RUTGERS
UNIVERSITY

Course Name: Introduction to Deep Learning

Course Number and Section: **16:332:530:01**

Final Project : Prune a LeNet-5 convolutional NN on MNIST dataset with high compression ratio and negligible performance loss

Date Submitted: 05/07/2024

Submitted by:

Joel Joseph Kinny (jk2112) joel.kinny@rutgers.edu
Sumedh Ajay Marathe (sam792) sumedhajay.marathe@rutgers.edu

Abstract

To optimize deep learning models for efficient deployment in resource-constrained environments, model pruning emerges as a crucial technique. This project investigates the efficacy of various pruning methods applied to the LeNet-5 convolutional neural network, traditionally used for the classification of MNIST dataset images. We focus on two primary pruning strategies: L1 Unstructured Pruning, which targets the smallest weights globally across the network, and Random Pruning, which indiscriminately zeros weights throughout the architecture. Our experiment was structured to first train the standard LeNet-5 model to a high baseline accuracy and subsequently apply these pruning methods iteratively. The objectives were to achieve significant model compression while maintaining performance close to the unpruned baseline. Our results indicate that L1 Unstructured Pruning significantly reduces the number of parameters, achieving a substantial compression ratio with only a slight decrease in accuracy. Random Pruning, while also effective in reducing parameter count, leads to a slightly greater performance reduction. These outcomes demonstrate the potential of unstructured pruning techniques in maintaining high model efficiency without substantial loss in accuracy. This study contributes to the ongoing advancements in model optimization, suggesting that selective pruning can substantially decrease computational demands without critically compromising performance.

Introduction

In the world of deep learning, convolutional neural networks (CNNs) have established themselves as a pivotal technology in numerous applications, ranging from image and speech recognition to autonomous systems (LeCun et al., 1998; Krizhevsky et al., 2012). Despite their success, the deployment of CNNs in resource-constrained environments such as mobile devices, embedded systems, and real-time applications poses significant challenges due to their computational and storage requirements (Howard et al., 2017). As the demand for deploying deep learning models on edge devices grows, the need for efficient neural network architectures becomes imperative.

Among the various strategies for neural network optimization, model pruning stands out as a particularly effective method to reduce model complexity and resource consumption without severely impacting performance. Pruning techniques aim to eliminate redundant, non-informative weights and neurons, effectively streamlining the network for faster inference and lower power consumption (Han et al., 2015).

This project focuses on the LeNet-5, a foundational CNN architecture historically significant for handwritten digit recognition tasks using the MNIST dataset (LeCun et al., 1998). While the model's relatively modest size compared to modern architectures makes it an ideal candidate for deployment in constrained environments, further efficiency can be achieved through pruning. We explore two distinct pruning methodologies: L1 Unstructured Pruning and Random Pruning. L1 Unstructured Pruning systematically removes weights with the smallest absolute value, presumed to have the least impact on the output, while Random Pruning eliminates weights indiscriminately, serving as a baseline to evaluate the efficacy of more structured approaches (Liu et al., 2018).

The objectives of this project are twofold: firstly, to implement and assess the impact of these pruning techniques on the LeNet-5 model in terms of performance metrics such as accuracy and inference time; and secondly, to achieve significant compression of the model to facilitate its deployment in environments where

resources are limited. Through iterative application of these pruning methods, we aim to identify the optimal balance between model size reduction and performance retention, providing insights that could guide the deployment of lightweight neural networks in practical applications.

This study contributes to the broader field of model optimization by offering a comparative analysis of pruning strategies, shedding light on their potential to enhance computational efficiency in neural networks. By advancing our understanding of how different pruning techniques affect model performance and efficiency, we can better tailor CNNs for the increasingly diverse demands of real-world applications.

Related Work

The optimization of convolutional neural networks (CNNs) through pruning has become a focal area of research, especially for enhancing the deployment of deep learning models in resource-constrained environments. This section outlines some of the pivotal contributions and various approaches in the field of neural network pruning, highlighting methodologies that relate closely to the work presented in this project.

One innovative approach to pruning is described by Ganjdanesh et al. (2024), who introduced a channel pruning method that employs a reinforcement learning (RL) agent to optimize the pruning process. Their method uniquely integrates the training of model weights with the pruning process, enabling dynamic adjustment based on the performance impact, which aligns with the objectives of reducing computational demands while maintaining model accuracy (Ganjdanesh et al., 2024).

