Distillation from an Ensemble of Pruned Networks

Huzaifa Naseer
Department of Computer Science and Engineering
National Institute of Technology
Srinagar, India.

Hurmat Khalid
Department of Computer Science and Engineering
National Institute of Technology
Srinagar, India.

Tausifa Jan Saleem
Bharti School of Telecommunication technology and Management
Indian Institute of Technology
Delhi, India.

Brejesh Lall
Department of Electrical Engineering
Indian Institute of Technology
Delhi, India.

***Abstract*—Knowledge Distillation transfers knowledge from a complex teacher to a lightweight student. Recent results demonstrate that the pruned teacher teaches better than the unpruned one. In this work, we propose a novel approach to further improve the performance of the student model by distilling from an ensemble of pruned models instead of a single one. Ensembling leverages the collective intelligence and diversity of the pruned models. This improves the robustness and generalization ability of the teacher model. Experimental results demonstrate substantial improvement in the accuracy of the student model using this approach.**

***Index Terms*—Knowledge distillation, Pruning, Ensembling, Learning rate rewinding.**

1. INTRODUCTION

Neural networks [1] have revolutionized the field of artificial intelligence [2], enabling unprecedented advancements in tasks ranging from image recognition [3] to natural language processing [4]. However, their widespread adoption has been impeded by inherent computational complexity and memory requirements. As neural networks grow in size and depth to tackle increasingly complex problems, they demand substantial computational resources during both training and inference phases. This overhead poses significant challenges for deploying neural network models on resource-constrained devices such as mobile phones, embedded systems, and Internet of Things (IoT) devices.

To address the inefficiencies associated with large neural networks, researchers have developed various optimization techniques. Among these, four prominent methods stand out: knowledge distillation [5], pruning [6], low-rank factorization [7], and quantization [8]. Knowledge distillation [5] involves transferring learned insights from a large teacher model to a smaller student model, enabling the student to achieve comparable performance with fewer computational resources. Pruning [6] reduces model size by eliminating redundant neurons and connections, resulting in a sparser and more efficient network. Low-rank factorization [7] approximates the original weight matrices with lower-rank versions, simplifying the model and enhancing computational efficiency. Quantization [8] reduces the precision of model weights and activations, decreasing model size and accelerating inference. These techniques enable the creation of lightweight models suitable for deployment on edge devices, mobile platforms, and other resource-constrained environments.

Among these techniques, knowledge distillation [5] stands out as particularly promising. This approach involves transferring the learned insights from a teacher model, typically a deep neural network, to a smaller student model. By leveraging the distilled knowledge, the student model can achieve performance comparable to its teacher while requiring fewer computational resources for inference. Recent research [9] has shown that a pruned teacher model can be even more effective in transferring knowledge than a complex, unpruned model. Building on this insight, our paper explores the effectiveness of distillation from the ensemble of pruned models in enhancing student model performance. Ensembling [10] involves combining multiple models to improve overall performance and robustness. We aim to determine the effectiveness of our method in enhancing both efficiency and accuracy.

This paper is structured into several sections and subsections that provide a comprehensive exploration of our methodology. We begin with the Preliminaries, discussing the foundational concepts required to further study this topic. Next, we review recent work in the field, focusing on advancements in knowledge distillation. Following this, we detail our methodology, emphasizing how we have distilled from an ensemble of pruned models to achieve more generalized results. We then describe our experimental results, including the datasets used, model architectures, and training procedures. Subsequently, we discuss how our optimization technique impacts model performance across various evaluation criteria. Finally, we conclude by summarizing our findings, emphasizing the advantages of our methodology, and suggesting directions for future research.

