II. METHODOLOGY

C. Global Unstructured Pruning and Fine-tuning

Global Unstructured Pruning, as referenced in [4], [7], and [8], is an advanced model compression strategy that uniformly targets all model parameters, including weights and biases, eschewing the specificity of individual layers or structures. The core strategy is to create a sparse model by nullifying a pre-determined fraction of parameters, generally guided by a pruning ratio. This procedure involves sorting weights by their absolute magnitude, and setting the smallest percentile to zero, thereby effectively reducing computational overhead and storage requirements while aiming to preserve model performance.

Nevertheless, the pruning process may instigate a decrease in model performance. Hence, a crucial step that ensues pruning is fine-tuning. This stage involves the recalibration of the weights of the pruned model, typically employing a reduced learning rate to inhibit drastic fluctuations that could lead to overfitting. Fine-tuning is pivotal in facilitating the pruned model's adaptation to its new architecture, ensuring it remains proficient in tackling the original task while maintaining minimal model size and computational complexity. In the context of this experiment, fine-tuning effectively reinstated the model's performance to a level comparable to the original, pre-pruned model.

D. Knowledge Distillation

Following the pruning and fine-tuning processes, Knowledge Distillation is enacted as proposed in [4] and [14]. This methodology necessitates the pruned model to function as a student, with the original model serving as the teacher. Specifically, ResNet18 from torchvision.models is utilized as the student model. Through Knowledge Distillation, the pruned model learns to emulate the performance of the original, larger model, effectively distilling the "knowledge" of the expansive model into the more compact, pruned model.

III. EXPERIMENTAL STUDY

⚫ Optimizer

Our training methodology employs the Adam optimizer, an acronym for Adaptive Moment Estimation. Specifically designed for deep learning applications, Adam extends the stochastic gradient descent method. This optimizer brilliantly amalgamates the salient features of two other well-established optimization algorithms - AdaGrad and RMSProp, thereby facilitating an optimization strategy that can adeptly handle sparse gradients in noisy problem scenarios.

Adam's computational strategy involves calculating an exponential moving average of the gradient and the squared gradient. These moving averages decay at rates determined by the hyperparameters beta1 and beta2. The initial step size, which is tantamount to the learning rate, can be regulated using the hyperparameter alpha.

Adam offers several distinguishing characteristics:

⚫ It is simple to implement and demonstrates high computational efficiency with minimal memory requirements.

⚫ Its parameter updates exhibit invariance to the rescaling of the gradient, which aids in maintaining the relative updates of model weights.

⚫ Its hyperparameters have intuitive implications and generally require minimal tuning.

⚫ The step sizes are effectively bounded by the learning rate, which assists in preventing excessively large parameter updates.

⚫ It exhibits robust capability in handling sparse gradients and noisy data, making it particularly suitable for scenarios with extensive data and parameters.

⚫ It is effective for non-stationary objectives.

⚫ It implements a form of step size annealing, which aids in stabilizing the learning process and reduces the need for manual tuning of the learning rate.

In practical applications, Adam often outperforms SGD in terms of convergence speed and can potentially yield superior performance on the test set.

⚫ Loss

For our multi-class problem, we deploy Cross-Entropy Loss as our objective function. This loss function is particularly suitable for multi-class classification problems where the target labels are one-hot encoded. Cross-Entropy Loss comprises a Softmax activation and a Cross-entropy Loss.

The computation of Cross-Entropy Loss involves two integral parts:

Softmax activation: This activation function transforms raw score vectors, or logits, into probabilities by exponentiating them and then normalizing them.

Cross-Entropy Loss: The Cross-Entropy Loss is computed as the negative log likelihood of the correct class:

