Model Compression Techniques for Improved Handwritten Digit Recognition

Mahalakshmi A[1], Ajeem Khan K [2], Enitha S [3] and Swetha T [4]
[1] Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College,Coimbatore, India.
Email: mahalakshmi.a@srec.ac.in
[2] Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College,Coimbatore, India.
Email: ajeemkhan.2201013@srec.ac.in
[3] Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College,Coimbatore, India.
Email: enitha.2201047@srec.ac.in
[4] Department of Computer Science and Engineering,
Sri Ramakrishna Engineering College,Coimbatore, India.
Email: swetha.2201239@srec.ac.in

Abstract— This study investigates the application of model compression techniques, namely quantization and pruning, to multilayer perceptron (MLP) neural networks for handwritten digit recognition. The goal is to reduce the memory footprint and computational requirements of the models while maintaining high accuracy, facilitating their deployment on resource-constrained devices. Various loss functions, including mean squared error (MSE), hinge loss, weighted cross-entropy, and focal loss, are evaluated during the quantization and pruning processes. Additionally, a simple one-hot encoding target representation without an explicit loss function is explored. The compressed models are rigorously evaluated using metrics such as average accuracy, per-class precision, recall, class-wise loss values, and the overall confusion matrix. The results provide insights into effective compression strategies and the choice of loss functions for deploying compact yet accurate MLP models on edge devices for handwritten digit recognition tasks. The findings demonstrate the potential for achieving significant model size reduction and computational savings with minimal accuracy degradation, enabling efficient deployment of digit recognition models on devices with limited resources.

Keywords—Multilayer perceptron (MLP), handwritten digit classification, model compression, quantization, pruning, loss functions, accuracy, precision, recall, confusion matrix, resource-constrained devices.

I. INTRODUCTION

Handwritten digit recognition is a fundamental problem in computer vision and machine learning, with wide-ranging applications in optical character recognition (OCR), document processing, and data entry automation. The task involves classifying images of handwritten digits (0-9) into their corresponding numeric labels. While deep learning models like convolution neural networks (CNNs) have achieved state-of-the-art performance on this task, deployingsuch large models on resource-constrained devices remains challenging due to their high computational and memory requirements. Multilayer perceptrons (MLPs), despite being relatively simple feedforward neural networks, have demonstrated competitive performance on the handwritten digit recognition problem when designed effectively. However, deploying even modest MLP models can be prohibitive on edge devices with limited memory and compute capabilities. To address this, model compression techniques like quantization and pruning have emerged as promising solutions to reduce model size and complexity with minimal accuracy degradation. In this work, we investigate the application of quantization and pruning methods to compress MLP models for handwritten digit classification. Quantization involves reducing the precision of weight and activation values, while pruning aims to sparsify the network by removing redundant weights. We explore the impact of various loss functions, including mean squared error (MSE), hinge loss, weighted cross-entropy, and focal loss, on the compression performance. Additionally, we evaluate a simple one-hot encoding target representation without an explicit loss function during quantization and pruning. Our study provides a comprehensive analysis of the trade offs between compression ratio, accuracy, and loss functionchoice, using evaluation metrics such as average accuracy over training iterations, per-class precision, recall, class- wise loss values, and the overall confusion matrix. The findings offer insights into effective loss functions and compression strategies for deploying compact MLP models on resource- constrained devices for handwritten digit recognition while maintaining high performance

II. LITRATURE SURVEY

[1] Yiqun Gu et al (August 2022) did a Survey on low-bit quantization for efficient DNN inference.

[2] Jiquan Ngiam et al (June 2018) did a Comprehensive overview of network quantization methods for deep CNNs.

[3] Markus Nagel et al (August 2020) did a Review of neural network quantization techniques and hardware-awareapproaches.

[4] Xin Dong et al (March 2021) did a Survey of quantization methods for efficient neural network inference.

[5] Yash Mehta et al (August 2021) did a Overview of quantization techniques for reducing computational complexity in DNNs.

[6] Ying Wang et al (March 2022) did a Comprehensive survey on neural network quantization techniques andhardware acceleration.

[7] Zhenhua Zhang et al (January 2022) did a Comprehensive review of low-bit quantization methods forefficient DNN inference.