2. PRELIMINARIES

A. Knowledge Distillation

Knowledge distillation [5] is a method in machine learning used to transfer knowledge from a large, complex model

(teacher) to a smaller, more lightweight model (student). The goal is to distill the knowledge contained in the teacher model into a more compact form that can be efficiently deployed on resource-constrained devices. This process involves training the student model to mimic the behavior and predictions of the teacher model by minimizing a distillation loss function, typically defined as a combination of the standard cross-entropy loss between the student's predictions and the ground truth labels, and the Kullback-Leibler [11] divergence [11] between the soft targets provided by the teacher model and the student's predictions. Mathematically, the distillation loss function can be expressed as:

$$\mathcal{L}_{\text{distill}} = (1-\lambda)\cdot\text{CE}(y_{\text{true}}, y_{\text{student}}) + \lambda\cdot\text{KL}(P_{\text{teacher}}, P_{\text{student}}) \quad (1)$$

Where CE denotes the cross-entropy loss, KL is the Kullback-Leibler divergence [11], y_{true} are the true labels, y_{student} are the student model's predictions, P_{teacher} and P_{student} are the soft target distributions provided by the teacher and predicted by the student, respectively, and λ is a hyperparameter controlling the trade-off between the two terms in the loss function.

In knowledge distillation [5], the soft targets provided by the teacher model are often generated by applying a softmax function [12] over the logits produced by the teacher network. Let's denote the logits produced by the teacher model for a given input as $\mathbf{z}_{\text{teacher}}$, and the corresponding soft targets (probability distribution) as $\mathbf{P}_{\text{teacher}}$. The softmax function [12] is applied element-wise to the logits as follows:

$$P_{\text{teacher}}(i) = \frac{\exp(z_{\text{teacher}}(i)/T)}{\sum_j \exp(z_{\text{teacher}}(j)/T)} \quad (2)$$

where T is a temperature parameter, which is typically set to a value greater than 1 to soften the distribution. Softening the distribution helps in transferring more nuanced information from the teacher to the student.

Various types of knowledge distillation [5] techniques have been proposed:

- **Vanilla Knowledge Distillation** [13] - Traditional method where a student model learns from either hard labels or soft targets provided by a teacher model.
- **Attention Transfer** [14] - Technique involving the transfer of attention maps from a teacher model to a student model.
- **FitNets** [15] - Approach that utilizes intermediate representations from a teacher model to guide the training of a student model.
- **Similarity-Preserving Knowledge Transfer** [16] - Method ensuring that similarity relationships between instances are maintained during knowledge transfer.
- **Self-Knowledge Distillation** [17] - Process of distilling knowledge from different stages of the same model.
- **Multi-Teacher Knowledge Distillation** [18] - Knowledge distillation from multiple teacher models to a single student model.
- **Dark Knowledge** [19] - Utilization of additional knowledge contained in the distribution of logits generated by a teacher model.
- **Probabilistic Knowledge Distillation** [20] - Incorporation of uncertainty estimates from a teacher model into the distillation process.

B. Pruning

Pruning [6] is a technique used in deep learning to reduce the size of a neural network by removing redundant or less important parameters. The objective is to create a more compact model with minimal loss of performance. Pruning can be applied to various parts of the network, including weights and entire neurons, based on certain criteria such as weight magnitudes, activations, or gradients. This process typically involves iteratively identifying and removing unimportant components while retraining the pruned model to recover or maintain its performance. Pruning consists of two main types:

1) Structured Pruning: Structured pruning [21] involves removing structured patterns while preserving the network's structure. This includes techniques such as filter pruning [22], channel pruning [23], or pruning entire layers [24]. By removing entire structures, structured pruning can often achieve significant compression with minimal loss of performance. Mathematically, structured pruning can be represented by modifying the optimization objective of the neural network training process. For example, in filter pruning, the optimization objective may be augmented with a regularization term that encourages sparsity at the filter level:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \sum_i \|w_i\|_1 \quad (3)$$

Where $\mathcal{L}_{\text{data}}$ is the standard data loss term, w_i are the weights corresponding to a filter, and λ is a hyperparameter controlling the strength of the regularization.

2) Unstructured Pruning: Unstructured pruning [25] involves removing individual weights or neurons without preserving the network's structure. Unlike structured pruning, unstructured pruning may result in irregular sparsity patterns and require specialized techniques for efficient implementation and deployment. Mathematically, unstructured pruning can also be represented by modifying the optimization objective of the neural network training process. For example, in magnitude-based pruning [26], the optimization objective may be augmented with a sparsity-inducing regularization term, such as L1 regularization [27] :

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{data}} + \lambda \sum_i |w_i| \quad (4)$$

Where $\mathcal{L}_{\text{data}}$ is the standard data loss term, w_i are individual weights, and λ is a hyperparameter controlling the strength of the regularization.

