Measuring Robustness in Deep Neural Networks

Contributors: Adi Yafe, Ori Howard, Eitan Kats

Instructors: Roi Weiss, Assaf Hoogi

Introduction

Modern deep neural networks are over-parameterized, yet their generalization ability does not deteriorate. Such networks perform well without overfitting even though the amount of free parameters is greater than the amount of training examples! This observation raised the question of whether these models learn redundant representations.

In this paper, we explore the relationship between dropout, network architecture, and model generalization by incorporating dropout layers at various positions within ResNet18 models trained on CIFAR-10. We attempt to detect if the network learns redundant representation by zeroing out units during testing and checking if the generalization ability retains.

Related Work

Our research was mainly inspired from the paper “Robustness and/or Redundancy Emerge in Overparameterized Deep Neural Networks”[1] which discusses how increasing overparameterization does not cause the model to overfit. They claim this is because the networks adjust by developing robustness or redundancy. Robustness is the amount of units that can be removed without affecting accuracy, and redundancy is having units with similar activity. These concepts are closely related.[2]

Furthermore, dropout is a popular regularization technique that deals with overfitting. Complex co-adaptation of units during the training of the network is avoided by randomly dropping some of them. As explained in the paper “Dropout: A Simple Way to Prevent Neural Networks from Overfitting”[3], the dropout method provides a suitable solution for overfitting by training randomly sampled sub-networks, all having shared weights, while maintaining reasonable computational time. The method also helps in cases where the training data available is limited. There may not be enough data, especially if the models are large which typically requires more samples. The paper states 0.5 is a good dropout percentage for many tasks, although for input units it is usually 0. We expect to see the effects of the dropout rates in our experiments.[4]

We combine the key aspects of these papers in our research by assessing dropout as a metric for robustness. More specifically, we turned off units in the testing phase to measure how many neurons are necessary to make accurate predictions. Typically, a model does not drop units during testing. It is trained using dropout so during testing the weights need to be multiplied by the dropout rate to compensate.[5] Therefore, the use of *turnoff* is unconventional and specific to our paper.

---

[1] Stephen Casper et al., “Robustness and/or Redundancy Emerge in Overparametrized Deep Neural Networks,” Under review as a conference paper at ICLR 2020. https://openreview.net/forum?id=S1xRbxHYDr.
[2] Ibid.
[3] Nitish Srivastava et al., “Dropout: A Simple Way to Prevent Neural Networks from Overfitting,” Journal of Machine Learning Research 15 (2014) 1929-1958. https://jmlr.org/papers/v15/srivastava14a.html.
[4] Ibid.
[5] Ibid.

The Goal

The goal of our project was to investigate the presence of redundant neurons and gain valuable insights by conducting experiments on deep neural networks. We searched for a simple and computationally viable way to measure robustness.

To accomplish this, we incorporated dropout layers at different locations within the network, aiming to provide valuable insights related to network architecture and the role of redundant neurons.

We aimed to test the limits and see how models trained with various dropout rates are affected by turnoff. Our expectation is that dropout will prevent co-adaptation of units, and higher turnoff rates will have less effect compared to models trained with lower to no dropout. Setting high turnoff values should allow us to see this clearly.

Experimental Setup

You can find our result files and code on GitHub: https://github.com/adiy55/Final-Project**.**

Utilized Tools and Frameworks: Python (mainly Jupyter notebooks) and PyTorch in particular to run our Machine Learning models.

Model Architecture: In our experiment we used the ResNet18 model, a deep convolutional network primarily used for image classification tasks. It comprises 18 layers, including residual units or blocks. The key feature of ResNet models is the inclusion of shortcut connections, also known as residual connections, which allow the output of one layer to be directly added to the input of the next layer. These connections address the issues of vanishing gradients and enable improved generalization and easier training of deep networks.

Modifications: During our experiment, we modified the model by adding dropout layers in between residual blocks.

Data: CIFAR-10 is a widely used benchmark dataset. It consists of 60,000 color images, each measuring 32x32 pixels, belonging to 10 different classes (airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck). The dataset is divided into 50,000 training images and 10,000 test images. CIFAR-10 is popularly used due to its diversity of object classes and its relatively small image size, making it computationally efficient for experimentation.

Regularization: Dropout is a regularization technique used in neural networks to combat overfitting. It involves randomly disabling neurons during training, promoting robustness and preventing reliance on specific features. Dropout has proven effective in improving generalization and reducing overfitting.