Parallelly, the concept of Evolutionary Pruning (EAPruning) discussed by Liu et al. (2024) addresses the limitations of traditional pruning methods by incorporating evolutionary algorithms to optimize both CNNs and Vision Transformers. This method focuses on a coarse pruning granularity that significantly reduces the

computational overhead associated with fine-grained pruning, providing a scalable solution to model compression (Liu et al., 2024).

Further, the Bayesian Optimization (BO) techniques applied for auto pruning of CNNs offer a systematic approach to pruning by leveraging a probabilistic model to predict the efficacy of different pruning policies. This method enhances the efficiency of the pruning process by using a Gaussian Process model to estimate the impact of potential pruning strategies before their actual implementation (Smith et al., 2024).

Additionally, a comprehensive survey by He et al. (2024) provides an extensive taxonomy of pruning techniques, detailing the various strategies like one-shot pruning, iterative pruning, and the lottery ticket hypothesis. Their work emphasizes the importance of understanding the underlying mechanisms and effects of different pruning approaches, which is crucial for applying these techniques effectively in practical scenarios (He et al., 2024).

These studies collectively underscore the significant advancements in neural network pruning, illustrating a trend towards developing more efficient, adaptable, and less resource-intensive models. The methodologies explored highlight the diversity in pruning techniques, each catering to different aspects of network optimization such as compression ratio, inference speed, and minimal impact on performance. This project builds upon these foundational ideas, aiming to further refine the pruning techniques to suit specific deployment needs in resource-constrained environments.

By integrating insights from these related works, this project not only contributes to the existing body of knowledge but also pushes the boundaries of what can be achieved with model pruning in terms of efficiency and performance retention.

Data Description

In our project, we employ the MNIST dataset, a standard benchmark for deep learning models, specifically designed for the task of handwritten digit recognition. This dataset includes a training set of 60,000 images and a test set of 10,000 images. Originally, these images were 28x28 pixels in size, but for our project, we resized them to 32x32 pixels to fit the input layer specifications of our LeNet-5 model architecture. The images are in grayscale and are categorized into ten-digit classes ranging from 0 to 9.

To optimize the training process and enhance model performance, we apply normalization to each image in the MNIST dataset. This involves adjusting the images using mean and standard deviation values calculated from the dataset, which helps in scaling the input attributes to a standard range. This normalization is crucial as it accelerates the training process and aids the model in learning more effectively.

In managing the dataset, our approach involves the use of a sophisticated data loader setup. The train loader, integral to our training regimen, not only shuffles the dataset to prevent any sequential learning biases but also implements mini-batch gradient descent to optimize the learning process efficiently. For model evaluation, the test loader plays a critical role by assessing the model's performance on unseen data, thus providing insights into the generalization ability of our neural network outside the training dataset.

Our evaluation strategy focuses on two key metrics: accuracy and loss. Accuracy is the primary metric for evaluating how well our model performs on the test set, particularly in correctly classifying the images as one of the ten-digit classes. Loss is monitored throughout the training and testing phases to ensure that our model is genuinely learning from the training data and not simply memorizing it. This dual metric approach allows us to assess the effectiveness of our model comprehensively, ensuring that it not only achieves high accuracy but also maintains robustness during practical applications.

Method Description

In this project, the training and evaluation of the LeNet-5 model incorporate two main pruning techniques alongside standard training parameters and data processing methods. The first pruning approach, L1 Unstructured Pruning, methodically removes weights based on the smallest absolute values in each layer, ensuring that only the least influential weights are discarded. This systematic method is designed to preserve the integrity of the most critical features within the data. In contrast, Random Pruning employs a less discriminative approach by randomly selecting weights across the network to be set to zero. This method can potentially lead to beneficial reductions in model complexity, though it risks removing weights that are crucial for the network's performance.

The model undergoes training over 10 epochs with a batch size of 128, employing a learning rate of 0.01 and a momentum of 0.9. These training parameters are chosen to optimize the learning process using stochastic gradient descent (SGD) with momentum. This type of optimization not only accelerates the convergence by directing gradient vectors more effectively but also helps in smoothing out the updates to avoid getting trapped in local minima. The use of a negative log-likelihood loss function is particularly appropriate for classification tasks, as it deals effectively with scenarios involving mutually exclusive classes by penalizing incorrect classifications.

For data processing, the images are resized to 32x32 pixels, converted into tensors, and normalized using specific mean and standard deviation values to standardize input data, enhancing the model's ability to generalize from the training data to unseen data.