C. Learning Rate Rewinding

Learning rate rewinding [28] in pruning algorithms periodically resets the learning rate to a previously recorded value, improving convergence and final performance during training. It stabilizes the optimization process, enabling better exploration of the parameter space. Mathematically, let θ_t denote the model parameters at iteration t, and η_{rewind} represent the learning rate. With learning rate rewinding, the update rule becomes:

$$\theta_{t+1} = \theta_t - \eta_{\text{rewind}} \cdot \nabla J(\theta_t) \tag{5}$$

Where $J(\theta_t)$ denotes the loss function evaluated at θ_t. This resetting facilitates smoother optimization trajectories, leading to enhanced convergence and performance in pruned neural networks.

D. Ensembling

Ensembling [10] is a powerful technique used in machine learning to improve the performance and robustness of predictive models by combining the predictions of multiple individual models. The basic idea behind ensembling is that by aggregating the predictions of diverse models, it is possible to achieve better overall performance than any single model alone. Ensembling can be applied to various types of models, including decision trees, neural networks, and other types of classifiers or regressors.

***Types of Ensembling**:*

- **Bagging (Bootstrap Aggregating)**: Bagging [29] involves training multiple instances of the same model on different subsets of the training data, typically using bootstrapping, and then aggregating their predictions, often by averaging or taking a majority vote. This helps reduce variance and improve generalization.
- **Boosting**: Boosting [30] is a sequential ensemble technique where each subsequent model is trained to correct the errors made by the previous ones. Popular boosting algorithms include AdaBoost, Gradient Boosting Machines (GBM), and XGBoost.
- **Stacking (Stacked Generalization)**: Stacking [31] combines the predictions of multiple diverse models by training a meta-model (or blender) on top of their outputs. The meta-model learns to combine the predictions of the base models to make the final prediction.
- **Voting**: Voting [32] combines the predictions of multiple models by taking a majority vote (for classification) or averaging (for regression). Different types of voting include hard voting (simple majority) and soft voting (weighted average based on confidence scores).
- **Random Forest**: Random Forest [33] is an ensemble learning method that builds multiple decision trees during training and outputs the mode (classification) or average prediction (regression) of the individual trees.
- **Ensemble Averaging**: Ensemble averaging [34] involves combining the predictions of multiple models by taking the mean or weighted average of their outputs. This helps smooth out biases and errors in individual models, leading to more robust predictions.

Ensembling [10] techniques leverage the diversity among individual models to capture different aspects of the data distribution, leading to improved performance, robustness, and generalization ability. Each ensemble method has its strengths and weaknesses, and the choice of technique often depends on the specific characteristics of the dataset and the problem at hand.

A general equation for ensembling can be expressed as follows:

$$\hat{y}_{\text{ensemble}} = \frac{1}{N} \sum_{i=1}^{N} \hat{y}_i \tag{6}$$

Where $\hat{y}_{\text{ensemble}}$ is the ensemble prediction, $\hat{y}_i$ are the predictions of individual models, and N is the number of models in the ensemble.

3. Related Works

Knowledge distillation, introduced by Bucilu˘a et al. [35], involves training smaller models (***students***), to mimic larger models (***teachers***). The foundational framework for this approach was established by Hinton et al. [36] who defined a learning paradigm where a larger teacher network guides the training of a smaller student network using soft labels. This technique has evolved to include various methods, such as distillation from logits or intermediate features. Logit distillation [37] focuses primarily on regularization and optimization, whereas state-of-the-art methods often transfer intermediate features or sample correlations directly from the teacher to the student. Although feature-based methods generally offer superior performance compared to logits-based methods, they also come with higher computational costs.

Efforts to enhance knowledge distillation have led to the exploration of additional strategies. For instance, some researchers have introduced extra losses on intermediate feature maps of the student to better align them with the teacher's representations. Others have investigated bidirectional knowledge distillation [38], the averaging of consecutive student models [39], or modifications to the teacher's loss function [40]. Sequential knowledge distillation [41], involving repetitive distillation, has yielded mixed results. Additionally, knowledge distillation has been successfully applied to sequence modeling, semi-supervised learning, domain adaptation, and multi-modal learning, showcasing its broad applicability.

