Experimental design of the Dropout method in Neural Networks

Apratim Banerjee[1], Abirami Sabbani[2], and Lakshya Agarwal[3]

[1]apratimbanerjee@berkeley.edu
[2]abirami.sabbani@berkeley.edu
[3]lakshya@berkeley.edu

May 11, 2024

Abstract

We are exploring the application of the dropout technique to reduce over-fitting in neural networks, using the MNIST dataset for empirical evaluation. Dropout, as a regularization method, randomly disables network neurons during training, thereby inhibiting the development of complex co-adaptations that do not generalize well to unseen data. In this paper, we are further analyzing the concepts presented in [1] by answering critical questions taught in the class **CS294-Experimental Design for Machine Learning**.

1. Introduction

Overfitting remains a significant challenge in the training of deep neural networks. Dropout is a compelling regularization technique that addresses this issue by randomly omitting subsets of features and hidden units at each training stage. This method researched by Srivastava et al. in their 2014 paper, significantly reducing overfitting and improving model generalizability across various domains and dataset types.

2. Methodology

The experimental design for the machine learning technique discussed in the paper, which is focused on dropout as a method for preventing overfitting in neural networks, focusing on image classification tasks using the MNIST dataset. Dropout is a regularization technique that enhances the generalization capabilities of neural networks by randomly omitting a subset of neurons during the training process. This approach prevents the network from forming complex co-adaptations that perform well on the training data but poorly on unseen data as demonstrated by [1]. We also introduce several questions that were introduced in [2] that challenges the dropout technique used and address key aspects of the machine learning model.

2.1 Experimental Design

We implemented our models using the TensorFlow framework, specifically designed to test the effectiveness of dropout regularization in neural networks across different configurations:

- **Data Sets:** The experiments utilize the MNIST dataset, consisting of hand-written digit images for training and testing the model. This dataset is chosen to evaluate the effectiveness of dropout in reducing overfitting in a controlled setting.

- **Network Architecture:**
  - **Convolutional Layer:** The model includes one convolutional layer with 28 filters of size 3×3, using the ReLU activation function and Adam optimizer. This layer is designed to extract features from the input images.
  - **Flattening Layer:** A flattening layer follows to transform the 2D feature maps into a 1D feature vector, making it suitable for the subsequent fully-connected layers.
  - **Fully-Connected Layers:** There are three fully-connected layers in the model. All three layers each consist of a variable number of neurons based on the dropout probability p, calculated as 2048 multiplied by p. Each layer uses the ReLU activation function and Adam optimizer.
- **Dropout Implementation:**
  - Dropout is strategically applied after each fully-connected layer to randomly drop units during training, thus helping to prevent overfitting. The dropout rate p is a variable parameter in the experiments, reflecting the probability of retaining a unit. The rate varies from 0.1 to 0.99 in the experiments to observe the effect of different levels of regularization.
- **Evaluation Metrics:**
  - The effectiveness of dropout is quantitatively assessed by measuring the classification error rates on the validation portion of the MNIST dataset. The performance metrics are the training and validation (or test) error rates, which provide insights into how well the network generalizes to unseen data compared to its performance on the training set.

This design aims to provide a robust framework for understanding dropout's role in enhancing generalization in neural networks, utilizing systematic changes in dropout rates and network architectures across standardized tasks. Figure 1 shows that the error rate decreases as the dropout probability increases.