Models: The models were trained using dropout rates ranging from 0.0 to 0.5 with a step size 0.1, with dropout applied in either the first, middle, or last layer. Furthermore, we trained an additional batch of ResNet18 models with 512 neurons in the first and middle layers to match the last layer, ensuring accurate comparison of results.

Epochs: Each model underwent training for 50 epochs.

Model Initialization: In order to mitigate the randomness associated with model initialization, we initialized a base mode that had predetermined weights that were loaded into the models we trained during our

experimentation. This approach ensured consistency in the initial states of the models and allowed us to make fair comparisons between different dropout rates and locations.

Turnoff Rate: The term “turnoff rate” refers to a percentage of neurons that are not activated during model evaluation. In other words, when we feed the test data to the model a percentage of the neurons have their activation set to 0 . We used turnoff throughout the experiment in various ResNet layers. The turnoff layer is the same layer as the dropout.

Testing Phase: We evaluated the models by computing each model using a range of turnoff rates (ranging from 0.0 to 1.0 with a step size of 0.1). To ensure reliable results, we averaged the outcomes over 20 trials.

Additionally, to comprehensively assess the model’s performance, we repeated the evaluation for each layer with a size of 512 neurons. Typically in ResNet18 models, the first hidden layer consists of 128 neurons and the second with 256 neurons. The last layer has 512 neurons, so modifying the other layers in the model ensures the dropped units are of the same proportions and also adds to the overparameterization.

Plotting the accuracy of the trained models per turnoff rates, we noticed that the ordering of the dropouts is unclear. To look into this further, we evaluated turnoff rates in the range [0.91,1) with step size of 0.02 in the following architectures (Figure 3):

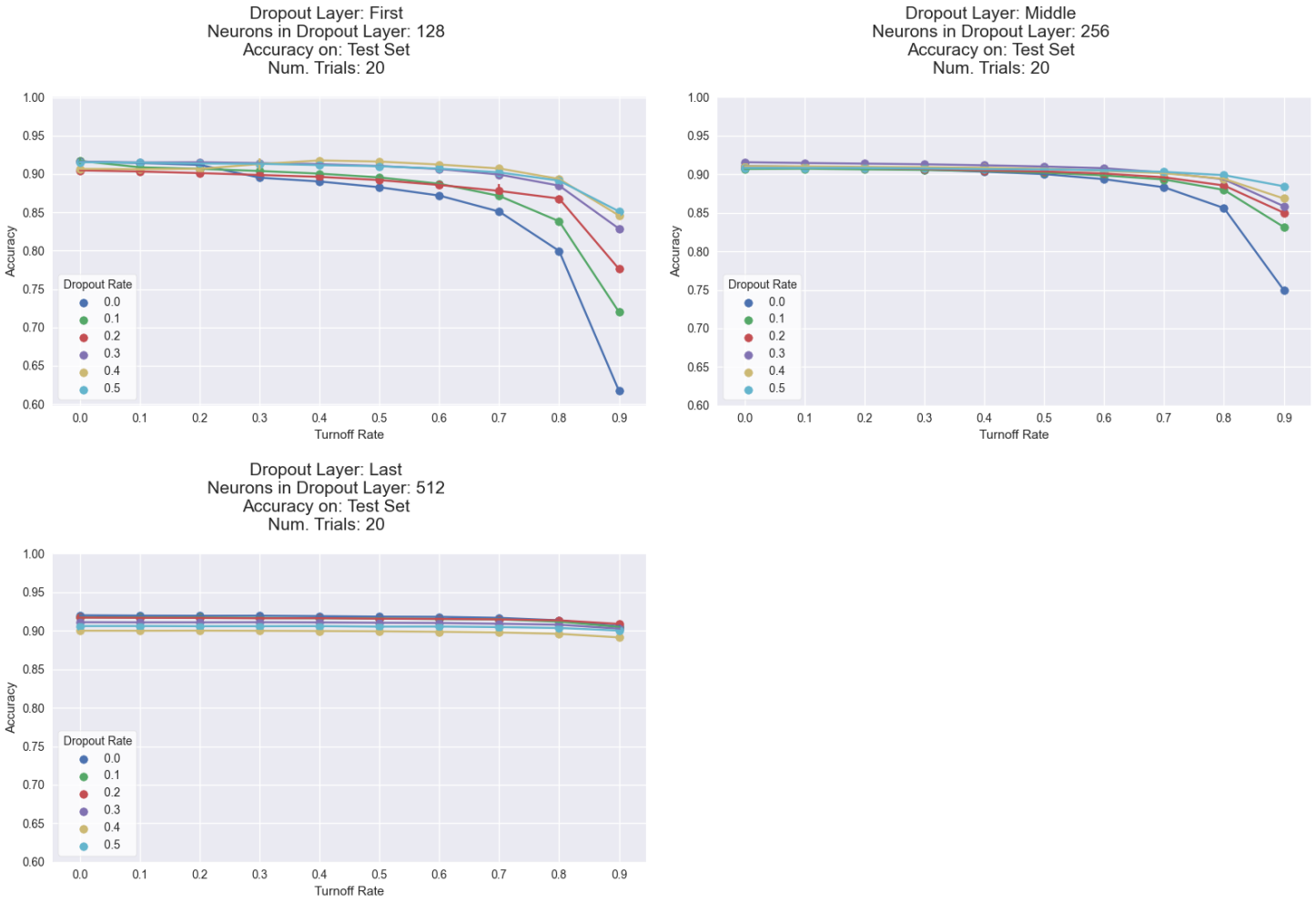
1. Modified version of ResNet18 with 512 neurons in the first layer
2. Modified version of ResNet18 with 512 neurons in the middle layer
3. Original version of ResNet18 with 512 neurons in the last layer