The concept of knowledge transfer between models, initially proposed by Breiman and Shang [42], has been adapted for neural networks, primarily for model compression. Various strategies have been explored, such as compressing an ensemble of networks into a single network or training a shallow network to replicate a deeper one. Innovations in this area include linear projection layers, attention map transfer, noise-based regularizers, and data-free knowledge distillation. Self-distillation [41], where a teacher model is distilled into a student model of identical architecture, has shown promise

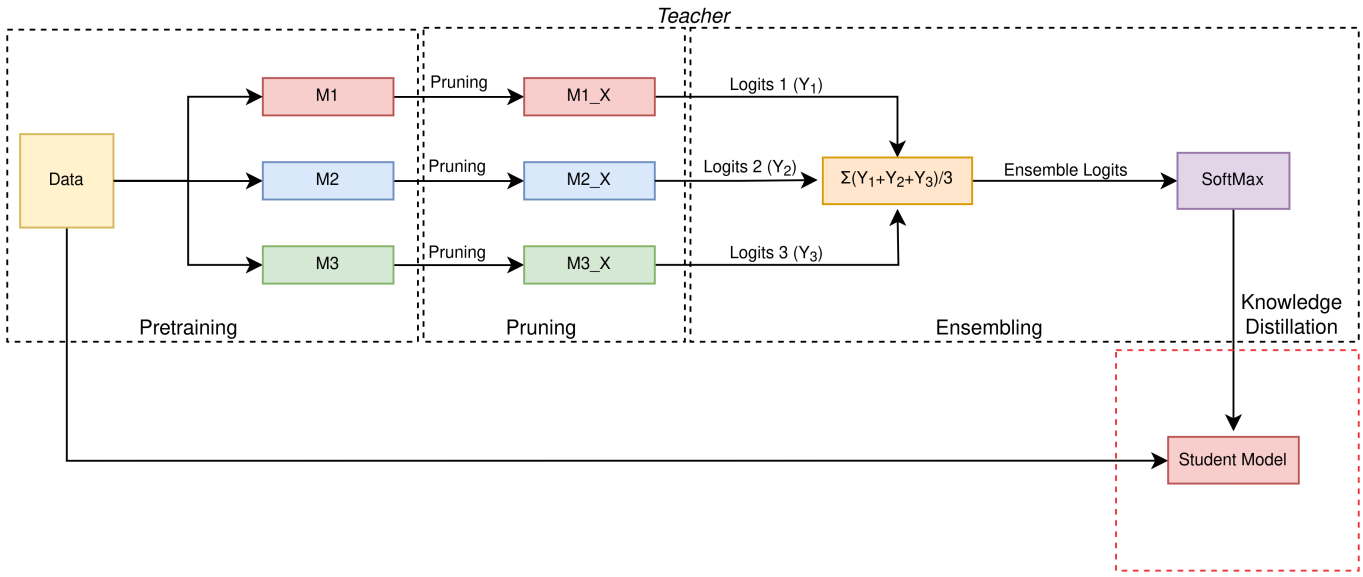


Fig. 1: Model Architecture

in enhancing student performance. Knowledge distillation has also been extended beyond supervised learning to encompass unsupervised, semi-supervised, and multi-task scenarios.

Our study builds on the ***prune then distill*** approach proposed by Jinhyuk Park and Albert [9], which demonstrates that pruned teacher models can be more effective than unpruned ones. In our work, we investigate how the ensemble of pruned networks behaves as a teacher. We first train a teacher model from different initialization points to capture diverse data representations. These teacher models are then pruned to reduce redundancy while retaining essential information. By aggregating the pruned teacher models into an ensemble, we leverage the collective intelligence of multiple networks. Finally, we perform knowledge distillation from this ensemble to train compact student models.