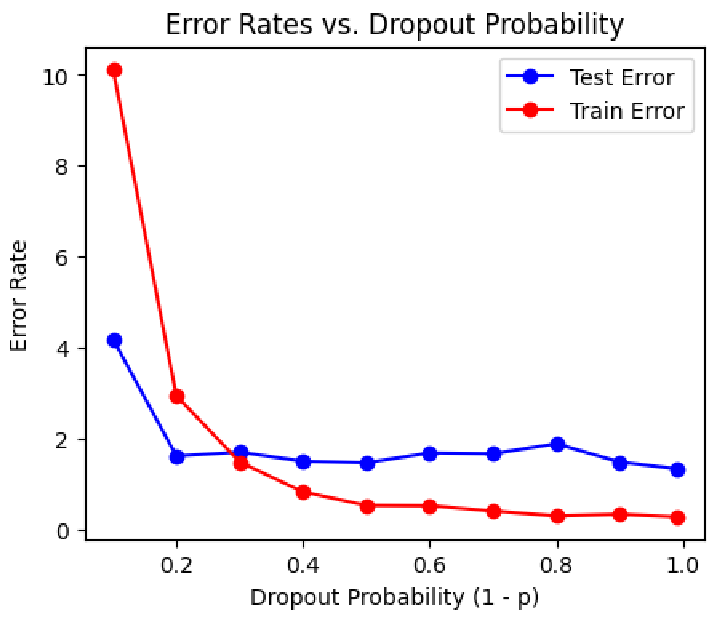


Figure 1: Impact of Dropout Rate on Model Performance. The figure shows training and test errors for models trained with varying dropout rates.

2.2 Final Project Questions

2.2.1 What is the variable the machine learner is supposed to predict? How accurate is the labeling? What is the annotator agreement?

The machine learner is supposed to predict the handwritten digits from the MNIST dataset. MNIST has grayscale images of handwritten digits (0 through 9). The model has to classify each image into one of these 10 categories. The output layer of the model uses a softmax activation function applied to 10 neurons, each representing one of the digit classes, to output the probabilities that the input image belongs to each digit class. The model predicts the digit class by selecting the neuron with the highest probability.

The accuracy of the labeling can be measured as,

$$Accuracy = \frac{\text{Number of correctly classified images}}{\text{Total number of images}} \times 100\% = 99.67\% \tag{1}$$

There is no annotator agreement. Annotator agreement is the level of consistency between different human annotators when they classify or label data and the model. It is very important in datasets where human input plays a significant role in creating the ground truth. The labels in the MNIST dataset are considered to be highly accurate. MNIST has images of handwritten digits which have normalized and centered. They are from 250 people, half which are U.S. Census Bureau employees and half which are high school students. The labels for these images were created with a high degree of precision, so there are generally no significant issues regarding label accuracy or annotator agreement for the MNIST dataset.

2.2.2 What is the required accuracy metric for success? How much data do we have to train the prediction of the variable? Are the classes balanced? How many modalities could be exploited in the data? Is there temporal information? How much noise are we expecting? Do we expect bias?

The required accuracy metric for success is 97% or higher. The script uses the MNIST dataset which has 70,000 images of handwritten digits, split into a standard configuration of 60,000 images for training and 10,000 images for testing. The code splits 10 percent of the training data is used as a validation set during the training process. So, training data - 90 percent of 60,000 images, which equals 54,000 images and validation data - 10 percent of 60,000 images, which equals 6,000 images.

The classes are well balanced as the images of handwritten digits from 0 to 9, distributed almost equally across the 10 classes. This even distribution is important for training neural network models because it prevents the model from developing a bias toward more frequently represented classes.

Since the MNIST dataset consist of grayscale images of handwritten digits where each image is a 28 x 28 pixel array, the primary modality is visual modality. Within visual modality, there are about four aspects that can be exploited for better performance in machine learning tasks. 1) Pixel Intensity - The grayscale values can be used directly as features. These values range from 0 to 255, and they can be normalized to help the model learn more effectively. 2) Spatial Relationships - The arrangement of pixels in terms of shapes and edges that form the digits is important. Convolutional neural networks (CNNs) are used in this dataset. They are effective because they can capture these spatial hierarchies. 3) Image Augmentation - transforming the images through rotations, scaling, translations, and other geometric modifications can create variations that help in training more robust models. This technique artificially increases the diversity of training data and also exploits the geometric properties of the images. 4) Derived Features - Advanced preprocessing techniques could extract specific features like corners, edges, or other significant markers that are particularly distinctive for handwritten digits.

There is no temporal information as MNIST consists of static handwritten images. There is no account of which strokes were made at which point in time.

We expect a very small amount of noise as small variations in pixel intensity and slight misalignments can occur due to the digitization of handwritten notes.