To assess the impact of the pruning techniques and training regimen, the project includes comprehensive visualization and analysis components. It tracks training and test losses and accuracies, graphically presenting these metrics to monitor the model's performance across the training epochs. Additionally, a compression analysis is conducted to compare the model's size and the number of parameters before and after pruning, evaluating the effectiveness of the compression.

Performance metrics, specifically accuracy and loss on the test dataset, are scrutinized to determine the overall impact of pruning on the model's performance. This methodological approach not only clarifies the effects of weight removal strategies on network efficiency but also provides insights into the trade-offs between model simplicity and performance accuracy.

Model Description

The LeNet-5 architecture, a type of convolutional neural network (CNN), is specifically designed for image recognition tasks, such as classifying MNIST dataset images. The model structure begins with an input layer that accepts pre-processed 32x32 pixel images, an adjustment from the original MNIST images which are 28x28 pixels. This resizing is necessary to match the architecture's specifications.

The first convolutional layer, known as C1, contains six filters, each with a kernel size of 5x5. These filters operate with a stride of 1 and no padding, reducing the image dimension to 28x28. This layer is followed by the first subsampling layer, S2, which performs 2x2 max pooling with a stride of 2, effectively halving the feature map size to 14x14. The network's second convolutional layer, C3, increases the complexity with 16 filters of the same 5x5 size, again followed by a second subsampling layer, S4, which further reduces the feature map size through another round of 2x2 max pooling.

The third convolutional layer, C5, significantly expands the network's capacity with 120 filters, preparing the data for high-level feature extraction before transitioning to the fully connected layers. The first of these layers, F6, contains 84 neurons and plays a critical role in learning and synthesizing the features extracted from the previous layers. The architecture culminates in the output layer, F7, which consists of 10 neurons, each corresponding to one of the ten-digit classes (0-9). This layer employs a softmax activation function to generate a probability distribution across the digit classes, representing the network's final output.

Throughout the model, ReLU activation functions are utilized in both convolutional and fully connected layers to introduce non-linearity, enhancing the network's ability to learn complex patterns in the data. The LeNet-5's design, characterized by its sequence of convolutional and subsampling layers followed by fully connected layers, is optimized to capture essential features at various scales, making it highly effective for digit recognition tasks.

Experimental Procedure and Results

Experimental Setup

The experiments were conducted on an AMD Ryzen 5 CPU and NVIDIA 3050 RTX GPU 4 GB system. The base model of LeNet is trained to establish a high accuracy of 99% validation accuracy and the training is done in 161 seconds for 10 epochs. Once the base model is established the two methods of pruning are employed (a) Random Pruning (b) L1 Norm Pruning on the model using a global unstructured scheme where each layer is pruned by a certain amount to sum up to the specified amount.
The base trained model is applied with different sparsity ratios which vary from 0.1 to 0.99 for both of the methods and the model is then fine-tuned and retrained to get the new training losses. The newly trained model is now trained after the parameters have been pruned and we can observe the different compression ratios achieved using the two methods of pruning.

L1 Unstructured Pruning

In Fig. 1, we can see the best training loss curve for a compression ratio of 0.99 using the L1 norm method. In the figure, we can see that the accuracy was not hampered by applying the pruning and the training loss was minimized by the 2nd epoch. (not seen in the figure). The training curves were observed for the different sparsity ratios and this method was able to achieve a good training and validation accuracy for all cases as seen in Fig. 2. It is worth noting that the pruning of layers

reduces the size of the network by the value achieved and each of the layers have been reduced by a certain percentage based on the L1 norm. Table 1 shows the layer wise sparsity introduced by pruning the layers on a global scale. We also observed that the pruning method is carried the most for layer c5 which is a convolutional layer and layer f6 which is the fully connected layer. These two layers hold many parameters which can be pruned whereas the layers c1 and f7 are not pruned as it is the first input and last layer to the output respectively.