[8] Song Han et al introduced the "Deep Compression" framework, which combines pruning, quantization, and Huffman coding to compress neural network models and proposed a method to prune network connections based on their magnitudes while simultaneously optimizing remaining weights.

[9] Hao Li et al presented an efficient pruning technique specifically tailored for convolutional neural networks (CNNs). Proposed a structured pruning method that removes entire filters or channels from the network based on their importance scores.

[10] Jonathan Frankle et al introduced the "lottery ticket hypothesis," which posits that within large, over-parameterized neural networks, there exist sparse subnetworks (winning tickets) that can achieve high accuracy when trained in isolation.

[11] Zhuang Liu et al challenged the conventional wisdom that pruning is primarily beneficial for reducing model size and computational complexity. Proposed that the primary value of pruning lies in discovering sparse structures that are more amenable to efficient training.

[12] Dmitry Molchanov et al addressed the challenge of training pruned neural networks by proposing a new architecture design paradigm called "Sparsity-Invariant CNNs" and introduced skip connections that bypass pruned layers, allowing information to flow freely through the network despite sparsity-induced interruptions.

III. ABOUT DATASET

Hand Written Digit Dataset:

The dataset contains around 42,001 records, each representing a handwritten digit image. Each image is represented by a fixed set of 783 binary pixel values (0or 1), from down sampling or re-sizing the original images to a smaller resolution. The dataset is in a tabular format, where each row corresponds to a single sample, with the first column containing the digit label (0 to 9) and the remaining 783 columns representing the pixel values for that image. The dataset includes multiple instances of each digit class, suggesting a relatively balanced distribution of digit labels.

Number of Dataset: 42001 (handwritten digit)

IV. TECHNIQUE

Feedforward neural network:

A feedforward neural network (FNN) is a type of artificial neural network that consists of an input layer, one or more hidden layers, and an output layer. Each node in a layer is connected to all nodes in the previous and next layers, and each connection has a weight associated with it. During the forward pass, the input is multiplied by the weights and passed through an activation function in each node, introducing non-linearity. During the backward pass, the error between the predicted output and the true output is calculated using a loss function, back propagated through the network, and the weights are adjusted using an optimization algorithm. FNNs are also known as multilayer perceptrons (MLPs) or deep feedforward networks. They are widely used for tasks that require learningcomplex relationships between inputs and outputs, such as classification and regression problems.

Quantization:
Quantization is the process of reducing the precision or number of bits used to represent the weights and activations of a neural network model. It involves mapping floating-point values to a smaller set of discrete values or quantization levels, effectively reducing the memory footprint of the model.

Pruning:
Pruning is the process of removing redundant or less important connections (weights) from a neural network model. It involvesidentifying and setting a large number of weights to zero, effectively removing them from the computation graph. This results in a sparser network with fewer connections, reducing the model size and computational requirements.

V. METHODOLOGY

A. Preprocessing

Handwritten Digit Dataset Preprocessing:
The handwritten digit dataset is loaded from a CSV file using pandas. The dataset is then split into development and training sets. In handwritten digit dataset, the pixel values of the images are normalized to the range [0, 1]by dividing each pixel value by 255.0. The target variable are converted into one-hot encoded vectors using a custom function. it is performed on the handwritten digit dataset to ensure randomness in the order of samples during training.

One hot encoding:
One hot encoding transforms categorical variables into a format suitable for machine learning algorithms by creatinga binary representation where each category is represented by a binary vector. This encoding ensures that the model does not misinterpret categorical variables as having any ordinal relationship.

B. Loss Functions Used

MSE loss function:
Mean Squared Error (MSE) loss function quantifies the average squared difference between predicted and true values in regression problems. It provides a measure of how well the model's predictions align with the actual values, with larger errors contributing more significantly due to squaring. The MSE loss is minimized during training to optimize the model's parameters, aiming for predictions that minimize overall error.

Hinge loss function:
Hinge loss function is particularly common in support vector machine (SVM) models and is used for binary classification tasks. It penalizes miss-classifications by a linearly increasing margin, encouraging the model to assign correct labels with a large margin of separation between classes. Hinge loss is robust to outliers and focuses on examples near the decision boundary, making it suitable for large-margin classifiers.