We do not expect significant bias as the classes in MNIST are well-balanced. However, there is a small amount of bias because MNIST is composed of handwritten digits collected from a group of American high school students and employees from the Census Bureau. This specific demographic might not represent global handwriting styles comprehensively. For example, the way digits are written by individuals in different countries or cultural backgrounds might vary, potentially leading to slightly lower performance when the model is applied to digit recognition tasks involving handwriting from non-represented populations.

2.2.3 What is the Memory Equivalent Capacity for the data (as a dictionary). What is the expected Memory Equivalent Capacity for a neural network?

MEC of for data (as a dictionary):

$$MEC = 60,000 \times \log_2(10)$$

Given that $\log_2(10) \approx 3.32193$, the MEC can be computed as:

$$MEC \approx 60,000 \times 3.32193 = 199,315.8$$

MEC for neural network with Dropout depends on the probability of how many neurons are in the layers. Table below calculates the MEC of the Network described previously for varying Dropout Probabilities:

Dropout Probability	**MEC**
0.1	41855.6
0.2	61721.2
0.3	81586.8
0.4	96122.0
0.5	98170.0
0.6	100218.0
0.7	102266.0
0.8	104314.0
0.9	106362.0
0.99	108205.2

Table 1: MEC of Model with Dropout Probability

2.2.4 What is the expected generalization and as a consequence the average resilience in dB? Is that resilience enough for the task? How bad can adversarial examples be? Do we expect data drift?

Expected Generalization

Dropout is designed to improve the generalization of neural networks by preventing them from becoming overly dependent on any specific neurons or features. This enforced redundancy means that the network is less likely to overfit and is much better at handling unseen data from the same distribution. The expected generalization can be measured by their performance improvement after using the dropout technique for varying probabilities.

- Configuration: Three layers, each with 2048 units with MNIST Dataset.
  - Error rate: 1.04%

  - Probability of correct classification, $p_1 = 0.9896$.
  - Self-information: $I_1 = -\log_2(0.9896) \approx 0.0148$ bits.

MEC and Generalization Calculation

Generalization, denoted as G, is defined by the formula:

$$\mathrm{G} = -\frac{(\sum_{i=1}^{c} k_i \log_2 p_i)}{\mathrm{MEC}} \tag{2}$$

where:

- c is the number of classes,
- k_i is the number of correctly classified instances in class i,
- p_i is the probability of the class being i,
- MEC is the memory-equivalent capacity of the machine learner used.

For each dropout probability p_{drop}, we first compute p as:

$$p = 1 - \left(\frac{\text{testError}}{100}\right)$$

Then we substitute p_{i}, k, and MEC into the formula for G. Since the output has 10 classifications, we iterate from $c = 1$ to $c = 10$. We use the average value of p_{i} across all classifications.

For $p_{\textbf{drop}} = 0.1$

- testError $= 4.18$
- $MEC = 41855.6$
- Substituting values, we get $G = 8.8$ bits/bit

$$\vdots$$

For $p_{\textbf{drop}} = 0.5$

- testError $= 1.47$
- $MEC = 61721.2$
- Substituting values, we get $G = 16.3$ bits/bit

$$\vdots$$

For $p_{\textbf{drop}} = 0.99$

- testError $= 1.33$
- $MEC = 108205.2$
- Substituting values, we get $G = 19.4$ bits/bit

We observe that G is positive and greater than 1, as expected. Additionally, we notice that as p_{drop} increases, G increases up to $p_{\text{drop}} = 0.99$ This trend suggests that as the dropout rate increases (leading to a decrease in train/test error), the model becomes more generalizable.

Expected Resilience:

Given the formula for resilience:

$$\text{Resilience (dB)} = 20 \times \log_{10}(G)$$

We calculate the resilience for different values of G associated with various p_{drop}:

- For $p_{\text{drop}} = 0.1$ and $G = 8.8$:

$$\text{Resilience} = 20 \times \log_{10}(8.8) \approx 18.88 \text{ dB}$$

$$\vdots$$

- For $p_{\text{drop}} = 0.5$ and $G = 16.3$:

$$\text{Resilience} = 20 \times \log_{10}(16.3) \approx 24.23 \text{ dB}$$

$$\vdots$$

- For $p_{\text{drop}} = 0.99$ and $G = 19.4$:

$$\text{Resilience} = 20 \times \log_{10}(19.4) \approx 25.75 \text{ dB}$$

Adequacy of Resilience for the Task: For the MNIST dataset, the resilience provided by dropout rate p_{drop} of 0.5 is usually sufficient. This dataset does not involve complex or noisy images, and the task is relatively straightforward compared to more complex image recognition tasks that might involve natural scenes or objects in varying conditions.

