NNDL Lab Assignment – 2

Divya Sai Sindhuja Vankineni

Overview

This experiment aims to investigate how different regularization techniques influence model data efficiency. For this, we design a baseline model on a chosen dataset that has at least 1000 samples and is suitable for multi-class classification tasks. The baseline model contains 5 non-linear activation layers along with 1 learned layer in between each of them. The model is trained using the cross-entropy loss with the SGD optimizer for the chosen epochs such that the model converges well. Once, the baseline model matches the standard performance metric of established models, we fix on the number of epochs, E. Then, the baseline model is trained using 25%, 50% and 75% of the training data to measure the baseline data efficiency. We then perform architectural regularization and data/loss regularization experiments on the baseline architecture to investigate their influence on the model data efficiency.

Experimental Design

Objective

The goal of this experiment is to investigate the impact of different regularization techniques on the performance and data efficiency of a Convolutional Neural Network (CNN) trained on the CIFAR-10 dataset. Specifically, we aim to study:

1. The impact of Dropout Regularization - Experiment 1
2. The impact of Data/Loss Regularization including L2 regularization and data augmentation techniques - Experiment 2

on the model's ability to generalize across various data sizes (25%, 50%, 75%, and 100%) of the training dataset. We also investigate how these regularization techniques affect the model's performance metrics (test accuracy, precision, recall) and how they influence the data efficiency of training.

Experimental Setup

- Baseline Architecture:

  The CIFAR-10 dataset is chosen to conduct this experiment as it is easily available from the existing torchvision library in PyTorch and has enough images (60000) for conducting this experiment. However, for simplicity I chose 2500 random images from the total dataset to conduct these experiments. The CIFAR-10 dataset consists of 32x32 color images in 10 classes, which is appropriate for our multi-class classification experiment.

  2D-Convolutional layers are thus used as this is an image dataset. I selected the standard 3x3 kernel (filter) size for each convolutional layer so that it captures local patterns efficiently and padding of 1 is applied after each convolution to preserve the spatial dimensions of the input image. For the model to capture patterns ranging from simple (such as edges, textures etc.) to complex features, I doubled the number of filters after each layer.

ReLU activation is chosen as it is non-linear and prevents the vanishing gradient problem. After each convolution, a max pooling layer with a 2x2 kernel is used to down sample the feature maps by a factor of 2. This is done to reduce the number of parameters and computational complexity, while retaining the most important spatial features.

The final fully connected layer maps the flattened output to a vector of size 10, corresponding to the 10 classes of the CIFAR-10 dataset. After experimenting with the number of epochs, E = 15 is giving good convergence on the model, hence the model is trained for 15 epochs every time. Thus, the baseline model is a convolutional neural network (CNN) designed to classify images from the CIFAR-10 dataset. It consists of 5 convolutional layers followed by 1 fully connected layer. The architecture is as follows:

- Convolutional Layer 1:
    - Input: 3x32x32 (RGB image of size 32x32)
    - Operation: 2D Convolution
    - Filters: 32 filters, each of size 3x3
    - Padding: 1 (to preserve spatial dimensions)
    - Activation: ReLU (Rectified Linear Unit)
    - Pooling: Max Pooling (2x2) to down sample the output
- Convolutional Layer 2:
    - Input: 32x32x32 (after Conv1 and pooling)
    - Operation: 2D Convolution
    - Filters: 64 filters, each of size 3x3
    - Padding: 1
    - Activation: ReLU
    - Pooling: Max Pooling (2x2)
- Convolutional Layer 3:
    - Input: 64x16x16 (after Conv2 and pooling)
    - Operation: 2D Convolution
    - Filters: 128 filters, each of size 3x3
    - Padding: 1
    - Activation: ReLU
    - Pooling: Max Pooling (2x2)
- Convolutional Layer 4:
    - Input: 128x8x8 (after Conv3 and pooling)
    - Operation: 2D Convolution
    - Filters: 256 filters, each of size 3x3
    - Padding: 1
    - Activation: ReLU
    - Pooling: Max Pooling (2x2)

- Convolutional Layer 5:
  - Input: 256x4x4 (after Conv4 and pooling)
  - Operation: 2D Convolution
  - Filters: 512 filters, each of size 3x3
  - Padding: 1
  - Activation: ReLU
  - Pooling: Max Pooling (2x2)
- Fully Connected Layer:
  - Input: 512x1x1 (flattened after Conv5 and pooling)
  - Operation: Linear transformation to 10 output classes (for CIFAR-10 classification)
  - Output: 10 (one output for each class in the CIFAR-10 dataset)
- Optimizer: Stochastic Gradient Descent (SGD) with a learning rate of 0.1.
- Loss Function: Cross-entropy loss.