4. Methodology

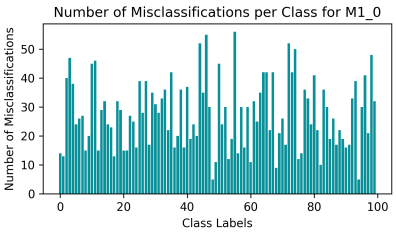
This section presents a detailed methodology of our approach, as illustrated in Fig. 1. Our method involves a systematic sequence of steps to optimize neural network performance and ensure computational efficiency:

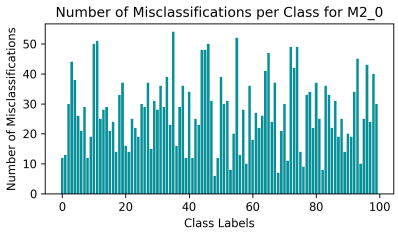
- Pretraining: We start by training multiple instances of the VGG-19 architecture on the CIFAR-100 dataset. Each model is initialized from different intialisation points to capture diverse data representations. Specifically, the models are denoted as M1, M2, and M3 with different initialization points having random seed values 800, 1000, and 1300, respectively. This diversity helps explore the impact of initialization on model performance and subsequent processing.
- Pruning: After training, we apply pruning to the models. Pruning is conducted using the learning rate rewinding algorithm [28], which iteratively removes less important weights or neurons based on criteria such as weight magnitude. The pruned models are labeled as M1_Y, M2_Y, and M3_Y, where Y denotes the pruning ratio applied. This step aims to reduce model redundancy while retaining critical information.
- Ensemble Learning: Following pruning, we combine the pruned models into an ensemble, denoted as E_Y. This ensemble, which includes M1_Y, M2_Y, and M3_Y, leverages the strengths of multiple models to improve overall performance and robustness. Ensemble learning helps aggregate the predictions of the pruned models to enhance accuracy and stability.
- Knowledge Distillation: Finally, we perform knowledge distillation [5] on the ensemble model to train a smaller, more compact student model. The ensemble model E_Y transfers its distilled knowledge to the student model (KD_E_Y), which is a VGG-11 architecture. Knowledge distillation facilitates the transfer of rich information encoded in the ensemble into a streamlined model, making it suitable for deployment in resource-constrained environments.

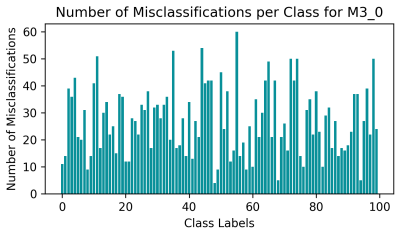
Through this methodology, we aim to strike a balance between model performance and computational efficiency. By systematically training, pruning, and distilling models, we explore the parameter space of neural network architectures to identify configurations that optimize both efficiency and accuracy. This comprehensive approach ensures that the resulting models are well-suited for practical deployment scenarios where resource constraints are a consideration.

5. Experimental Results

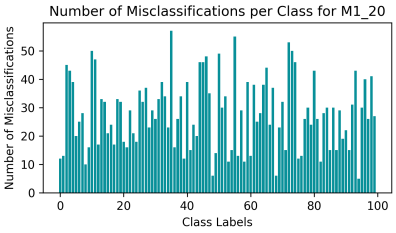
In this section, we detail the experimental results of our study, which encompass training multiple instances of the VGG-19 architecture on the CIFAR-100 dataset, exploring the impact of diverse model initialization on pruning and ensemble

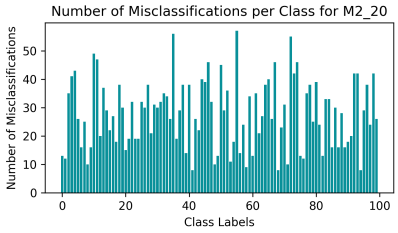

Number of Misclassifications per Class for M1_0
Number of Misclassifications
Class Labels

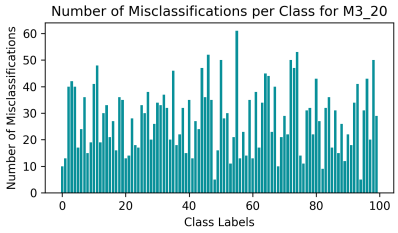

Number of Misclassifications per Class for M2_0
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M3_0
Number of Misclassifications
Class Labels

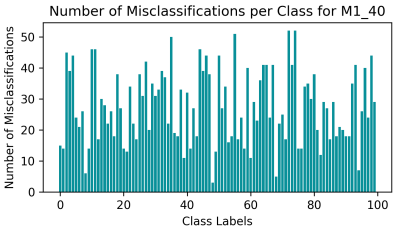
a

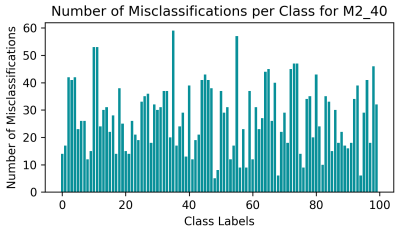

Number of Misclassifications per Class for M1_20
Number of Misclassifications
Class Labels

