COMP 551 - A4

Juan David Guerra, Mauricio Rivera, Ethan Kreuzer

April 2023

Abstract

Implementing dropout to counter a deep neural network's intrinsic tendency to overfit as its computational expressiveness increases is a widespread method in many machine learning applications. In short, randomly dropping units from the neural network during training prevents the units to co-adapt, hence preventing the model from overfitting. In this reproducibility project, we attempted to reproduce the findings of Srivastava et. al [1]. Therefore, we tested the effect of Dropout on the well-known MNIST digit-classification dataset. Indeed, we found dropout improved the neural networks' performance by reducing overfitting, regardless of the neural network's architecture, activation functions, regularization techniques, and dataset size.

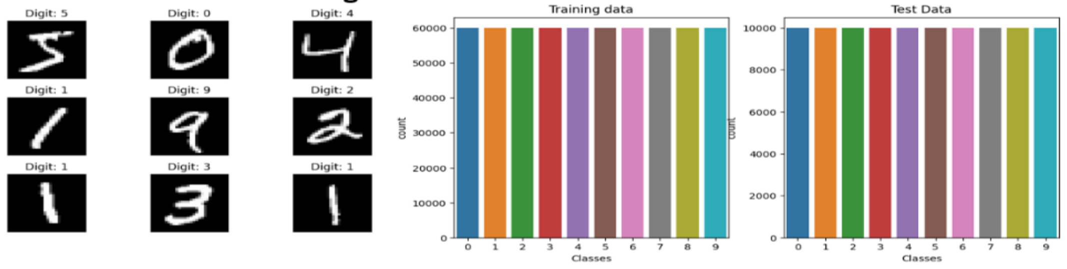
Introduction

Neural networks have revolutionized artificial intelligence through their ability to learn complex problems. Although these models are very powerful, they are highly susceptible to overfitting due to their ability to adapt too quickly to the training sets. Dropout has been proposed and shown experimentally to be a way to reduce the effect of overfitting [1]. This is done by randomizing through a Gaussian distribution whether or not a node and its weights are present during a training epoch and when completing that training epoch, dividing the weights of the nodes present during that epoch by the probability that a node will not appear. This causes the model's weights to "rely" on other weights to compensate for its mistakes since they are likely to not train together. This paper aims to elucidate the effects of hyper parameter search over dropout rate and other important hyper parameters on the accuracy of the models on the MNIST data set. We observe that due to training restrictions, we are not able to obtain the same performance as obtained by Srivastava et al. however we do see a trend leading to what they observed, suggesting that given more training instances we would obtain the same results. These results are that dropout would greatly increase the generalization of the models given the proper dropout rates.

Datasets

The MNIST dataset consists of 60 000 training and 10 000 test images of handwritten digits between 0 and 9 (see Fig. 1). Each instance contains an 28x28 pixel image and an associated digit label. Each pixel value represents its gray-scale value (between 0 and 255). The digit's class distribution was uniformly distributed (see Fig. 1). Considering most of the image's pixel values were 0 (because they represent the 'empty' space surrounding a digit), each instance was normalized, as it has shown to improve the model's accuracy [2] and training time.

