Adaptive Dropout: Enhancing Regularization in Neural Networks

Lior Kreimer Shahar Lavian Paz Adziashvili Avi Atanelov

Abstract

In deep learning, dropout has become a crucial technique for mitigating overfitting, as introduced by Srivastava et al. in their paper "Dropout: A Simple Way to Prevent Neural Networks from Overfitting." Dropout employs a stochastic method by randomly deactivating neurons during training, which enhances the generalization of neural networks. However, the fixed dropout rate can limit its ability to meet dynamic learning needs throughout training. This report investigates an adaptive dropout mechanism, inspired by recent advancements, which adjusts the dropout rate progressively during training.

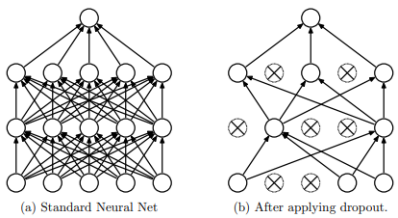
We hypothesize that a dynamic dropout rate can improve model performance by better balancing underfitting and overfitting. Initially, a higher dropout rate offers strong regularization when the model is more vulnerable, gradually decreasing to allow fine-tuning. Our study examines the effects of adaptive dropout on Multi-Layer Perceptrons (MLPs) and Convolutional Neural Networks (CNNs) using benchmark datasets like MNIST and CIFAR-10. The report details the proposed methodology, experimental setup, and performance metrics. Through rigorous experimentation, our results show that adaptive dropout enhances model robustness and accuracy, suggesting it as a superior regularization method in deep learning.

Code is available at - https://github.com/pazadz10/Deep-Learning-Project.git

1. Introduction

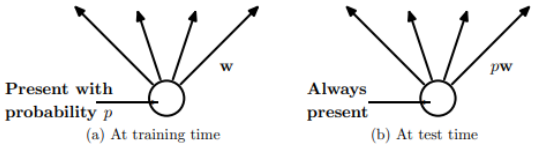
Deep learning has revolutionized numerous fields, including computer vision, natural language processing, and speech recognition. Despite its successes, one of the enduring challenges is over-fitting, where a model performs well on training data but poorly on unseen data. To address over-fitting, various regularization techniques have been proposed. One of the most effective techniques is dropout. Dropout is a simple yet powerful regularization method that involves randomly deactivating a fraction of the neurons during training. The term “dropout” refers to dropping out units (hidden and visible) in a neural network. By dropping a unit out, we mean temporarily removing it from the network, along with all its incoming and outgoing connections, as shown in Figure 1. Specifically, during each training iteration, a certain percentage of neurons is randomly selected and temporarily removed from the network, meaning their output is set to zero. This "dropout" is only applied during training, and during testing or inference, all neurons are used but with their outputs scaled by the dropout rate as shown in Figure 2. This technique prevents the co-adaptation of neurons, forcing the network to learn more robust and generalizable features (Srivastava et al., 2014).

Figure 1: Dropout Neural Net Model. **Left**: A standard neural net with 2 hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped (Srivastava et al., 2014).



(a) Standard Neural Net (b) After applying dropout.

Figure 2: Left: A unit at training time that is present with probability p and is connected to units in the next layer with weights w. **Right**: At test time, the unit is always present, and the weights are multiplied by p.(Srivastava et al., 2014).



The original paper (Srivastava et al., 2014). established dropout as a highly effective regularization technique. The authors demonstrated that dropout could significantly improve the performance of neural networks on several benchmark datasets, including MNIST, CIFAR-10, and ImageNet. They also provided theoretical insights into why dropout works, showing that it can be seen as a form of ensemble learning, where multiple sub-networks are trained simultaneously, and their predictions are averaged at test time.

However, the dropout rate in traditional dropout is static, which means it does not change throughout the training and it does not adapt to the changing dynamics of the training process. This static nature can be sub-optimal as the need for regularization changes throughout training. Early in training, when the model is rapidly learning and may prone to over-fitting, a higher dropout rate might be beneficial. Conversely, as training progresses and the model begins to fine-tune its parameters, a lower dropout rate might be more appropriate.