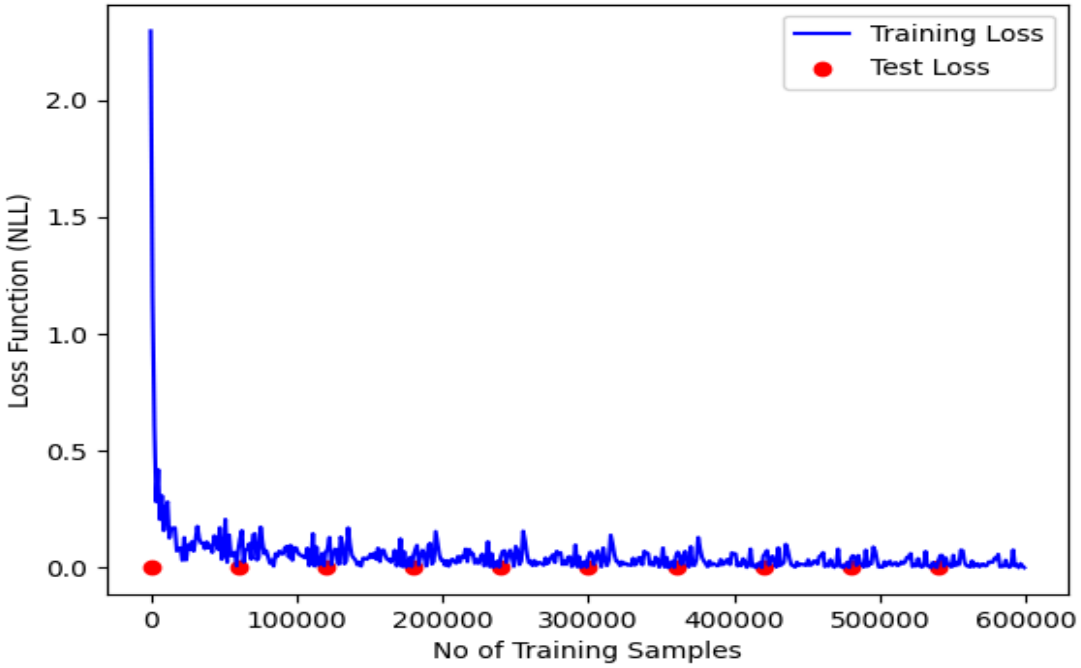

Fig. 1 Training curve for L1 Unstructured pruning

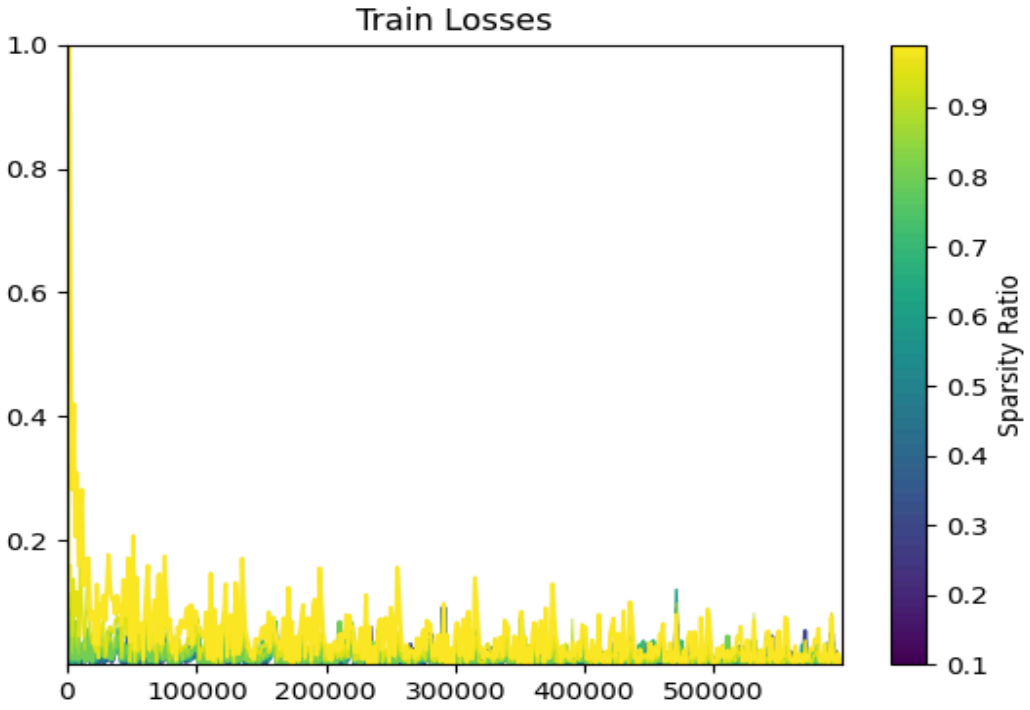

Fig. 2 Training loss curves for range of sparsity ratio

Layer	Sparsity %									
c1	35.33	24.67	16.67	11.33	7.33	6.67	6.67	6	4.67	3.33
c3	92.67	77.21	65.79	50.83	43	35.88	29.38	22.62	17.17	11.29
c5	99.94	98.46	95.68	87.52	77.37	66.59	55.51	44.6	33.48	22.41
f6	99.07	87.52	74.01	56.29	45.96	38.35	32.04	24.99	18.54	11.97
f7	73.69	50.48	39.64	30.24	25.6	21.55	17.38	12.98	9.76	6.67
Total	99	95	90	80	70	60	50	40	30	20

Table 1: Sparsity Percentage for Each Layer with L1 Unstructured Pruning

In Table 2, we can see the compression ratio achieved was the highest for sparsity ratio of 0.99 and retained the validation accuracy as seen in the base model. The compression ratio was minimal for values below 0.8 and it is almost negligible. The training time as captured shows that the pruning method saves space by 51% but increases the training time by 14.9% as compared to the base model. The reduced training time and speedup of operations while training is only seen when

the sparsity ratio is below 0.95. The tradeoff is seen due to the overhead of pruning.

Sparsity Ratio	Compression (%)	Validation Accuracy (%)	Train Time
0.99	51.3045	99	00:03:05
0.95	19.2866	99	00:03:14
0.9	9.0769	99	00:02:27
0.8	1.486	99	00:02:28
0.7	0.8297	99	00:02:29
0.6	0.0032	99	00:02:29
0.5	0.0081	99	00:02:27
0.4	0.0032	99	00:02:31
0.3	0	99	00:02:28
0.2	0.0016	98	00:02:29