Impact of Adversarial Examples: However, dropout alone does not necessarily safeguard against adversarial examples. These are inputs crafted to exploit the model's weaknesses by introducing perturbations that lead to incorrect predictions by the model. The severity of adversarial attacks can vary, but they can be particularly problematic in systems where security and reliability are critical.

Expectation of Data Drift: As for data drift, this refers to the change in the underlying data distribution over time, which can occur due to many factors such as changes in writing styles or the introduction of new types of handwriting. Dropout does not address data drift inherently; it merely helps the model generalize better to new data from the same distribution as seen during training. Continuous model monitoring and potentially retraining with new data are essential strategies to handle data drift effectively.

2.2.5 Is there enough data? How does the capacity progression look like?

There is enough data as the error does not significantly increase when the model is trained on less data. To investigate this, we decreased the amount of train data to 10% and gradually increased it to 100%. An error rate of 3% is considered acceptable because the **required accuracy metric** is 97%. As shown in Figure 2, the accuracy metric is met by training on 30% of the training data. The capacity progression is low at first, increases until about 30% of the training data is used, and then stabilizes.

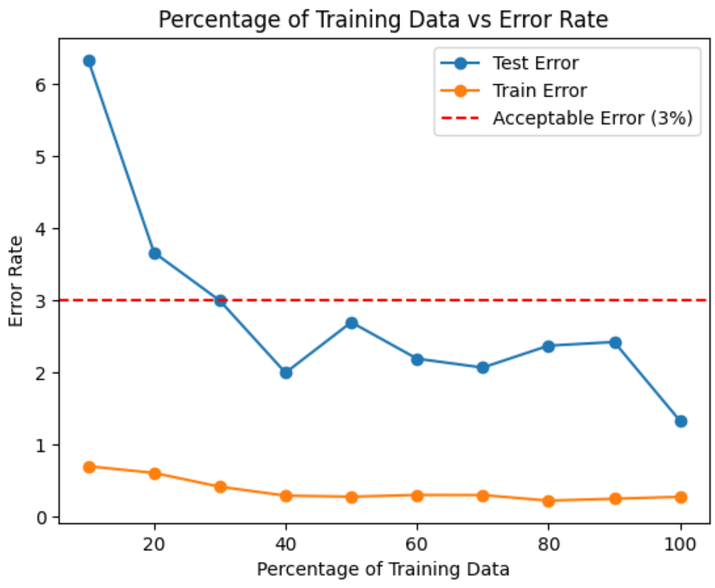


Figure 2: Impact of Dropout Rate on Model Performance. The figure shows training and test errors for models trained with varying dropout rates.

2.2.6 Train your machine learner for accuracy at memory equivalent capacity. Can you reach near 100% memorization? If not, why (diagnose)?

From training the machine learner at memory equivalent capacity, we can reach near 99.67% accuracy which is near 100% memorization. This occurs in 10 epochs. This happens when the input convolution layer has 59 filters of size 3×3 and the probability for retention, $p = 0.7$. Note that when using dropout regularization, during each training iteration, a certain fraction of neurons (determined by the dropout probability 1-p) are randomly set to zero, effectively removing them from the network for that iteration. This prevents the network from relying too heavily on any individual neuron and encourages generalization.

2.2.7 Train your machine learner for generalization: Plot the accuracy/ capacity curve. What is the expected accuracy and generalization ratio at the point you decided to stop? Do you need to try a different machine learner? How well did your generalization prediction hold on the independent test data? Explain results. How confident are you in the results?