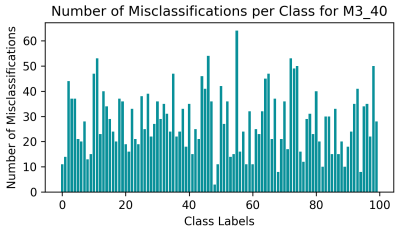

Number of Misclassifications per Class for M2_20
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M3_20
Number of Misclassifications
Class Labels

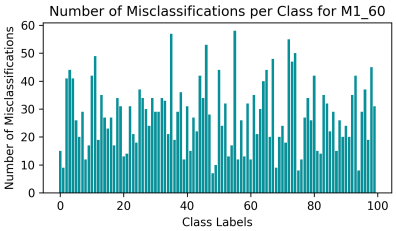
b

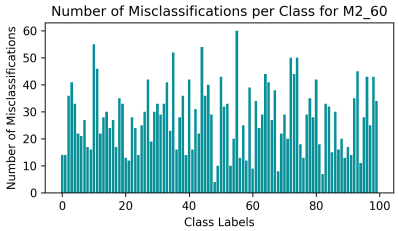

Number of Misclassifications per Class for M1_40
Number of Misclassifications
Class Labels

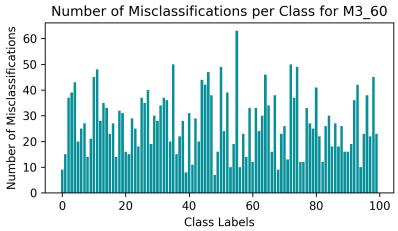

Number of Misclassifications per Class for M2_40
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M3_40
Number of Misclassifications
Class Labels

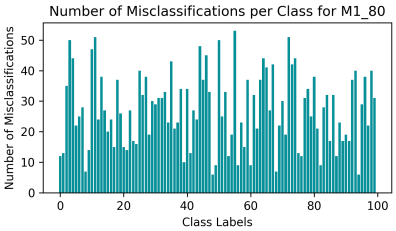
c

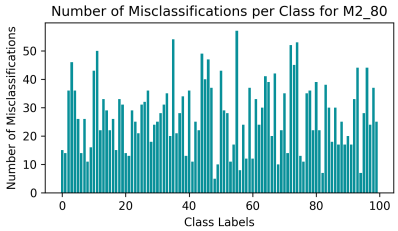

Number of Misclassifications per Class for M1_60
Number of Misclassifications
Class Labels

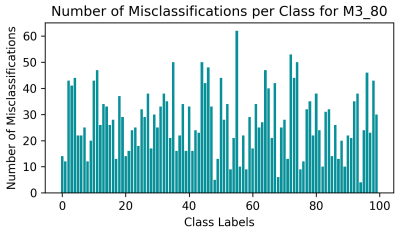

Number of Misclassifications per Class for M2_60
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M3_60
Number of Misclassifications
Class Labels

d


Number of Misclassifications per Class for M1_80
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M2_80
Number of Misclassifications
Class Labels


Number of Misclassifications per Class for M3_80
Number of Misclassifications
Class Labels