Table 2: L1 Unstructured Pruning Metrics

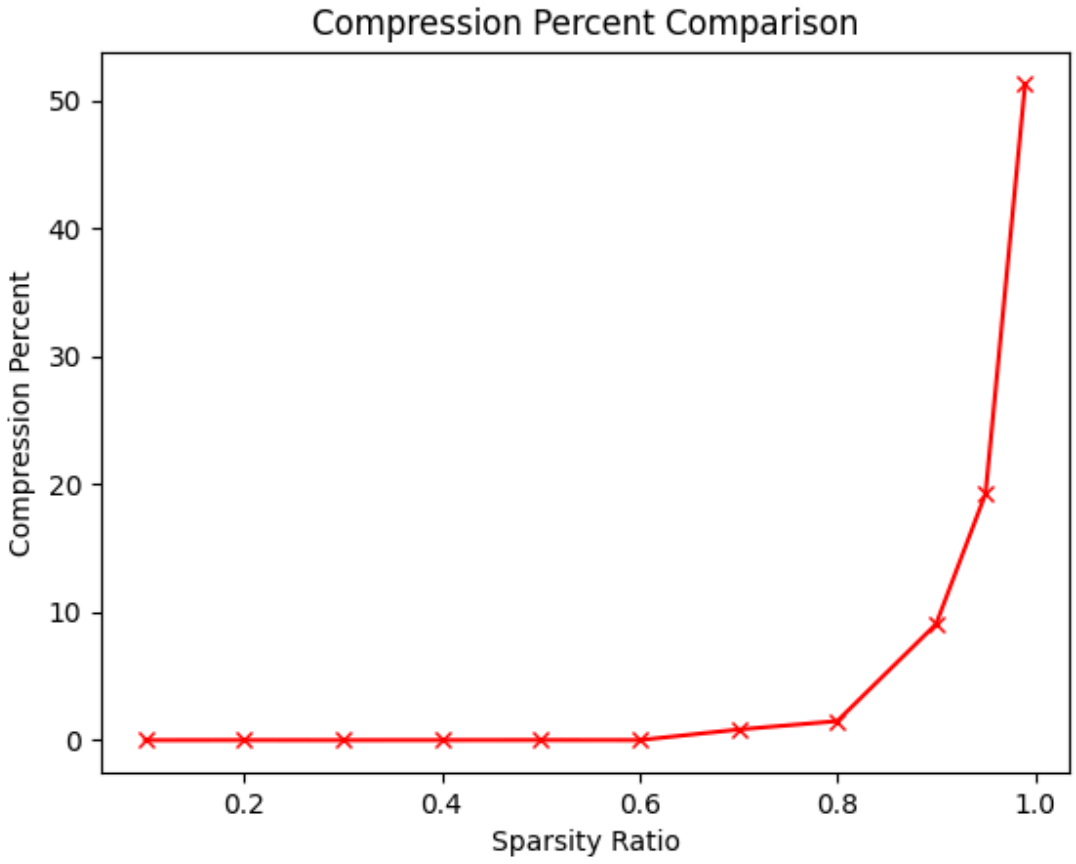

Fig. 3 Compression comparison of L1 Unstructured Pruning on LeNet5

Random Pruning

In Fig. 1, we can see the best training loss curve for a compression ratio of 0.95 using the random pruning method. In the figure, we can see that the accuracy was not hampered by applying the pruning and the training loss was minimized by the 4th epoch. (not seen in the figure). The training curves were observed for the different sparsity ratios and this method was able to conserve the accuracy for all cases but when the ratio is greater than 0.95. In Fig. 2, we can see that the accuracy drops down and is almost devoid of accuracy. Even though the final validation loss is preserved, it takes longer for the loss to be minimized. Table 1 shows the layer wise sparsity introduced by pruning the layers on a global scale. This method randomly prunes weights in every layer based on the percentage passed and applies it globally to all layers. However, using this method does not achieve good accuracy and the maximum accuracy is achieved with a ratio of 0.95.