When we trained our 3x3 Neural net model, we observed the following:

- The sharp decrease in test error with the initial increase in dropout probability suggests that the model benefits significantly from dropout in terms of improving its generalization to unseen data.
- The test error stabilizes and remains low beyond a dropout probability of about 0.4, suggesting an optimal range for dropout that balances training performance and generalization.
- Higher dropout probabilities do not significantly deteriorate the test performance, which indicates robustness of the model against various levels of noise introduced by dropout.
- The generalization prediction seems to hold well on the independent test data. The lowest test error of 1.33% occurs when the dropout probability is 0.99.
- We are fairly confident in our results our original CNN results and MLP results were similar.

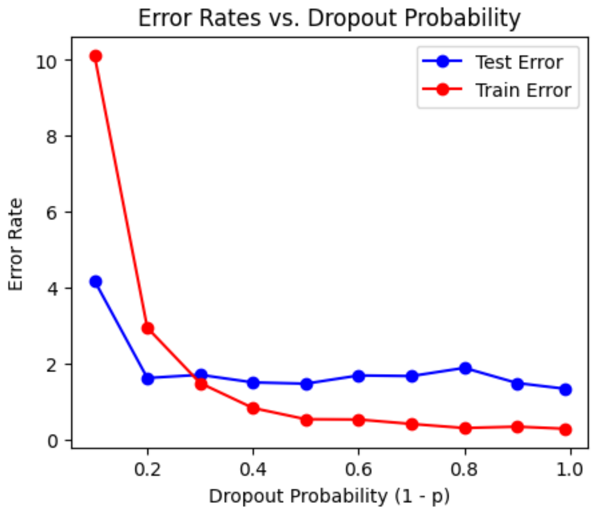
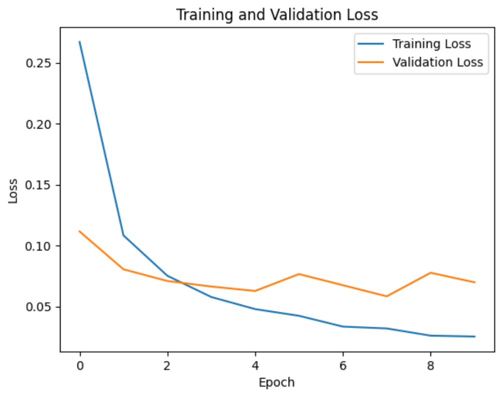


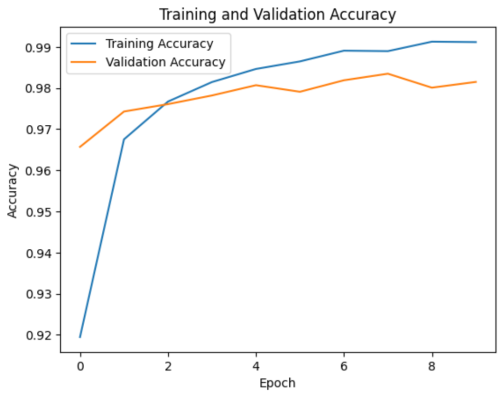
Figure 3: Accuracy curve for our original model

- Figure 3 indicates that the dropout probabilities around 0.4 to 0.6 might offer the best generalizability. At these levels, the test error is minimized and stable, which indicates effective learning that is neither too constrained (high bias) nor too varied (high variance).

When we tried a different machine learner, Multi-Layer Perceptron (MLP) the results for loss as shown in Figure 3 and accuracy as shown in Figure 4 were similar.



(a) MLP Loss Results



(b) MLP Accuracy Results

Figure 4: MLP Results

- The generalization prediction seems to hold well on the independent test data. The lowest test error of 1.33% occurs when the dropout probability is 0.99.
- We are fairly confident in our results our original CNN results and MLP results were similar.

2.2.8 Comment on any other quality assurance measures possible to take/the authors should have taken. Are there application-specific ones? If time is present: How did you deal with it?

We along with the authors could have taken some more quality assurance measures that they may have considered.

- Architecture Choices: The author could have chosen ResNet architecture for better gradient flow and information capacity in hidden layers. Exploring a variety of architectures (like adding layers or varying layer types) could provide deeper insights into how architecture influences model performance and robustness.