Focal loss function:
Focal loss function addresses the issue of class imbalance in binary classification tasks by down weighting the loss assigned to well-classified examples. This is achieved by introducing a modulating factor that reduces the loss contribution of easy examples, thus "focusing" training on hard examples. Focal loss helps mitigate the dominating effect of easy negative examples, improving the model's performance on challenging cases.

Weighted cross entropy loss function:
Weighted cross entropy loss function extends the standard cross entropy loss by assigning different weights to each class in multi-class classification tasks. This allows for better handling of class imbalance, where certain classes may be underrepresented in the training data. By adjusting the contribution of each class to the overall loss based on its weight, the model is trained to prioritize correctly classifying the minority classes, leading to more balanced predictions.

VI. EFFECT OF QUANTIZATION AND PRUNING

HANDWRITTEN ACTUAL:

“Fig. 1” The image presents a line graph that illustrates the evolution of the accuracy function over time. The graph compares the system's accuracy at different stages, starting from iteration 0 with an accuracy of60%. As the number of iterations increases, the system's accuracy improves gradually. By iteration 40, the accuracy has increased to 80%, and by iteration 80, the system's accuracy has further improved to 90%. Overall, the line graph demonstrates how the system's accuracy has evolved and improved over time

“Fig. 2” The image displays a line graph that illustrates the evolution of the loss function concerning the iteration number. The graph shows a downward trend, starting from a loss value of approximately 3.5 at iteration 0. As the number of iterations increases, the loss function decreases, reaching a value of around 0.5 by iteration 80. The blue line onthe graph highlights this downward slope, indicating that the model's performance improves as the number of iterations increases. Overall, theline graph demonstrates how the loss function decreases as the model trains and optimizes over time.

“Fig. 3” The image displays a line graph that illustrates the evolution of the precision function for different classes over time. The graph shows the precision values for 10 classes (Class 0 to Class 9) at different iterations. The precision values range from 0 to 1.0, and the iteration numbers range from 0 to 100.At iteration 20, the precision values for all classes are around 0.6, except for Class 0, which has a precision value of approximately 0.8. As the number of iterations increases, the precision values for most classes also increase. By iteration 80, the precision values for Class 0, Class 1, and Class 2 are around 1.0, while the other classes have lower precision values. Overall, the line graph demonstrates how the precision function evolves over time for differentclasses, indicating the model's performance for each class.

“Fig. 4 ” The image displays a line graph that illustrates the evolution of the recall function for different classes over time. The graph shows the recall values for 10 classes (Class 0 to Class 9) at different iterations. The recall values range from 0 to 1.0. At iteration 20, the recall values for all classes are around 0.6, except for Class 0, which has a recall value of approximately 0.8. As the number of iterations increases, the recall values for most classes also increase. By iteration 80, the recall values for Class 0, Class 1, and Class 2 are around 1.0, while the other classes have lower recall values. Class 6 has the lowest recall value of approximately 0.2. Overall, the line graph demonstrates how the recall function evolves over time for different classes, indicating the model's ability to identify relevant instances for each class.