- Hyperparameters (fixed across all experiments):
  - Batch Size: 32
  - Learning Rate: 0.1
  - Epochs: 15
  - Optimizer: SGD (with momentum)
  - Regularization: In Experiment 1, dropout regularization is applied to the model. In Experiment 2, L2 regularization and data augmentation are used separately.

Experiments and Hypothesis

- Experiment 1: Evaluating the impact of Dropout Regularization (rate = 0.25)

  Dropout was applied as a regularization technique during training. This method randomly drops (sets to zero) a fraction of the neurons in each layer during training, forcing the model to rely on a wider range of neurons and features, thus preventing overfitting and encouraging generalization.

  - Hypothesis: Dropout will help the model generalize better, particularly when trained on smaller subsets of the data (25%, 50%).
- Experiment 2: Evaluating the impact of L2 Regularization

  L2 regularization (weight decay) was applied by adding a penalty to the loss function based on the squared values of the model's weights. This regularization technique helps prevent the model from learning overly large weights, which can lead to overfitting, especially with small datasets.

- Hypothesis: L2 regularization (weight decay) will help reduce overfitting by penalizing large weights, leading to better generalization, especially when the model is trained on larger data subsets (75%, 100%).

- Experiment 3: Evaluating the impact of Data Augmentation

  Data augmentation was performed by applying random transformations such as horizontal flipping and rotations to the input images. This artificially increases the size of the training set, enabling the model to generalize better by learning from more diverse examples, even with a limited dataset.

  - Hypothesis: Data augmentation will increase the diversity of the training data, allowing the model to generalize better to unseen data, particularly when trained on smaller data subsets (25%, 50%).

Training on Balanced Class Distribution Datasets

The model is trained on 25%, 50%, 75%, and 100% of the CIFAR-10 training data to simulate different data efficiency scenarios. The performance of the model is compared across different regularization techniques and data sizes.

Results

Table 1: Test Accuracy for each Regularization Technique and Data Size

Regularization Technique	*25% of Training Data*	*50% of Training Data*	*75% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	67.133	69.126	70.226	71.793
Dropout Regularization	70.793	72.72	74.906	75.166
L2 Regularization	68.066	70.246	71.96	72.186
Data Augmentation	68.586	69.953	70.473	72.7

Table 2: Average Precision for each Regularization Technique and Data Size

Regularization Technique	*25% of Training Data*	*50% of Training Data*	*75% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	68.419	70.523	71.807	72.879

Dropout Regularization	71.864	72.175	74.397	76.322
L2 Regularization	70.804	71.136	72.368	73.179
Data Augmentation	69.673	70.128	72.095	73.724

Table 3: Average Recall for each Regularization Technique and Data Size

Regularization Technique	*25% of Training Data*	*50% of Training Data*	*75% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	67.103	69.077	70.230	70.905
Dropout Regularization	70.791	71.512	74.853	75.178
L2 Regularization	68.037	70.112	71.668	72.462
Data Augmentation	68.466	69.046	70.570	72.578

All the performance metrics (test accuracy, precision, recall) range from around 50 to 75 and follow similar trends. The baseline model gives moderately good performance while the L2 regularization gives the least and the Dropout regularization technique gives the highest performance on all the data sizes. Overall, 25% data size seems to be the least performing, which might be influenced by factors like class imbalance or random initialization.

Figure 1: Test Accuracy vs. Training Data Size with Regularization Techniques (Line Plot)

This line plot illustrates the test accuracy as a function of the amount of training data used (x-axis) for different regularization techniques (y-axis).