This observation has led to the exploration of dynamic dropout techniques, where the dropout rate is adjusted during training. Previous works have explored different strategies for varying the dropout rate. For instance, Variational Dropout, where the dropout rate is learned as a variational parameter (Molchanov et al., 2017). Similarly, Ba and Frey introduced Adaptive Dropout, where the dropout probability is conditioned on the input (Ba & Frey, 2013). These methods have shown promising results, but they often involve complex modifications to the network architecture or the training process.

Our proposed Adaptive Dropout method builds on these ideas by dynamically adjusting the dropout rate based on the training epoch, providing a straightforward yet effective way to improve regularization. The idea is to start with a high dropout rate in the early stages of training to combat over-fitting and gradually decrease it as training progresses, allowing the network to fine-tune its learned features. This approach is inspired by the natural learning process in biological systems, where synaptic pruning and strengthening occur dynamically based on experience and learning (Peter R., 1979).Another variation of Adaptive Dropout involves starting with a low dropout rate and increasing it as training progresses. This approach recognizes that early in training, the model may be less prone to over-fitting, and lower regularization may suffice, allowing the model to learn more from the data. As training proceeds and the risk of over-fitting increases, a higher dropout rate can help in maintaining model generalization.

2. Proposal

The proposed Adaptive Dropout method aims to enhance the traditional dropout technique by introducing a dynamic component that adjusts the dropout rate throughout the training process. The core idea is to start with a high dropout rate in the initial phases of training, where the model is more susceptible, and gradually reduce it as training progresses, thereby allowing the network to fine-tune its learned features.

Mathematically, let p_i and p_f denote the initial and final dropout rates, respectively. The dropout rate $p(t)$ at epoch t for the decreasing dropout schedule is defined as:

$$p(t) = p_i - (p_f - p_i)\frac{t}{T}$$

where T is the total number of training epochs. This linear schedule ensures a smooth transition from high to low dropout rates.

Another variation of Adaptive Dropout is opposite approach that involves starting with a low dropout rate in the initial phases of training and increasing it as training progresses. The dropout rate $p(t)$ at epoch t for the increasing dropout schedule is defined as:

$$p(t) = p_i + (p_f - p_i)\frac{t}{T}$$

This schedule starts with a lower dropout rate p_i and gradually increases it towards p_f as training proceeds.

To implement Adaptive Dropout, we modify the dropout layer in the neural network to accept the current epoch as an input and compute the dropout rate according to the schedule defined above. This requires minimal changes to the existing network architecture and can be easily integrated into any deep learning framework.

We evaluate the effectiveness of both variants of Adaptive Dropout on the MNIST and CIFAR-10 datasets. we compare the results obtained using Adaptive Dropout with those achieved using traditional dropout, on part of the architectures from the original dropout article.

The network architectures used in our experiments include:
1.multiple fully connected layers (MLPs) with sigmoid activations.
2. multiple fully connected layers (MLPs) with ReLU activations and max-norm constraints.
3. convolutional neural networks (CNNs) with 3 convolutional layers, 3 fully connected layers and ReLU activations.

3. Methodology

In this section, we delve into the methodology employed for implementing adaptive dropout in neural networks, which is the core innovation of our project. The methodology is structured into four primary subsections: Datasets, Network Architectures, Code Implementation, and Evaluation Metrics. Each subsection comprehensively details the processes and decisions involved in our approach.

Datasets

For our experiments, we utilized two benchmark datasets: CIFAR-10 and MNIST. CIFAR-10- The CIFAR-10 dataset consisting of 60,000 32x32 color images in 10 classes, with 6,000 images per class.

MNIST- The MNIST dataset is a large database of handwritten digits that consists of 70,000 28x28 grayscale images of the digits 0-9, with 60,000 images in the training set and 10,000 images in the test set.

Network Architectures

We implemented three variations of neural network architectures (that we mention in the proposal) to evaluate the impact of adaptive dropout rates These architectures are taken from the original dropout paper. For all architectures we applied a SoftMax function for the output layer because we are dealing with multiclass classification. These architectures are designed to compare the performance of adaptive dropout with traditional dropout methods as outlined in the original paper (Srivastava et al., 2014).

Code Implementation

During the process of constructing the architectures and models, we utilized PyTorch, Our goal was for comparisons to the realization made by (Srivastava et al., 2014) and to check the results of our improvement against the original dropout as we mentioned, therefore we used the same parameters or optimization function that he mentioned in the article, the learning rate was not mentioned in the article, so we chose it randomly.