Results

You can view all of our plotted results here**.**

In all of the graphs, each plotted line represents a model and the legend shows the dropout rate used to train it. The X-axis represents the turnoff rate and the Y-axis represents the accuracy of the predictions averaged over 20 trials. Additional information is displayed above each graph.

Figure 1

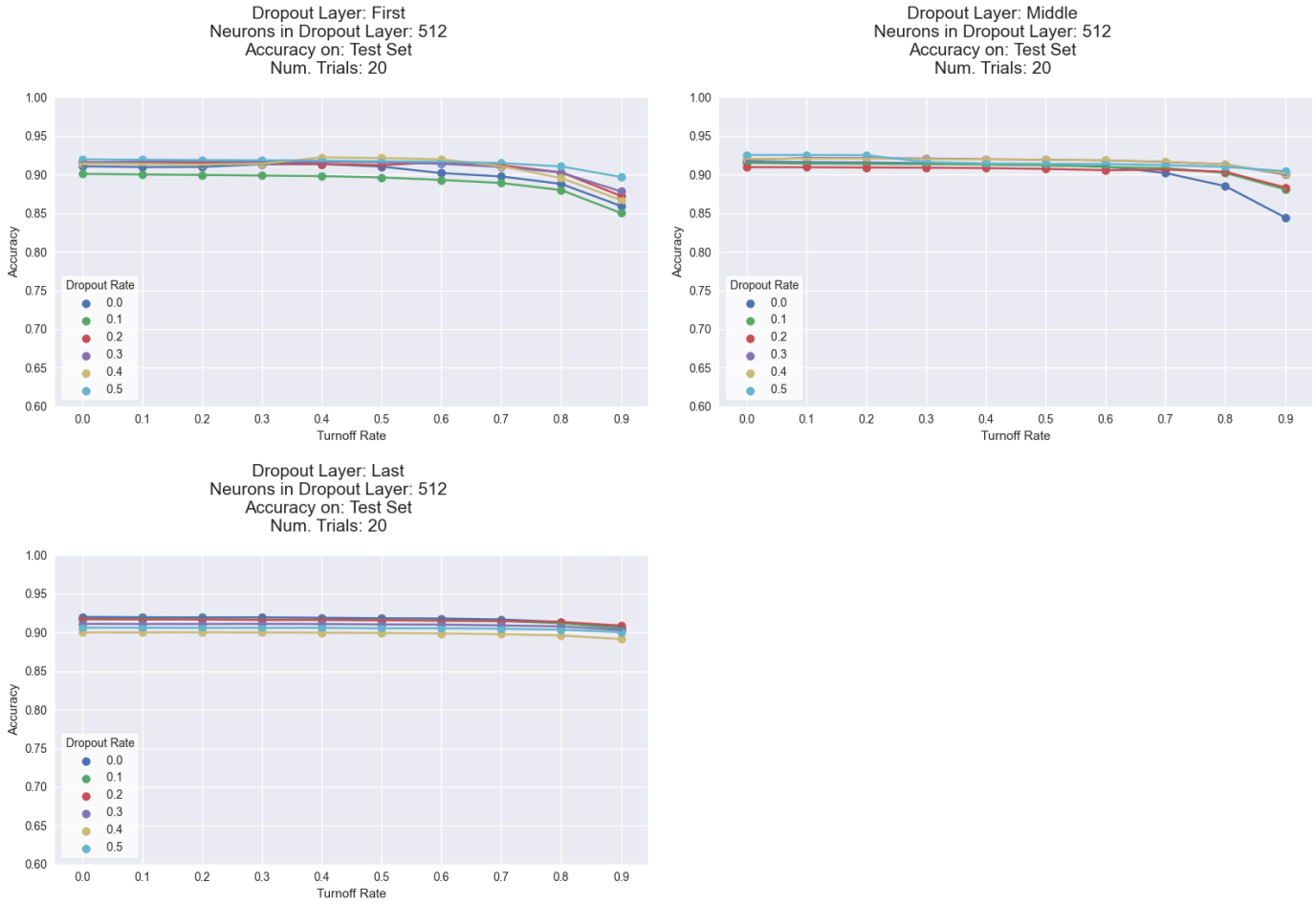


In the initial figure, there are 3 graphs, each representing a dropout location within the model. The original model was employed without any alterations to the layer size.

Upon analyzing the results, we observe that the last layer displays the highest stability in terms of accuracy degradation. Additionally, the first and middle layers, trained with a dropout rate of 0.5, exhibit minimal impact on their accuracy. These particular models seem to possess greater robustness and can withstand the turnoff rates better compared to the other models.

Furthermore, a noteworthy trend is observed: as we move the dropout location through the layers, the degradation of accuracy significantly diminishes, indicating an improvement in stability.