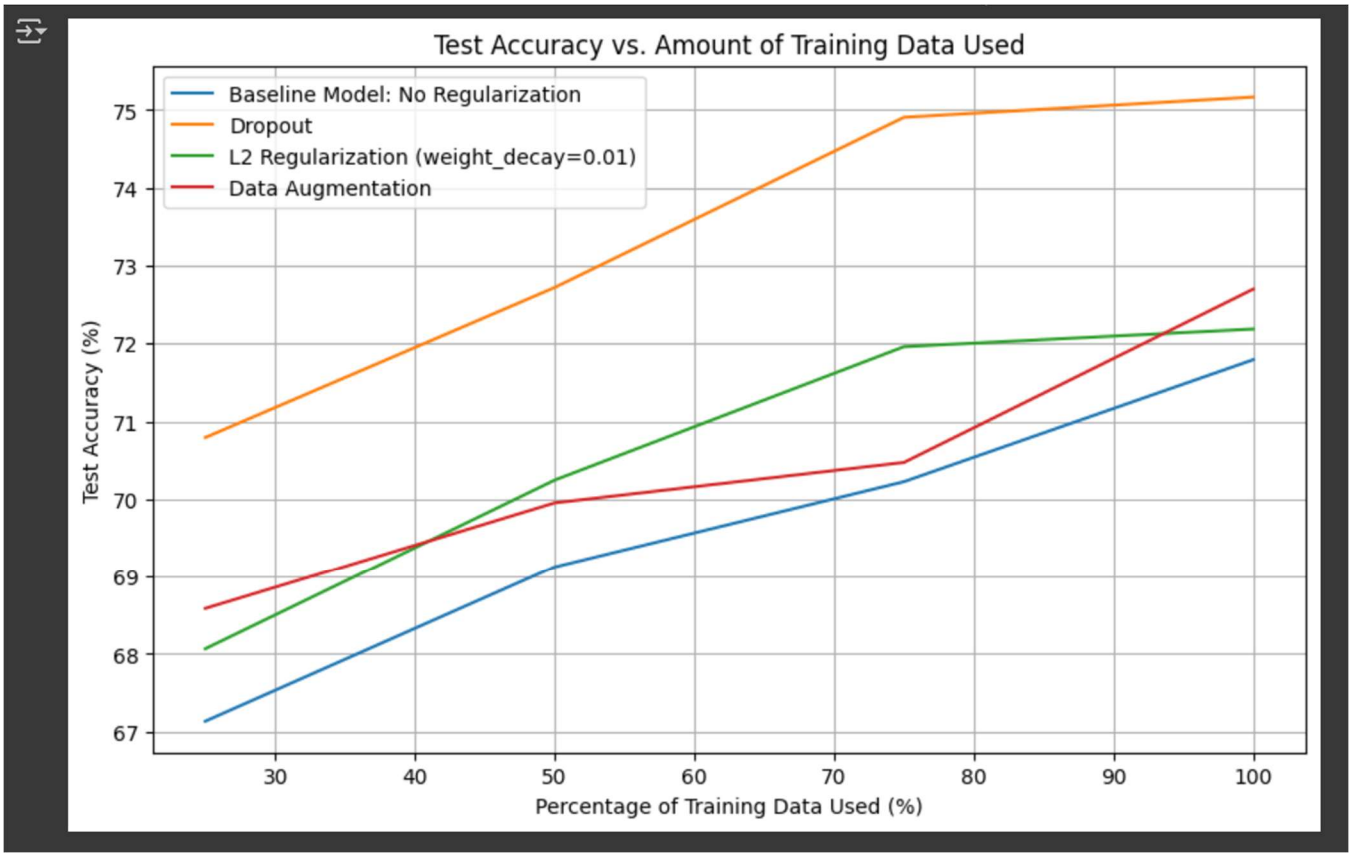