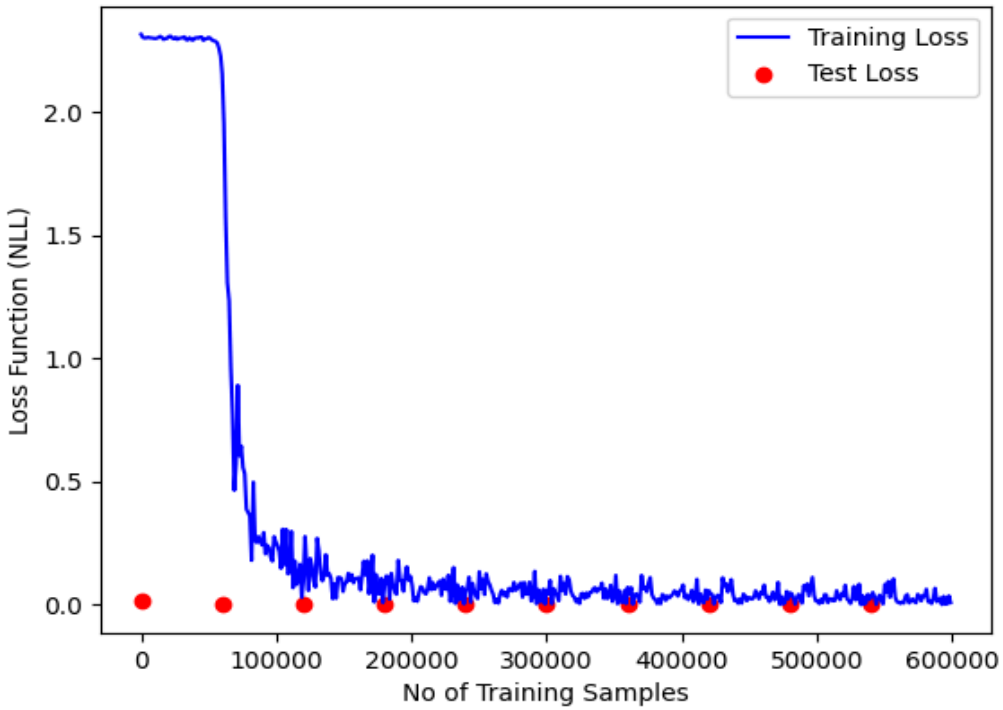

Fig. 4 Training curve for Random pruning

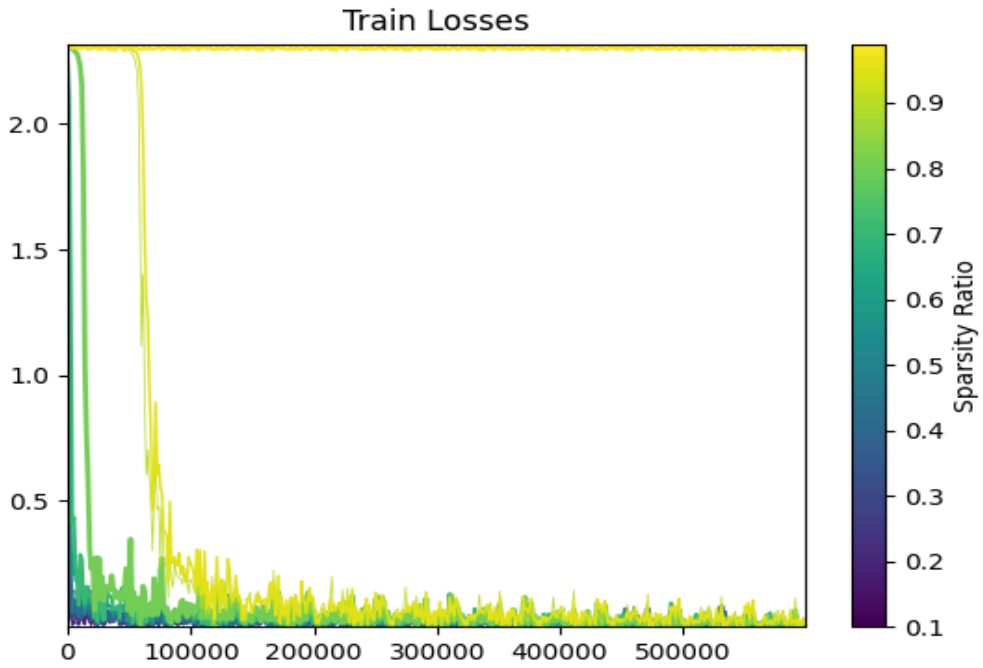

Fig. 5 Training loss curves for range of sparsity ratio

Layer	Sparsity %									
c1	99.33	96	96	80	64	62	46.67	44.67	30	17.33
c3	99.12	94.92	90.04	79.54	72.17	60.46	49.46	38.62	29.17	19.71
c5	98.99	94.9	89.85	80	70.05	60.06	50.01	39.94	30.14	19.83
f6	99.03	95.42	90.6	80.02	69.42	59.77	50.23	40.36	29.5	20.71
f7	99.05	95.71	90.12	81.19	69.05	57.62	48.93	42.14	30.6	22.38
Total	99	95	90	80	70	60	50	40	30	20

Table 3: Sparsity Percentage for Each Layer with Random Pruning

In Table 4, we observe the compression ratio of 76% being achieved but with a complete hit on accuracy of the model. A sparsity ratio of 0.9 can be used to achieve a compression of 10% and still preserve the accuracy of the model. This is done with a speedup of almost 6.3% as compared to the base model. The other compression ratios are minimal and not worthy of consideration for this study.

Sparsity Ratio	Compression (%)	Validation Accuracy (%)	Train Time
0.99	76.4544	11	00:03:05
0.95	1.1441	99	00:03:06
0.9	10.2988	99	00:02:30
0.8	0.0016	99	00:02:29
0.7	0	99	00:02:28
0.6	0.0291	99	00:02:32
0.5	0.0032	99	00:02:28
0.4	0	99	00:02:27
0.3	0.02106	98	00:02:30
0.2	0	99	00:02:28