For each of the three architectures we implemented three models:
1. A model with a static dropout of p=0.5 for the hidden layers.

2. A model with adaptive dropout that starts at p=0.5, decreases as the epoch progresses and ends at p=0.1 according to the formula we showed in paragraph 2.

3. A model with adaptive dropout that starts at p=0.1, increases as the epoch progresses and ends at p=0.1 according to the formula we showed in paragraph 2.

4. A model with static dropout of p=0.5 using a geometric distribution instead of Bernoulli for the hidden layers.

In MLP architectures our implementation, we utilize stochastic gradient descent (SGD) as our optimization algorithm to train the models. The configuration of the SGD optimizer in our code includes parameters such as learning rate (lr=0.01), momentum (momentum=0.95) and 10 epochs. This consistent approach across different model architectures ensures that the evaluation of each model's performance is fair and unbiased, allowing us to accurately compare their effectiveness under similar training conditions. Finally, we use cross-entropy loss function, and the classification error metric to evaluate the model's performance.

4. Results

<u>MNIST dataset</u>

MLP -Sigmoid: All three variations (normal, ascending, and descending) start with high training and test classification errors, with the training error being notably higher compared to the test error, this suggests that three models generalize well to unseen data. The classification errors for all models decrease rapidly in the first 2-3 epochs, indicating quick initial learning.

The ascending model consistently shows the lowest classification errors but, in the end, the descending model overtaking him, suggesting better generalization. After the initial rapid decrease, the classification errors curve flattens out, Finally, the descending dropout model achieves the lowest test error, followed closely by the ascending model, with the normal dropout model showing the highest test loss. Although the differences are not large, we are talking low loss for everyone, so a small change can be significant in the resolution of these.

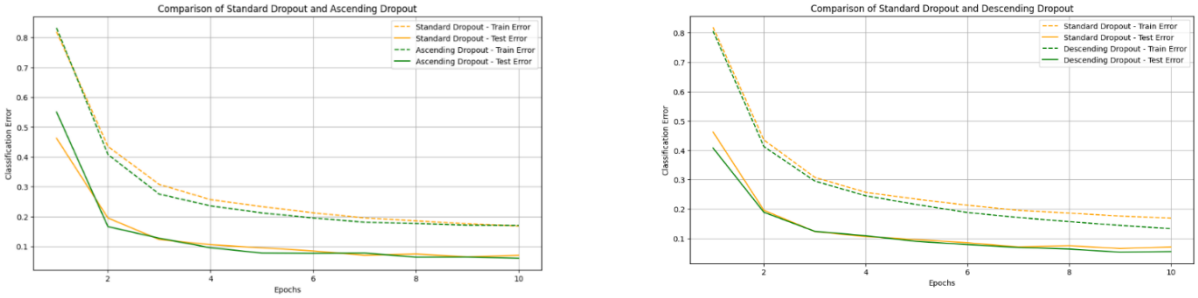


Figure 4: plots of the train and test classification errors of different strategies on MNIST dataset with MLP -Sigmoid Architecture. **left** Ascending Dropout, **right** Descending Dropout.

MLP-ReLU-max norm (Figure 4):

The classification errors for the MLP-ReLU-max norm model show different characteristics: Initial Loss: Starting losses are lower compared to the sigmoid model. The errors decrease is more gradual compared to the sigmoid model, suggesting a different learning dynamic. The gap between training and test errors is much more pronounced than in the sigmoid model, particularly for the ascending and normal models. indicating better generalizability and better performance than the model with the sigmoid. By epoch 10, the test losses converge to similar values for all models, while training losses show more variation.

The two adaptive models gave better results than the normal dropout, with a greater splendor than the sigmoid model. where the descending model presented the best results followed by the ascending model.

