neural nets are effective in domains with limited data, such as medical diagnosis and genetics, but they are slow to train and scale to large networks. In contrast, dropout nets are faster and easier to use at test time. An experiment was conducted to compare the performance of Bayesian neural nets and dropout nets on a MNIST dataset where Bayesian nets excel, aiming to assess the extent to which dropout falls short in comparison.

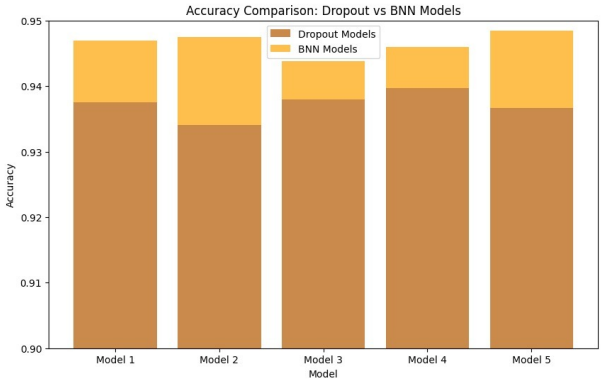

Figure 4: Dropout Neural Networks vs Bayesian Neural Network on MNIST

5 Dropout vs Standard Regularizers

Dropout can be seen as a regularization technique, the experiment conducted here aims to observe how dropout fares in comparison with other standard regularizes like max-norm and L2, the regularizers were applied onto a model preforming a classification task on the MINST dataset, the architecture for this model is 1024-1024-2048, dropout rate is set at 0.2 for all layer

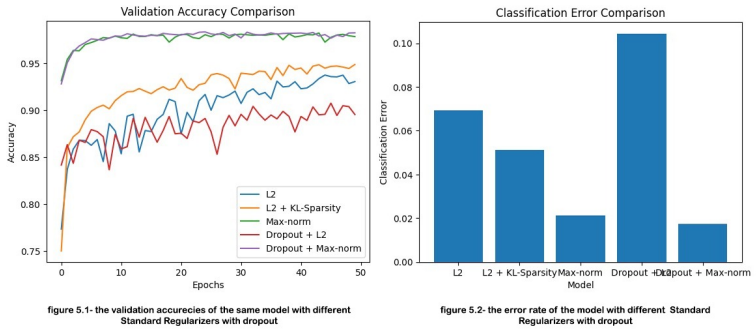

figure 5.1- the validation accurecies of the same model with different Standard Regularizers with dropout

figure 5.2- the error rate of the model with different Standard Regularizers with dropout

Figure 5: Shows validation accuracy and classification error of the same model (1024-1024-2048) max-norm+dropout achieves the best results, followed by max-norm. L2 regularizer performs noticeably lower, with L2+dropout performing the worst

6 Conclusion and Discussion

Dropout is considered an innovative regularization technique, it is an efficient and simple method to prevent overfitting in deep neural networks, a recurring challenge in machine learning. Its distinctive approach of randomly deactivating certain units during training prevents complex co-adaptations and curbs model over-reliance on specific features. Experiments using various architectures and data sets, such as MNIST and CIFAR, demonstrate the technique's efficacy, although the impact can vary across different models and configurations. Notably, dropout outperforms traditional regularizers like max-norm and L2 in various scenarios, further solidifying its utility in improving model generalization. However, certain cases witnessed a drop in performance, which brings the question, wheather or not dropout is the optimal method to reduce overfitting acorss machine learning models. Comparatively, dropout can be seen as another method of combining multiple pre-trained models, although it usually underperforms in comparison to Bayesian Neural Networks, it provides an efficient and computationally cost-effective method of averaging many trained models. In the end, it is essential to understand and consider the strengths and limitations of dropout for specific applications and datasets in the continually evolving field of machine learning.

7 Github Repository

`https://github.com/Magmuma/AIN3002TermProject`.

References

[1] Y. LeCun, Y. Bengio, and G. Hinton, *Deep learning*, Nature, vol. 521, no. 7553, pp. 436–444, 2015.

[2] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, *Dropout: a simple way to prevent neural networks from overfitting*, The Journal of Machine Learning Research, vol. 15, no. 1, pp. 1929–1958, 2014.

[3] A. Krizhevsky and G. Hinton, *Learning multiple layers of features from tiny images*, 2009.

We utilized the Keras API, a user-friendly neural network library written in Python, in our work. For more information on Keras, refer to the online documentation available at `https://keras.io/`.