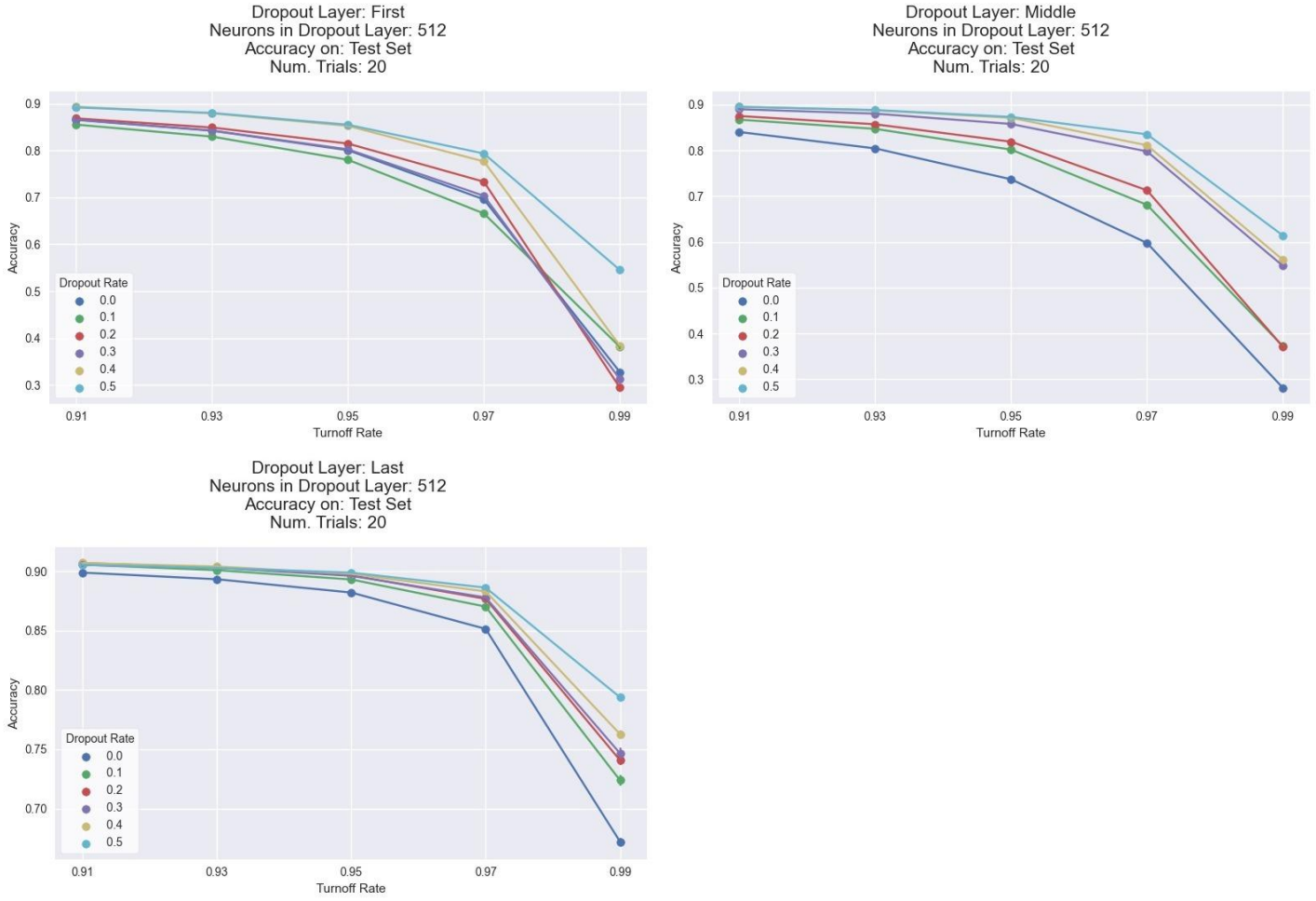
Figure 2



Observing the results in Figure 2, we find that the most stable models among the first 2 layers are the ones trained with a dropout rate of 0.5. This conclusion is drawn from the fact that these models exhibit the smallest accuracy degradation compared to the other models.

Figure 3

Figure 3 displays the turnoff rates, starting from 0.91 and ending at 0.99 with step size of 0.02.



In the graph representing the first layer dropout:
We observed that the model with a 0.5 dropout rate demonstrated remarkable stability. It began with an accuracy of 0.9 and consistently maintained accuracy levels between 0.5 and 0.6 as the turnoff rate increased to 0.99. This implies that even with only 1% of neurons remaining, the model still achieved predictions with over 0.5 accuracy.

However, when analyzing the dropout models from 0.0 to 0.4, we noticed a significant decrease in performance. All these models started with an accuracy close to 0.9 but experienced a substantial decline, ending up with accuracy levels under 0.4 or even 0.3 at 0.99 turnoff, with only 1% of neurons remaining.

In the graph representing the middle layer dropout:
We observed more models exhibiting stability compared to the first layer. Not only the 0.5 dropout rate but also the 0.4 and 0.3 dropout rates achieved accuracy levels above 0.5, even with only 1% of neurons remaining.

However, the dropout models ranging from 0.0 to 0.3 still experienced a significant decrease in performance, with accuracy levels falling below 0.4 when 1% of neurons were retained.

In the graph representing the last layer dropout:
We observed strong stability across all dropout models, ranging from 0.0 to 0.5. Even with only 1% of the neurons remaining, these models consistently achieved accuracy levels over 0.65, with the highest accuracy reaching 0.8 for the 0.5 dropout model.

Conclusion