e

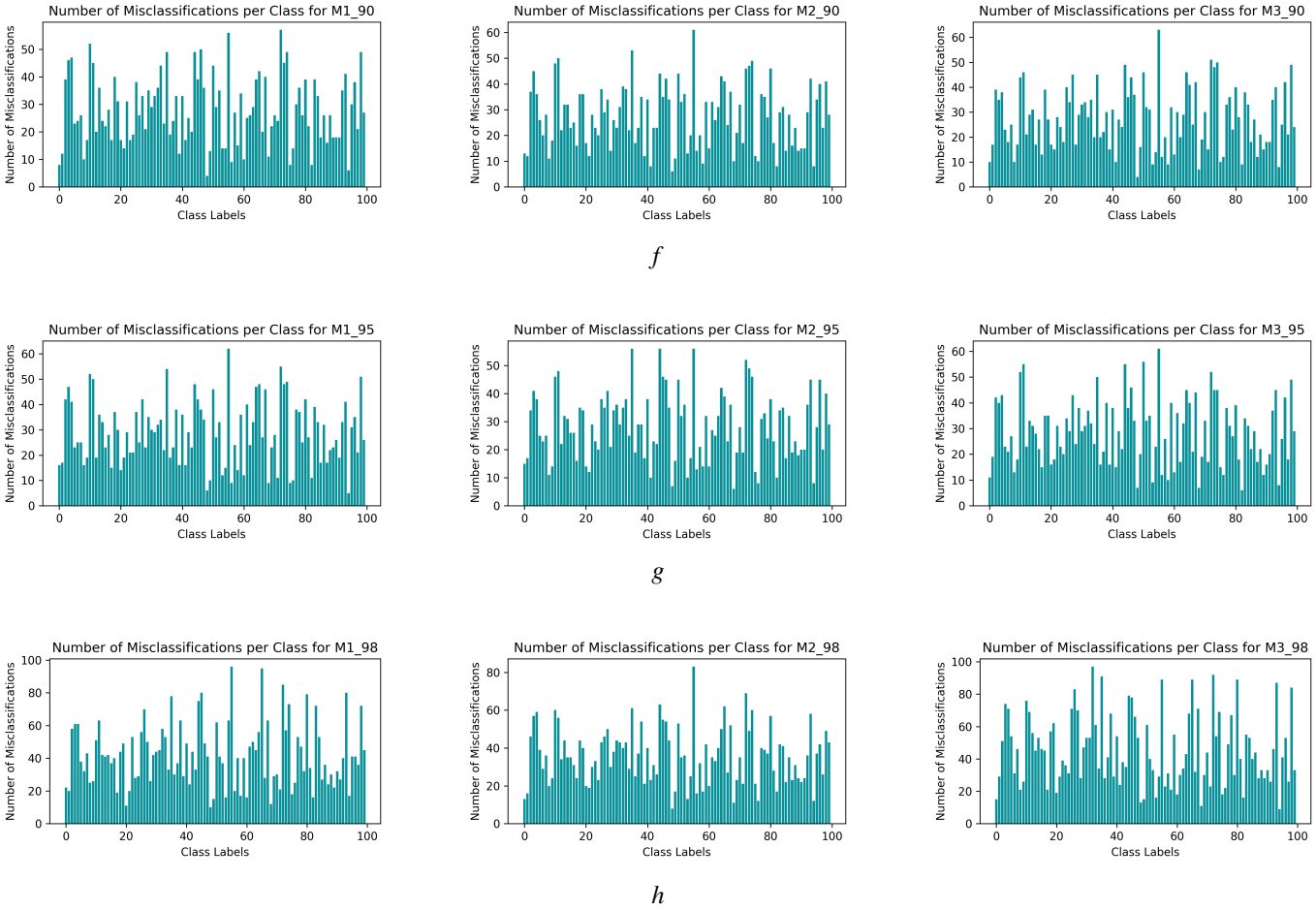


Fig. 2: Misclassifications per class for models MX_Y, with each row representing the same pruning ratio Y and each graph in a row corresponding to different initialization points.

learning, and navigating the delicate balance between model sparsity and performance optimization.

Initially, we trained a VGG-11 model without any teacher guidance shown in Table I, achieving an accuracy of 70.81%. This served as our baseline performance metric.

Following that, we pretrained multiple instances of the VGG-19 architecture on the CIFAR-100 dataset, with each instance initialized from a distinct initialization point. The pretraining process aimed to explore the diversity in model initialization and its impact on subsequent pruning and ensemble learning. The hyperparameters involved in the pretraining process are listed in Table II

TABLE I: Accuracy of the student model when trained independently, without the guidance of a teacher model.

Teacher Model Accuracy (%)	Student Model Accuracy (%)
None	70.81

In Table III, we present the performance of three models, which we refer to as M1, M2, and M3.