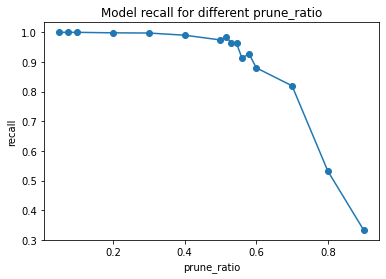
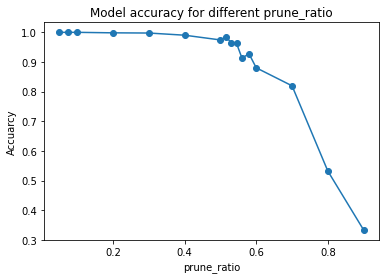
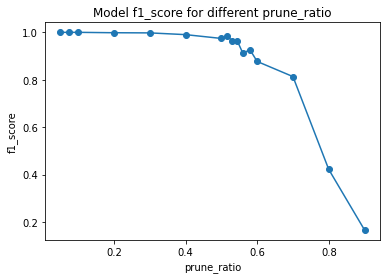
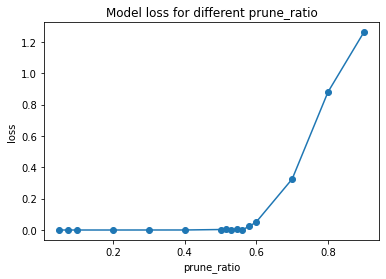
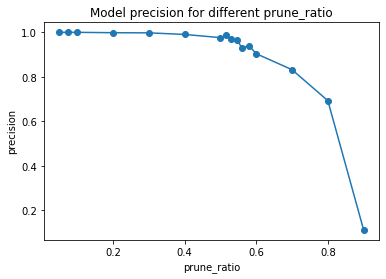
CE = -∑\_i (y\_i \* log(p\_i))

where CE denotes the cross-entropy, y\_i represents the ith class label (in one-hot encoded form), and p\_i signifies the predicted probability of the ith class.

The softmax operation ensures the sum of the output probabilities across all classes is unity, with the output probability of each class being influenced by other classes. This property makes Cross-Entropy Loss suitable for multi-classification problems, as it penalizes the model heavily when it assigns high probability to the incorrect class. Therefore, the Cross-Entropy Loss guides the model to refine its predictions by iteratively adjusting the parameters to minimize the loss.

⚫ Global Unstructured Pruning and Fine-tuning

In alignment with the methodologies proposed in [4], [7], and [8], our initial step involved applying Global Unstructured Pruning to the baseline model. A critical decision in this procedure was the determination of an appropriate pruning ratio. Following extensive experimentation with a spectrum of ratios, we identified that a pruning ratio of 0.515 struck an optimal balance between model sparsity and computational performance. With this ratio, we proceeded to nullify the least significant 51.5% of weights, as sorted by their absolute value, within the model parameters.



Subsequent to the pruning process, a slight decrement in model performance was perceptible. In response, we initiated a fine-tuning phase, involving the delicate adjustment of the pruned model's weights utilizing a diminished learning rate. This process enabled the pruned model to accommodate its new, sparser architecture, thereby enhancing its efficacy in executing the original task. The fine-tuning phase successfully restored the model's performance to a level commensurate with the original, unpruned model.

⚫ Knowledge Distillation

Upon the conclusion of the pruning and fine-tuning stages, we implemented Knowledge Distillation as delineated in [4] and [14]. In this phase, the pruned model assumed the role of a teacher model, while a ResNet18 model, procured from torchvision.models, was employed as the student model.

The student model was subjected to a 50-epoch training regime, during which it learnt to replicate the behavior of the teacher model. The overarching aim was to distill the 'knowledge' encapsulated in the pruned model into the more compact ResNet18 model. Upon the completion of the distillation process, the ResNet18 model not only matched the performance of the pruned model but occasionally outperformed it, all while maintaining a significantly reduced model size.

IV. RESULT AND EVALUATION

Model Compression Results

The original model size was 92180 KB, while the compressed model size after pruning and distillation was 43750 KB. The compression rate was calculated as:

Compression rate = (Original model size - Compressed model size) / Original model size = (92180 - 43750) / 92180 ≈ 52.5% This model compression method, inspired by the PEEL technique mentioned in [4], resulted in a compression rate of 52.5% with only a minor trade-off in terms of accuracy. These results demonstrate the efficacy of this model compression method.

涉及到的参考文献如下：

1. **[4] Hou, Y., Ma, Z., Liu, C., Wang, Z., & Loy, C. C. (2021). Network pruning via resource reallocation. arXiv preprint arXiv:2103.01847. ：（主要参考）**
2. [7] Iofinova, E., Peste, A., & Alistarh, D. (2023). Bias in pruned vision models: In-depth analysis and countermeasures. arXiv preprint arXiv:2304.12622.
3. [8] Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. In Advances in neural information processing systems (pp. 1135-1143).：
4. [14] Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. arXiv preprint arXiv:1503.02531.