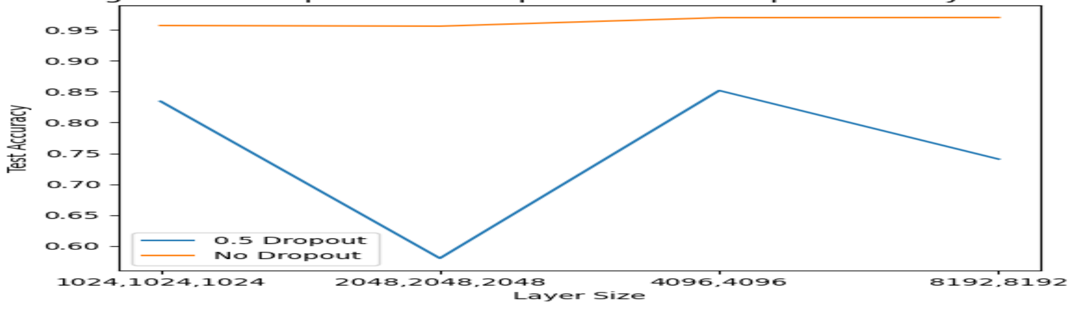
Figure 1. - MNIST dataset



Results

The results of the main model comparison based on the MLP models provided by the reference paper [CITE THE THING HERE] were replicated using Sklearn libraries with dropout layers, max-norm layers, as well as adjusted momentum and learning rate from the standard MLP rates as was stated in Srivastava et al. These adjusted rates are as follows: Learning rate was set to 0.1, momentum to 0.95, dropout rates to 0.5 for each hidden layer, and max norm to 3 at each layer. To compare the results with standard non-dropout MLPs, we created a graph that plots all models' accuracy after 20 epochs of training with dropout and without against the accuracy obtained with the test set (Fig. 2).

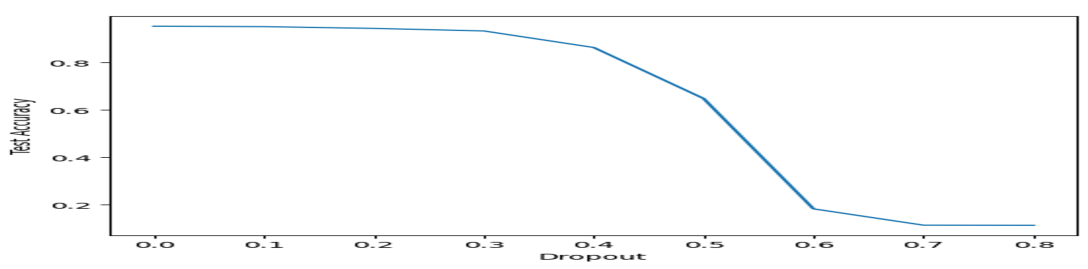
Figure 2 - Comparison of dropout vs. non dropout and layer sizes



As we can see from the figure, the dropout rate is negatively affecting the model's performance across all layers. The non-dropout models stagnate around a 96% accuracy rate with the dropout having a more varied, less accurate on average performance. This contrasts with what was observed in the original Srivastava et al. where the dropout models significantly outperformed the non-dropout models. This could be a result of training constraints due to time as training these models is a very costly process, and to obtain the performance from the original paper, significantly more training epochs are required.

To investigate which dropout rate would work the best for our training abilities, we ran a search over different dropout rates for the models and trained them over 20 epochs to see their accuracy on the test set (Fig 3).

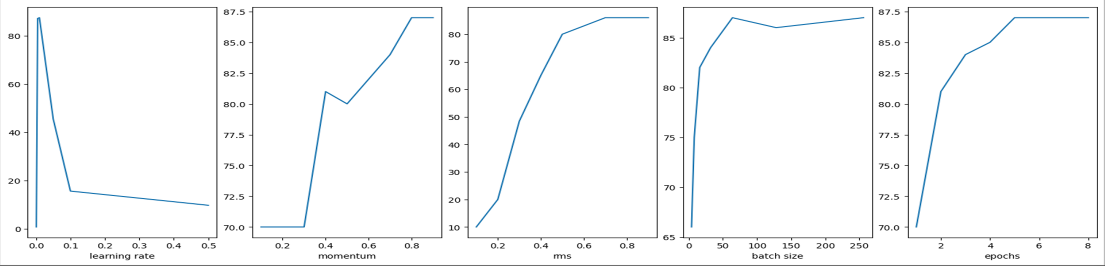
Fig 3 — Effect of Dropout rate on Test Set Accuracy