Table 4: Random Pruning Metrics

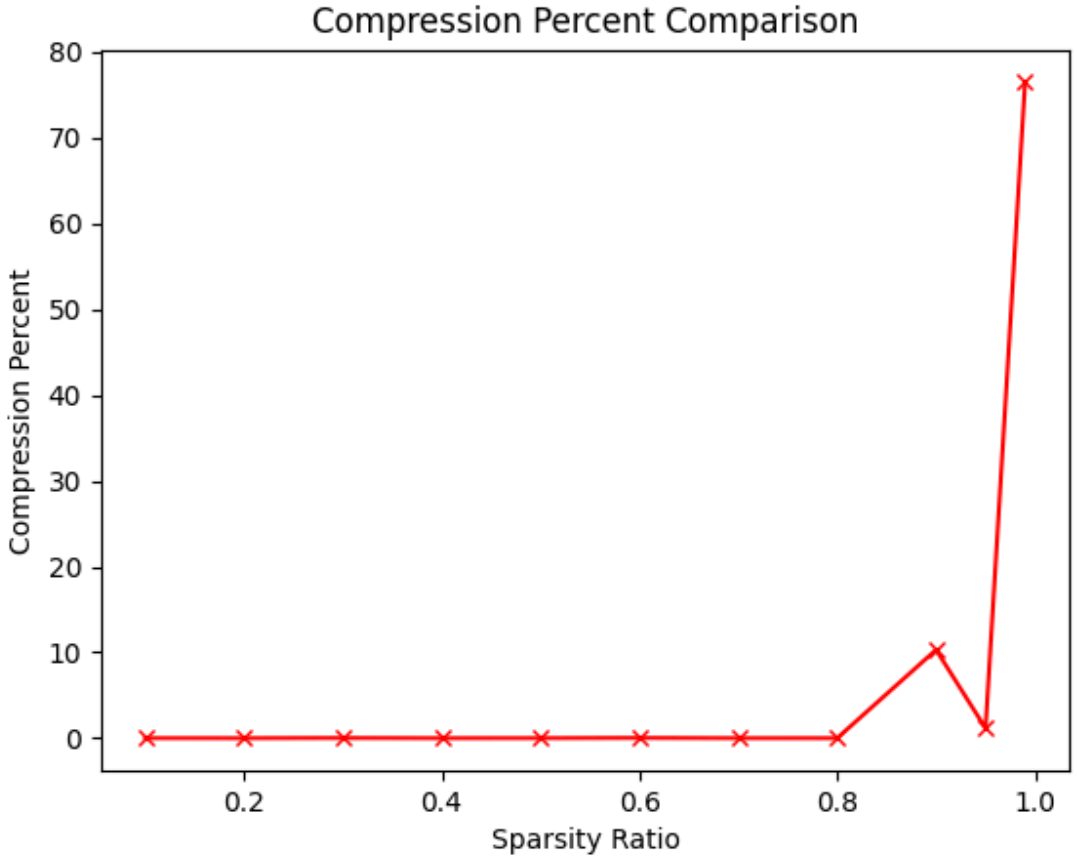

Fig. 6 Compression comparison of Random Pruning on LeNet5

Conclusion

The experimental results of the two above methods clearly show that L1 unstructured pruning achieves a better compression ratio as compared to the random pruning without affecting the accuracy of the model. This pruned model takes less time to train as compared to the base model of LeNet5. In this study, we have successfully pruned the LeNet5 model using two state-of-the-art methods. The over-parameterization of deep learning models has always been an issue and compression is important when there is a need for implementation of such models on resource constrained devices or platforms. This study can be further extended by implementing better pruning techniques and employing post-quantization methods to further compress the model which we reserve as future work.

References

1. S. Han, J. Pool, J. Tran, and W. Dally, "Learning both weights and connections for efficient neural networks," in Advances in Neural Information Processing Systems, pp. 1135-1143, 2015.
2. A. Howard et al., "MobileNets: Efficient convolutional neural networks for mobile vision applications," arXiv preprint arXiv:1704.04861, 2017.
3. A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097-1105, 2012.
4. Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278-2324, 1998.
5. Z. Liu et al., "Learning efficient convolutional networks through network slimming," in Proceedings of the IEEE International Conference on Computer Vision, pp. 2736-2744, 2018.
6. A. Ganjdanesh, S. Gao, and H. Huang, "Jointly Training and Pruning CNNs via Learnable Agent Guidance and Alignment," 2024. [Online]. Available: https://arxiv.org/abs/2403.19490
7. Q. Li, B. Zhang, and X. Chu, "EAPruning: Evolutionary Pruning for Vision Transformers and CNNs," 2022. [Online]. Available: https://arxiv.org/abs/2210.00181
8. H. Fan, J. Mu, and W. Zhang, "Bayesian Optimization with Clustering and Rollback for CNN Auto Pruning," 2022. [Online]. Available: https://arxiv.org/abs/2109.10591
9. Y. He et al., "A Survey on Deep Neural Network Pruning: Taxonomy, Comparison, Analysis, and Recommendations," Journal Name, vol. volume(issue), pages, 2024.
10. S. Vadera and S. Ameen, "Methods for Pruning Deep Neural Networks," in IEEE Access, vol. 10, pp. 63280-63300, 2022, doi: 10.1109/ACCESS.2022.3182659.
11. H. Tanaka, D. Kunin, D. L. K. Yamins, and S. Ganguli, "Pruning neural networks without any data by iteratively conserving synaptic flow," CoRR, vol. abs/2006.05467, 2020. [Online]. Available: https://arxiv.org/abs/2006.05467
12. PyTorch. (n.d.). Pruning Tutorial. [Online]. Available: https://pytorch.org/tutorials/intermediate/pruning_tutorial.html