We generated plots to illustrate the impact of varying dropout rates at different locations on the model's accuracy during the evaluation phase. Our findings revealed that in the last block of the ResNet model, by retaining only 10% of the neurons, we observed a decrease of at most 2% in the model's performance (Figure 1, Last Layer). This result highlights the robustness and redundancy of the model, demonstrating its ability to maintain high accuracy even with a substantial reduction in active neurons.

One of the main results we found was that training models with dropout increases the robustness to the turnoff rate, meaning they maintain better performance on the test set with turnoff. This can be clearly seen in the graphs, where the model trained with 0.5 dropout is less sensitive to turnoff than the one trained without dropout. In other words, more dropout increases the model’s stability to turning off network units.

Additionally, we noticed that the layer size and dropout location impact the accuracy of the model. As the dropout layer is closer to the input layer, the accuracy drops substantially more. This makes sense, as models with dropout in deeper layers have the opportunity to process the data. Regarding the layer size, the dropout rate is relative to the number of neurons. If we train two models with the same dropout rate but with different layer sizes, the model with more neurons will retain more neurons, allowing more information to pass through the network. Our comparison of models that equally have 512 neurons in the layer with dropout was more accurate due to this observation.

Future Work

The results raised discussions about the possible potentials of our turnoff method. We were unable to explore these ideas due to time constraints.

First and foremost, it can be useful to attempt to remove neurons when the model is close to converging. This can be performed gradually by removing more neurons each time and eventually result in a model with few to no redundant neurons, and reduces the model size. We expect this to be most effective with dropout in the last layer since it maintained the stablest accuracy.

Secondly, this method can be used for model compression. Model compression includes training a large model and reducing its size.

Lastly, our method can be utilized for determining the optimal size and architecture of a network.