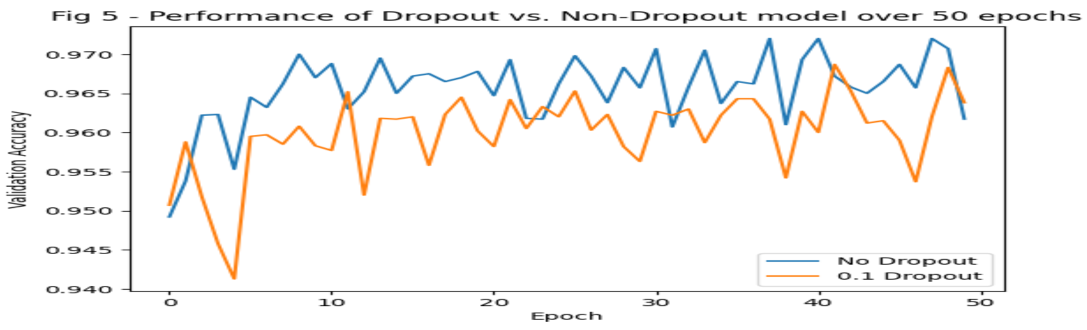
As we can see, the more dropout we add to the model, the less accurate the model becomes with a drastic reduction in test accuracy performance around 0.5 dropout. This was interesting to us, as Srivastava et al. state that the optimal dropout for hidden layers for a model testing on MNIST is 0.8, the worst-performing dropout rate we found. This furthers our belief that our limitations with training epochs could be the driving force for these skewed results.

To further understand what could be affecting these models, we conducted a hyperparameter search over the learning rate, momentum, RMS rate, batch size, and the number of epochs (Fig 4).

Fig 4 — Hyper Parameter Search



These models were trained with fixed parameters outlined by the original paper except for the hyperparameter that was being investigated. Therefore, the reduction in the model's performance must be with the construction of the models with dropout and/or the number of training epochs. To check if it was the number of epochs affecting the performance of the models, we trained a model with the highest experimental dropout rate (0.1) and hyperparameters. We tested over 50 epochs and compared it to a model with no dropout and the same other hyperparameters (Fig 5).



The figure shows that the accuracy over the validation set across the epochs is lower for the dropout model, albeit slightly less, than the non-dropout model. However, this experiment showed an interesting result, as we increased the number of training epochs, the performance of the dropout model over the test set was 0.9638 whereas the non-dropout model only achieved an accuracy of 0.9519. This shows that our initial hypothesis was correct that the dropout model requires more time to train to achieve the same performance as non-dropout models and eventually surpasses them since dropout reduces over-fitting.

Discussion

As was shown in the results, our experimental results aligned in general with that of the original paper except for the main feature point, dropout rate. After investigating the performance of various models with and without dropout and different layers, we hypothesized that the reduction in performance on the test set for the dropout models was due to training time restrictions. After comparing the performance of a model with 0.1 dropout and a model without dropout over a longer training period, we saw that although the dropout model did not perform as well on the validation set, it outperformed the non-dropout model on the test set. This shows the exact hypothesis proposed by Srivastava et al. suggesting that training a model with dropout will reduce the overfitting and allows the model to generalize better to unseen data. We hypothesize that the reduced validation accuracy from the dropout model can be attributed to the premise of the dropout architecture. This is because the dropout model is essentially training separate subsets of the full model to reduce dependence between weights since some may or may not be present together during different training epochs. Due to this, the model may not attain the same performance over the activation set but generalizes better to the test set since it is preventing overfitting. We suspect that given more computing resources, we would be able to attain significantly higher performance accuracy with higher dropout rates when training over a significantly longer period of time.

Statement of Contributions

Mauricio Rivera - Data preprocessing, MLP model building and report
Juan David Guerra - MLP model building and report
Ethan Kreuzer - Hyper Parameter Search and Report

References

[1] Srivastava, Nitish, et al. “Dropout: A Simple Way to Prevent Neural Networks from Overfitting.” Journal of Machine Learning Research, June 2014, jmlr.org/papers/volume15/srivastava14a/srivastava14a.pdf.

[2] Singh, D., & Singh, B. (2020). Investigating the impact of data normalization on classification performance. Appl. Soft Comput., 97, 105524.