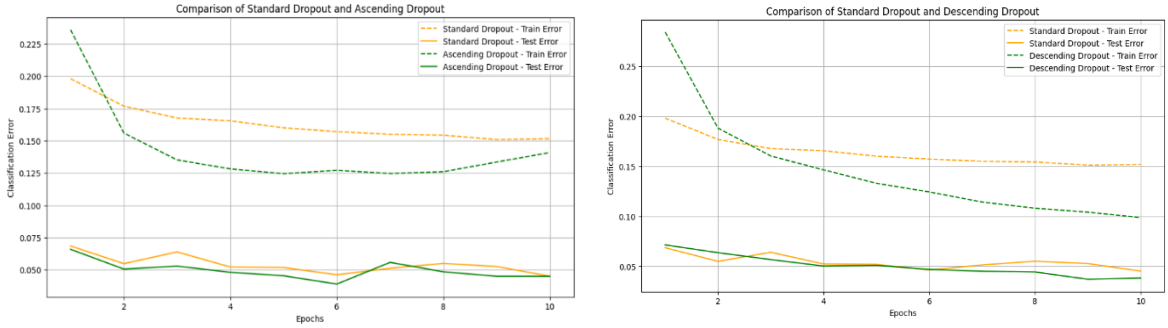


Figure 5: plots of the train and test classification errors of different strategies on MNIST dataset with MLP -ReLu max-norm Architecture. **left** Ascending Dropout, **right** Descending Dropout.

CNN model

Both variations, dropout growth and dropout decay, demonstrate a quick initial reduction in classification errors, reflecting rapid learning. The dropout growth model initially shows lower classification errors but is ultimately overtaken by the dropout decay model. The dropout decay model achieves the lowest test error in the final epochs, suggesting it has the best generalization among the models compared.

While the differences in classification errors are small, these improvements highlight the effectiveness of dropout decay strategy in enhancing model.

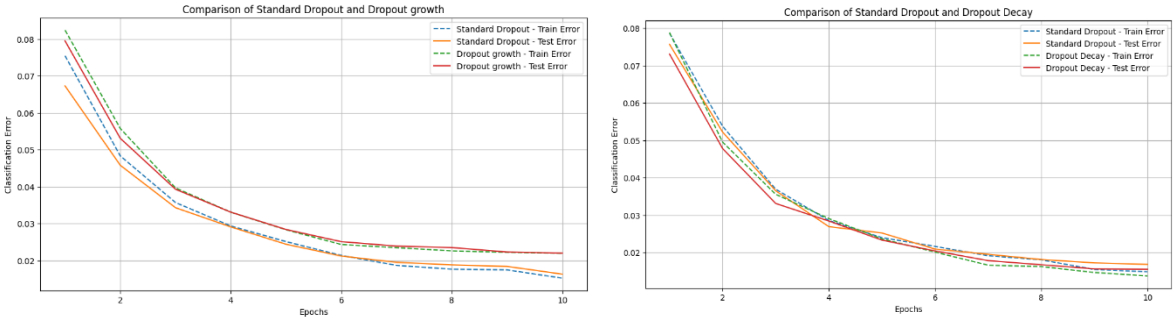


Figure 6: plots of the train and test classification errors of different strategies on MNIST dataset with CNN Architecture. **left** Ascending Dropout, **right** Descending Dropout.

CNN model with different distribution- Geometric

Both variations, standard dropout and geometric dropout, demonstrate a quick initial reduction in classification errors, reflecting rapid learning. The geometric dropout model initially shows slightly lower classification errors compared to the standard dropout model. However, as training progresses, the differences between the models become more apparent. The geometric dropout model achieves the lowest test error in the final epochs, suggesting it has better generalization among the models compared.

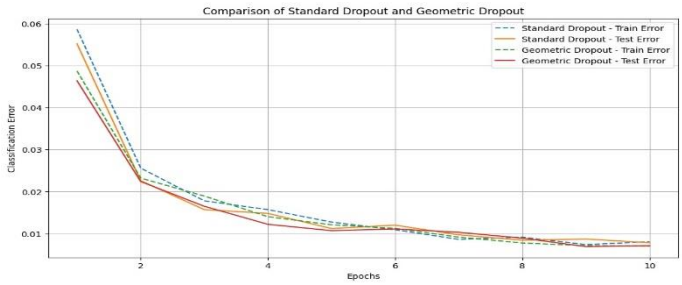


Figure 7: plots of the train and test classification errors on MNIST dataset with CNN Architecture with geometric distribution.