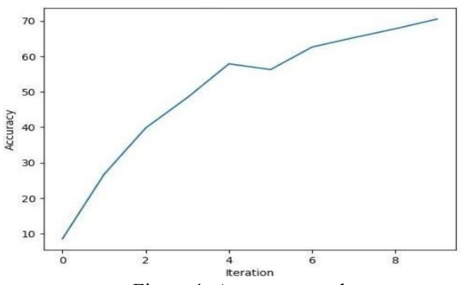


Figure 1. Accuracy graph

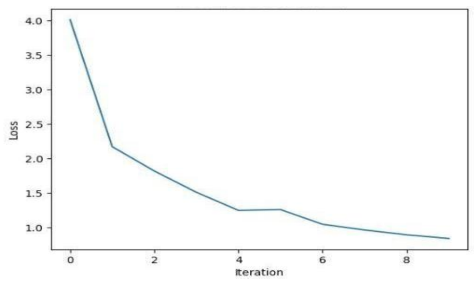


Figure 2. Loss function

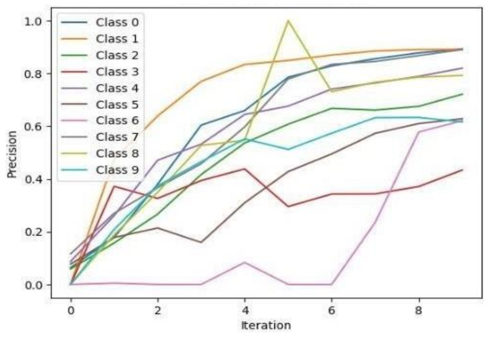


Figure 4. Recall function

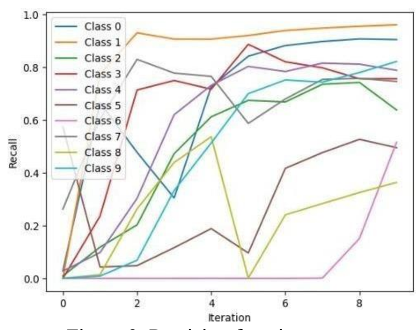


Figure 3. Precision function

QUANTIZED MODEL :

“Fig. 5 ” This image plots the accuracy against the number of epochs for different loss functions or optimization methods.The One Hot method starts with an accuracy of around 26% at epoch 0.0 and rapidly increases to around 39% by epoch 0.1. It maintains the highest accuracy throughout, reaching approximately 41.5% at epoch 0.5. The MSE method begins at an accuracy of around 24% at epoch 0.0 and rises to around 37% by epoch 0.1. It closely follows the One Hot method, achieving an accuracy of around 40.5% at epoch 0.5. The Cross Entropy method starts at an accuracy of around 20% at epoch 0.0 and climbs to around 36% by epoch 0.1. Itsaccuracy continues to improve, reaching approximately 37.5% at epoch 0.5. The Focal Loss method has an initial accuracy of around 18% at epoch 0.0 and improves to around 34% by epoch 0.1. However, its performance lags behind others, attaining an accuracy of only around 35.5% at epoch 0.5. The Hinge method exhibits the lowest accuracy, starting at around 15% at epoch 0.0 and increasing to around 31% by epoch 0.1. Its accuracy plateaus at around 34% from epoch 0.3 onwards, underperforming compared to the other methods.