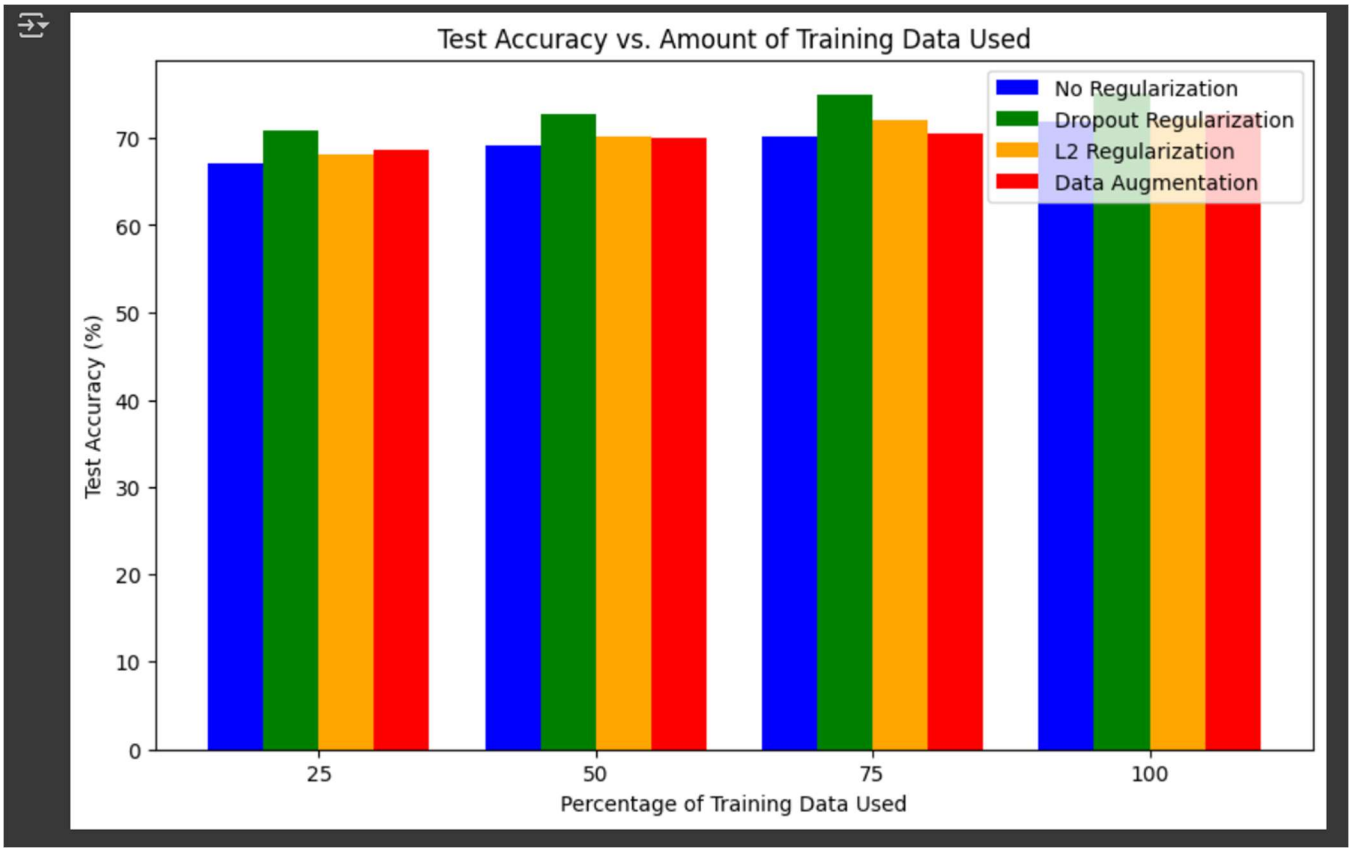