- Experimentation with Training Data Size: We reduced the training set size to assess model performance under different capacities. Extending this, we could have systematically varied the amount of training data to create learning curves. This would help identify at what points additional data stops improving the model's performance, or conversely, where performance dips due to insufficient training data.
- Diverse Learning Models: We used different machine learning models to verify the correctness of our results. Expanding this to include a broader range of models, including those with different learning dynamics like decision trees or ensemble methods, could further validate our findings.
- Advanced Dropout Techniques: We could have explored variations in Dropout Techniques like Spatial Dropout, which is designed for convolutional layers, or Alpha Dropout, which is suited for SELU activation functions.
- Long Short-Term Memory (LSTM) Models: LSTM could be specifically helpful given their different handling of dropout. They might provide additional insights into sequence data processing. Testing dropout's efficacy in recurrent architectures would add another layer of depth to your dropout analyses.
- Error Analysis: Systematically analyzing mis-classifications and errors can reveal specific weaknesses in the model, like biases or under-fitting on certain types of data. This could lead to targeted improvements in the model's architecture or training process.

2.2.9 How does your experimental design ensure repeatability and reproducibility?

The experiment ensures repeatability by defining a fixed architecture for the neural network model, which includes specific layers - Conv2D, Flatten, Dense, Dropout and hyperparameters - kernel size, activation functions, optimizer. It also uses a fixed random seed for initializing weights and biases in the neural network model. This ensures that each repetition of the experiment produces consistent initial conditions, leading to consistent results. The MNIST dataset is used for training and testing the model across repetitions. By using the same dataset, the experiment ensures that the input data and labels remain consistent, contributing to the repeatability of results. Additionally, the experiment has a fixed number of epochs and batch size for training the model. These parameters remain constant across repetitions, ensuring consistent training procedures and convergence criteria for error rate and accuracy.

The experiment ensures reproducibility in different ways. It follows standard practices for preprocessing the MNIST dataset, such as reshaping the input images and normalizing pixel values. These preprocessing steps are well-documented and reproducible, enabling other researchers to replicate the experiment. The experiment reports performance metrics (test error and train error) for different dropout probabilities, allowing other researchers to verify and reproduce the results by implementing the same experimental setup and analyzing the reported metrics. Also, the experiment provides detailed descriptions of the model architecture, training procedure, and evaluation metrics so future researchers can understand and replicate the experiment. The results can be accessed at [3].

3. Conclusion

Our experimental design challenges a variety of features and techniques used in the model. By varying dropout rates we not only enhance model generalization but also improve resilience to common issues such as overfitting and data noise. The systematic exploration of dropout probabilities from 0.1 to 0.99 provided valuable insights into the optimal settings that balance training efficiency and model accuracy, with the dropout rate around 0.5 being particularly effective for this dataset. While the results are promising, notice certain limitations the CNN model we were using, and hence introduced the MLP machine learner which was necessary to fully understand the potential and limitations of the dropout technique. Additionally, expanding comparisons to other regularization techniques or network architectures like ResNets could further validate and refine the use of dropout. We also introduced several probing questions designed to rigorously test the dropout technique within neural networks, addressing key aspects such as predictive accuracy and label reliability of the dataset. It assesses the sufficiency and balance of data for meaningful experimentation and inquires about Memory Equivalent Capacity (MEC) to explore computational demands. The paper also examines dropout's capability to enhance model generalization and resilience, crucial for real-world applications, and how well it holds up against adversarial examples and potential data drift. Additionally, it looks at capacity progression and training optimization to determine efficient strategies for deploying dropout without compromising on model performance. These questions not only challenge the experimental setup but also underline the robustness of dropout, guiding future research to refine its applicability in more complex scenarios.

Contributions

All three team members, Apratim, Abirami, and Lakshya worked equally on the project.

References

[1] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, & R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, 15, pp. 1929–1958, 2014.

[2] G. Friedland, *Information-Driven Machine Learning: Data Science as an Engineering Discipline*. Springer, 2024. DOI: 10.1007/978-3-031-39477-5. [Online]. Available: https://link.springer.com/book/10.1007/978-3-031-39477-5.

[3] A. Sabbani, A. Banerjee, & L. Agarwal, *Cs294-82 final project github repository*, 2024. [Online]. Available: https://github.com/abiramisab/CS294-82_FinalProject.