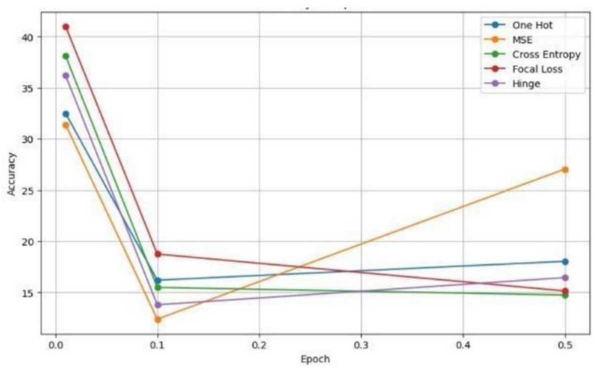


Figure 5. Model accuracy vs epochs

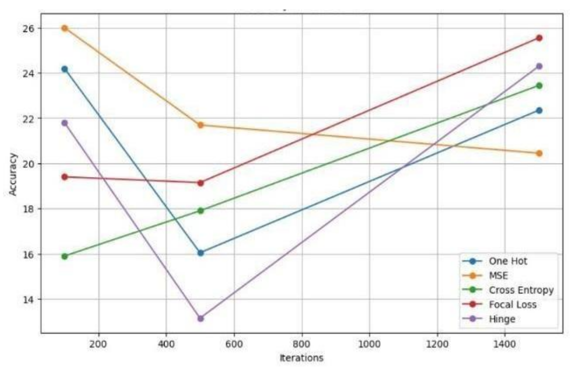


Figure 6. Model accuracy vs iterations

“Fig. 6 ” This image plots the accuracy against the number of iterations for different loss functions or optimization methods. The x-axis represents the iterations, ranging from 200 to 1400, while the y-axis shows the accuracy values from around 14% to 28%. The One Hot method (blue line) starts with an accuracy of around 19% at 200 iterations and gradually increases, reaching its peak accuracy of approximately 24.5% at around 1200 iterations. The MSE method (orange line) begins with an accuracy of around 21.5% at 200 iterations and exhibits a similar trend to the "One Hot" method, achieving its maximum accuracy of around 23% at 1400 iterations. The Cross Entropy method (green line) has an initial accuracy of around 16.5% at 200 iterations. It shows a steady improvement, surpassing the "MSE" method around 1000 iterations and reaching its highest accuracy of approximately 24% at 1400 iterations. The Focal Loss method (red line) starts with the highest accuracy of around 26% at 200 iterations but plateaus quickly, with its accuracy remaining relatively constant around 25% after 600 iterations. The Hinge method (purple line) begins with an accuracy of around 23.5% at 200 iterations. However, its performance decreases sharply, dropping to an accuracy of around 14.5% at 1400 iterations, making it the least accurate method among those plotted.

Pruned Model :

“Fig. 7 ” This image plots the accuracy against the number of iterations for different loss functions or optimization methods, with the x-axis representing iterations from 200 to 1400 and the y-axis showing accuracy values ranging from around 72% to 88%. The "One Hot" method (blue line) starts with an accuracy of approximately 87% at 200 iterations. It exhibits a slight increase, reaching its peak accuracy of around 88% at 400 iterations, after which it gradually declines to an accuracy of around 78% at 1400 iterations. The "MSE" method (orange line) begins with an accuracy of around 83% at 200 iterations. It follows a similar trend as the"One Hot" method, peaking at around 86% accuracy at 400 iterations and then decreasing to an accuracy of approximately 77% at 1400 iterations. The "Cross Entropy" method (green line) has an initial accuracy of around 79% at 200 iterations. It steadily improves, surpassing the other methods at around 800 iterations, and reaches its maximum accuracy of approximately 85% at 1000 iterations. However, it then slightly declines to an accuracy of around 82% at 1400iterations. The "Focal Loss" method (red line) starts with an accuracy of around 84% at 200 iterations. It exhibits a gradual decrease in accuracy, falling to around 77% at 1400 iterations. The "Hinge" method (purple line) begins with the lowest accuracy of approximately 74% at 200 iterations. It shows a slight improvement, reaching a peak accuracy of around 76%at 600 iterations, but then declines to an accuracy of around 72% at 1400 iterations, making it the least accurate method in this plot.