CIFAR-10 dataset

In general, the MLP models show worse results on CIFAR-10 than the MNIST since this is a more complex data set, therefore CNN also gave much better results here, which is more suitable for identifying such a data set. For MNIST 10 epochs were enough, due to lack of time and computing capabilities we did not run more epochs for the CIFAR-10 models.

MLP -Sigmoid: All models show a decreasing trend in classification errors over epochs, with test error generally lower than training error. The descending dropout model achieves the lowest final test error. The ascending dropout model closely follows the descending model in test error. The normal dropout model shows the higher test error. Minimum classification errors are

relatively high, suggesting limited performance of sigmoid activation on this task or the reason we mentioned above.

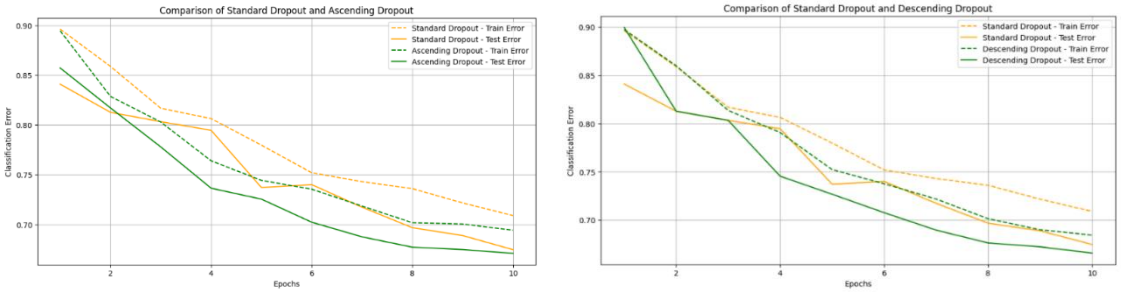


Figure 8: plots of the train and test classification errors of different strategies on CIFAR-10 dataset with MLP -Sigmoid Architecture. **left** Ascending Dropout, **right** Descending Dropout.

MLP-ReLU-max norm: There's a significant improvement in error compared to the sigmoid activation, with final test accuracies reaching close to 50%. All models show more stable and consistent improvement over epochs compared to the sigmoid activation. The maximum test accuracy of around 49% indicates substantially better performance than the sigmoid models. The descending dropout model and ascending dropout model show very similar final test accuracies, both outperforming the normal dropout model.

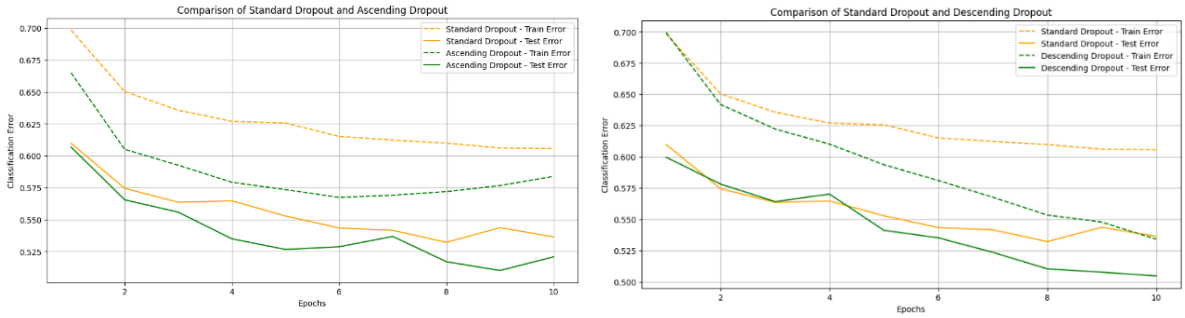


Figure 9: plots of the train and test classification errors of different strategies on CIFAR-10 dataset with MLP -ReLu max-norm Architecture. **left** Ascending Dropout, **right** Descending Dropout.

CNN model:

All variations, standard dropout and dropout growth, demonstrate a rapid initial reduction in classification errors, reflecting quick learning. The dropout growth model consistently shows slightly lower classification errors than the standard dropout model, especially in the final epochs. This suggests that the dropout growth strategy may provide a marginal improvement in performance and generalization. While the differences in classification errors are minor, these results highlight the potential benefits of using dropout growth to enhance model performance.