Figure 2: Test Accuracy vs. Training Data Size with Regularization Techniques (Bar Chart) This bar chart illustrates the test accuracy as a function of the amount of training data used (x-axis) for different regularization techniques (color-coded bars).



Analysis

Trend 1: Impact of Dropout Regularization

From the results, we observe that Dropout Regularization helps improve generalization, especially when training with smaller data subsets (25% and 50%). The baseline model without dropout shows lower test accuracy, as the model tends to overfit due to the limited data. Dropout regularization, which randomly disables a portion of the neurons during training, forces the model to rely on a broader set of features, resulting in better generalization. This is evident from the significant improvements in test accuracy, particularly in the 25% and 50% data cases.

Trend 2: Effect of L2 Regularization (Weight Decay)

From the results, we observe that L2 regularization (weight decay) provides the least improvements in generalization, and it seems to have less impact compared to other regularization techniques like Dropout. The performance metrics using L2 regularization are lower overall, especially for smaller data sizes (25%, 50%). L2 regularization can sometimes hurt performance when too much regularization is applied, especially if the model is not complex enough for the task. I think this might be the reason for the bad performance of L2 regularized model.

Trend 3: Impact of Data Augmentation

From the results, we observe that Data Augmentation provides consistent improvements in generalization, especially when training with smaller data subsets (25% and 50%). By artificially

increasing the size and variability of the training data through transformations like random rotations and horizontal flips, the model learns more robust features and is less prone to overfitting. This is evident in the higher test accuracy when using data augmentation, particularly in the 25% and 50% data cases, where the model benefits from the expanded training set. With larger data subsets (75% and 100%), the improvements from data augmentation are still visible, but they become less pronounced as the model has more real data to train on.

Optimal Trade-off Between Data Size and Regularization

The experiments suggest that there is a trade-off between the amount of training data and model performance. As expected, training with more data leads to better performance across all regularization techniques. However, the regularization techniques (Dropout, L2 Regularization, and Data Augmentation) help the model perform better with smaller datasets, closing the performance gap when less data is used. This highlights the importance of regularization in scenarios where obtaining large amounts of labeled data is expensive or time-consuming.

Conclusion

In summary, both Dropout Regularization and Data/Loss Regularization (L2 regularization and data augmentation) provide valuable benefits in improving model performance and data efficiency. Dropout regularization helps the model generalize better by preventing overfitting, especially on smaller datasets. L2 regularization controls model complexity, and data augmentation introduces diversity to the training set, improving performance on smaller datasets. Combining these techniques provides the best results, emphasizing the importance of regularization for efficient training with limited data.

Extra Credit

To investigate how model depth relates to regularization techniques and performance, we modify our baseline model to contain 15 non-linear activation layers (ReLU) with one learned layer in between each of them. We train this model for E = 15 epochs using the same hyper-parameters as the previous experiments. This model is then trained with only 50% of the training data and the results are noted. Next, the best performing regularization technique from our previous experiments, i.e., Dropout is applied to this new model and trained for 50% and 100% of the training data, noting down the results.

Overview of the architecture:

- Input Layer: CIFAR-10 image input (32x32x3).
- Convolutional Layers: 15 convolutional layers, each followed by a ReLU activation and first 5 layers are followed by a MaxPool in addition to ReLU for dimensionality reduction.
- Fully Connected Layer: A final fully connected layer leading to the 10-class output (as the CIFAR-10 dataset consists of 10 classes).

Regularization Technique Chosen:

Based on the results from Experiments 1 and 2, I selected **Dropout Regularization** as the best-performing technique. Dropout was particularly effective in preventing overfitting, especially when training on smaller datasets (25% and 50%), by randomly disabling certain neurons during training. This technique

forces the model to generalize better and prevents it from relying too heavily on specific neurons, which would otherwise lead to overfitting.

Experiment Setup and Results:

The deep model was trained under the following conditions:

- Without Regularization: The deep model was first trained without any regularization technique using 50% of the training data.
- With Dropout Regularization: The deep model was trained with **Dropout Regularization** (rate = 0.25) using both 50% and 100% of the training data.

The training was done for 15 epochs with the same hyperparameters as the previous experiments (batch size = 32, learning rate = 0.1, optimizer = SGD).

4. Results and Discussion:

Table 1: Test Accuracy without and with Dropout Regularization Technique vs Data Size

Regularization Technique	*50% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	37.426	
Dropout Regularization	50.725	65.196

Table 2: Average Precision without and with Dropout Regularization Technique vs Data Size

Regularization Technique	*50% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	39.323	
Dropout Regularization	51.29	67.085

Table 3: Average Recall without and with Dropout Regularization Technique vs Data Size

Regularization Technique	*50% of Training Data*	*100% of Training Data*
Baseline (No Regularization)	37.193	
Dropout Regularization	50.857	65.714

Without regularization and with only 50% of the data, the deep model exhibited significant overfitting, with lower test accuracy, precision and recall. This highlighted the need for regularization. When applying Dropout Regularization, the model's performance improved across all metrics—test accuracy, precision, and recall—in both 50% and 100% data scenarios. Dropout helped prevent the model from overfitting by forcing it to rely on different neuron combinations during training, making it more robust.

Additionally, using 100% of the data further minimized the performance gap between the training and test sets, aligning with trends observed in previous shallower models, where more data improved generalization. The deep model required stronger regularization to balance its complexity and the available data. The dropout regularization technique was effective in improving performance on the deep model, particularly with 50% of the training data. This result is consistent with the findings from experiments with smaller models, where regularization techniques like dropout helped boost generalization and reduced overfitting.

Source Code

All the source code for the main question and extra credit are included in a separate pdf file.