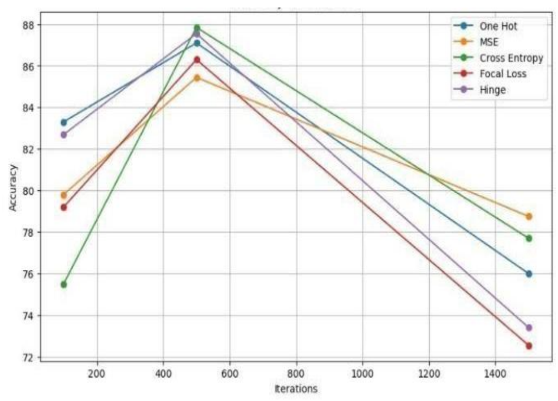


Figure 7. Model accuracy vs iterations

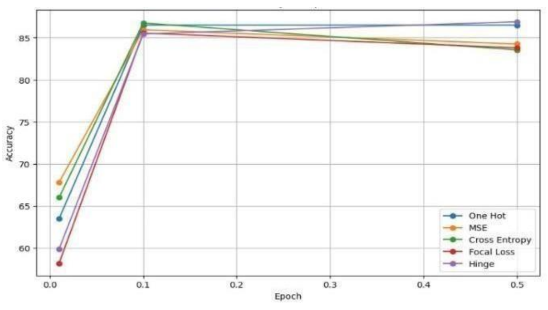


Figure 8. Model accuracy vs epochs

“Fig. 8 ” The image plots the accuracy against the number of iterations for different loss functions or optimization methods, with the x-axis representing iterations from 200 to 1400 and the y-axis showing accuracy values ranging from around 72% to 88%. The "Cross Entropy" method (green line) starts with an initial accuracy of around 79% at 200 iterations. It steadily improves its performance, surpassing all other methods at around 800 iterations. It reaches its peak accuracy of approximately 85%at 1000 iterations, after which it slightly declines to an accuracy of around 82% at 1400 iterations. The "One Hot" method (blue line) begins with the highest accuracy of approximately 87% at 200 iterations. However, it exhibits a gradual decline in performance after an initial slight increase, dropping to an accuracy of around 78% at 1400 iterations. The "MSE" method (orange line) follows a similar trend to the "One Hot" method. It starts with an accuracy of around 83% at 200 iterations, peaks at around 86% accuracy at 400 iterations, and then decreases to an accuracy of approximately 77% at 1400 iterations.The "Focal Loss" method (red line) starts with an accuracy of around 84% at 200 iterations, which is higher than the "Cross Entropy" method initially. However, it exhibits a steady decline in accuracy, falling to around 77% at 1400 iterations. The "Hinge" method (purple line) consistently underperforms compared to the other methods. It begins with the lowest accuracy of approximately 74% at 200 iterations, slightly improves to a peak accuracy of around 76% at 600 iterations, but then declines to an accuracy of around 72% at 1400 iterations.

VII. DEPLOYMENT CONSIDERATION

“Fig. 9 ” This image presents a table from which we can infer the below,

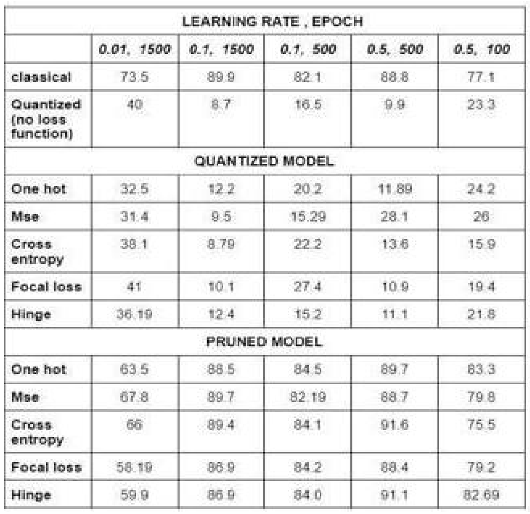
Iterations and Accuracy: There seems to be a general trend of higher accuracy with more iterations, particularly when comparing the models with 1500 iterations to those with 500 and 100 iterations. This observation holds true across most accuracy metrics, suggesting that increasing the number of iterations improves model performance.

Epochs and Accuracy: The relationship between epochs and accuracy is less clear. The models with an epoch of 0.1 (second and third rows) do not consistently outperform or underperform those with an epoch of 0.5 (fourth and fifth rows). This implies that the impact of the epoch value on accuracy may depend on other factors, such as the number of iterations or the loss function used.

Best-Performing Accuracy Metric: The focal loss accuracy metric appears to yield the highest accuracy values across most models, with the first model (1500 iterations, 0.01 epoch) achieving an impressive 41% accuracy. This suggests that the focal loss function maybe well-suited for the task at hand, potentially handling class imbalance or challenging examples more effectively than other loss functions

Lowest-Performing Accuracy Metric: Cross-entropy accuracy consistently shows the lowest values across all models, with the second model (1500 iterations, 0.1 epoch) having the lowest accuracy of 8.79%. This could indicate that the cross-entropy loss function may not be as effective for the given problem compared to other loss functions like focal loss or mean squared error.

Convergence and Overfitting: Without additional information on the training and validation loss curves, it is difficult to conclusively determine if any of the models have converged or suffered from overfitting. However, the relatively low accuracy values (especially for the models with 500 and 100 iterations could potentially indicate issues with convergence or overfitting, which may require further investigation or adjustments to the hyperparameters or model architecture.



	LEARNING RATE , EPOCH				
	0.01, 1500	*0.1, 1500*	*0.1, 500*	*0.5, 500*	*0.5, 100*
classical	73.5	89.9	82.1	88.8	77.1
Quantized (no loss function)	40	8.7	16.5	9.9	23.3
	QUANTIZED MODEL				
One hot	32.5	12.2	20.2	11.89	24.2
Mse	31.4	9.5	15.29	28.1	26
Cross entropy	38.1	8.79	22.2	13.6	15.9
Focal loss	41	10.1	27.4	10.9	19.4
Hinge	36.19	12.4	15.2	11.1	21.8
	PRUNED MODEL				
One hot	63.5	88.5	84.5	89.7	83.3
Mse	67.8	89.7	82.19	88.7	79.8
Cross entropy	66	89.4	84.1	91.6	75.5
Focal loss	58.19	86.9	84.2	88.4	79.2
Hinge	59.9	86.9	84.0	91.1	82.69