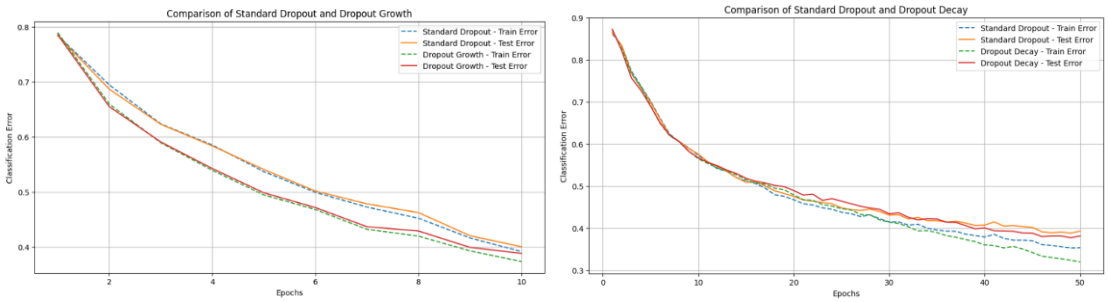


Figure 10: plots of the train and test classification errors of different strategies on CIFAR-10 dataset with CNN Architecture. **left** Ascending Dropout , **right** Descending Dropout.

CNN model with different distribution- Geometric:

Both variations, standard dropout and geometric dropout, demonstrate a consistent reduction in classification errors over the epochs, reflecting steady learning progress. The geometric dropout model initially shows marginally lower classification errors compared to the standard dropout model. As training progresses, the differences between the models remain subtle but become more noticeable. The geometric dropout model achieves the lowest test error in the final epochs, suggesting it has slightly better generalization among the models compared.

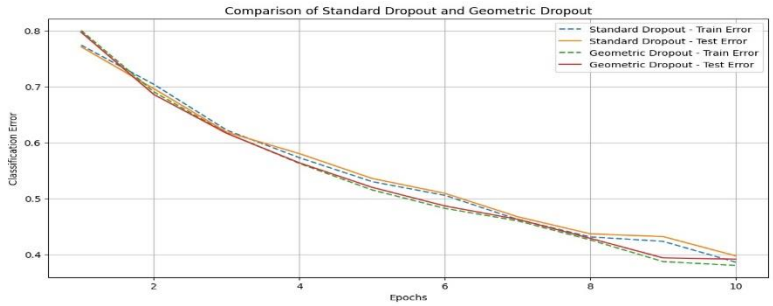


Figure 11: plots of the train and test classification errors on CIFAR-10 dataset with CNN Architecture with geometric distribution

5. Conclusions

Our comprehensive analysis of different models and dropout strategies on the MNIST and CIFAR-10 datasets reveals that adaptive dropout techniques significantly enhance model performance and generalization. In the MNIST dataset, both MLP and CNN models benefit from adaptive strategies, with the descending dropout model achieving the lowest test errors, suggesting superior generalization. The ascending model also performs well initially but is eventually surpassed by the descending model. For the CIFAR-10 dataset, which is more complex, CNN models outperform MLP models, with the geometric dropout strategy showing consistent improvement in test errors over standard dropout. Across both datasets, the descending dropout consistently yields the best results, highlighting its effectiveness in reducing classification errors and improving model robustness. These findings emphasize the importance of employing adaptive dropout strategies to optimize model performance, particularly in more complex datasets.

Individual Contributions

Shahar- responsible for writing the proposal and coding the decay and geometric distribution methods.

Avi- contributed by writing the introduction and methodology sections and developing the code for the growth method.

Paz and Lior- worked on writing the conclusions, abstract, and results, preparing the presentation, selecting architectures and databases, and organizing and unifying the code ,the presentation and report.

6. Bibliography

Ba, L. J., & Frey, B. (2013). Adaptive dropout for training deep neural networks. *Advances in Neural Information Processing Systems*.

Molchanov, D., Ashukha, A., & Vetrov, D. (2017). Variational dropout sparsifies deep neural networks. *34th International Conference on Machine Learning, ICML 2017*, *5*.

Peter R., H. (1979). Synaptic density in human frontal cortex - Developmental changes and effects of aging. *Brain Research*, *163*(2). https://doi.org/10.1016/0006-8993(79)90349-4

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, *15*.