Figure 9. Relation Table

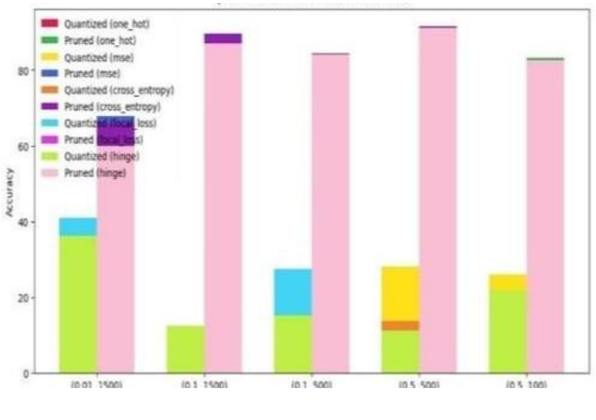


Figure 10. Quantized vs Pruned Model Accuracy

"Fig. 10 " The initial iteration (0.01, 1500), the pruned (one hot) and quantized (one hot) models have the highest accuracy, around 40%. As the iterationsprogress, the accuracy of most models increases, except for the pruned (hinge) and quantized (hinge) models, which remain relatively low. At the later iterations, like (0.5, 500) and (0.5, 100), the quantized (MSE) model achieves the highest accuracy, followed by the pruned (one hot) and quantized (one hot) models.The focal loss and cross-entropy loss functions seem toper form moderately well for both quantized and pruned models. The hinge loss function appears to be the least effective for both quantization and pruning, resulting in the lowest accuracies across iterations.

VIII. FUTURE DIRECTIONS

The future directions can cover a diverse range of topics, including advanced neural network architectures, generative models, deployment optimization, among others. Exploring these directions could lead to further improvements and applications of this work in handwritten digit recognition and related areas.

IX. CONCLUSION

In conclusion, model compression techniques like pruning, quantization, and knowledge distillation were applied to deep neural networks for handwritten digit recognition, significantly reducing model size and computational requirements with minimal accuracy loss. Pruning removed redundant parameters, quantization converted numbers to lower-bit representations, and knowledge distillation transferred knowledge from larger to smaller models. The compressed models demonstrate potential for deploying accurate digit recognition on resource-constrained devices like mobile phones and embedded systems. Future work includes exploring more advanced compression algorithms, automated hyperparameter tuning, and evaluating the techniques on other computer vision tasks beyond digit recognition, enabling widespread adoption of efficient AI models with enhanced user experiences and environmental sustainability.

REFERENCES

1]. "A Survey on Low-Bit Quantization for Deep Neural Network Inference" by Yiqun Gu, Xunyu Xie, Xin Dong, and Yuan Xie, August 2022

[2]."Quantization of Deep Convolutional Networks for Efficient Inference: A Survey" by Jiquan Ngiam et al, June 2018

[3]."A Survey on Neural Network Quantization" by Markus Nagel et al, August 2020

[4]."A Survey on Quantization Methods for Efficient Neural Network Inference" by Xin Dong et al, March 2021

[5]."Quantization in Deep Neural Networks: A Survey "by Yash Mehta et al, August 2021

[6] Ying Wang et al (March 2022) did a Comprehensive survey on neural network quantization techniques andhardware acceleration.

[7] Zhenhua Zhang et al (January 2022) did a Comprehensive review of low-bit quantization methods forefficientDNN inference.

[8] Han, S., Mao, H., & Dally, W. J." Learning both Weights and Connections for Efficient Neural Networks"

[9] Li, H., Cai, H., Chen, T. Q., & Dally, W." Pruning Convolutional Neural Networks for Resource EfficientInference"

[10] Frankle, J., & Carbin, M." The Lottery Ticket Hypothesis: Finding Sparse, Trainable Neural Networks"

[11] Liu, Z., De Sa, C., van der Maaten, L., & Tang, P. T. P." Rethinking the Value of Network Pruning"

Molchanov, D., Mao, H., & Carbin, M." Sparsity- Invariant CNNs.