TABLE II: Hyperparameters Used in Pretraining

Hyperparameter	Value
Batch Size	128
Number of Epochs	200
Learning Rate Schedule (Pretraining)	Epochs 60, 120, 160
Optimizer	Nesterov
Weight Decay (Pretraining)	0.0005

TABLE III: Pretraining Results

Teacher Model Type	Model	Accuracy (%)
VGG19	M1	72.37
VGG19	M2	72.44
VGG19	M3	72.96

Fig. 2 illustrates the three models trained from different initialization points with their corresponding pruning ratios, demonstrating how models with the same pruning ratio but different initialization points misclassify different labels. This highlights the complementary nature of the models, as each one captures unique aspects of the data, leading to different

TABLE IV: Hyperparameters Used in Pruning

Hyperparameter	Value
Pruning Ratios	0% to 98%
Fine-tuning Epochs	130
Batch Size (Fine-tuning)	128
Learning Rate Schedule (Fine-tuning)	Epochs 39, 84
Optimizer	Nesterov
Weight Decay (Fine-tuning)	0.0002

misclassification patterns.

Following the pretraining phase, we applied pruning techniques to induce sparsity in the trained models. The pruning process, facilitated by the learning rate rewinding algorithm, involved removing less important weights from the network based on magnitude based criteria. We pruned the models with selected pruning ratios ranging from 0 to 98% to strike a balance between model size reduction and performance degradation. The hyperparameters used in the pruning process are listed in Table IV.

In Table III, we compare pruning ratios and accuracies for models M1, M2, and M3. The pruned models are denoted using the naming convention MX_Y, where X = 1, 2, 3 corresponds to the unpruned models (M1, M2, M3), and Y represents the applied pruning ratio. For example, M1_0.20 indicates that the unpruned model M1, initialized with seed 800, was pruned with a ratio of 20%.

After pruning, we proceeded with ensemble learning by combining the predictions of these models. Specifically, we utilized averaging to ensemble the models obtained from different seeds. This approach allowed us to harness the complementary strengths of the individual models and improve overall accuracy.

Table V summarizes the accuracy achieved through ensembling for various pruning ratios. The ensemble models are denoted by the naming convention E_X, where X represents the pruning ratio applied. For instance, E_0.20 denotes the ensemble of models pruned with a ratio of 20%.

After pruning and ensembling, we applied knowledge distillation to transfer the collective knowledge of the pruned teacher ensemble to a smaller, more compact student model of type VGG-11. Knowledge distillation involved training the student model with the distilled knowledge from the teacher ensemble to mimic their behavior and predictions. The hyperparameters used in the knowledge distillation process are listed in Table VI.

In Table VII, KD_MX_Y represents the individual knowledge distillation results of models, where X (1, 2, 3) indicates models trained from different initialisation points, and Y denotes the pruning ratio applied during evaluation. For example, KD_M1_0.20 indicates knowledge distillation results from the model initialized with random seed 800 and pruned at 20% ratio.

Furthermore, KD_E_Y denotes the accuracy of the knowledge distillation ensemble, which represents the average predictions of the pruned models M1_Y, M2_Y, and M3_Y where Y is the pruning ratio. The ensemble combines the distilled knowledge from these models to improve overall predictive accuracy.

The detailed results of the knowledge distillation process, including individual models and ensemble performance, can be found in Table VII.

As the pruning ratio increased, the teacher models became less complex due to the removal of non-essential parameters, leading to a more compact representation of the knowledge encoded in the ensemble. Interestingly, we observed in Table VII that despite the decreasing complexity of the teacher models, the student model continued to learn better when trained using knowledge distilled from the pruned teacher ensemble. This phenomenon underscores the efficacy of knowledge distillation in transferring distilled knowledge from simpler teacher models to improve the learning of the student model.

In Fig. 3 we analyze the training loss vs epochs for models trained at different pruning ratios. Each graph represents the training loss trends as the number of epochs increases for a specific pruning ratio. Notably, models distilled from the ensemble of pruned models exhibit steeper curves and lower losses compared to those distilled directly from a single pruned model. This indicates that the distillation process involving an ensemble of pruned models results in reduced losses, emphasizing the potential benefits of combining pruning with ensemble learning and knowledge distillation.

Furthur, Fig. 4 illustrates the trends observed in our knowledge distillation experiments. In Fig. 4, we plotted the graphs for individual knowledge distillation results represented by KD_MX_Y where X = 1, 2, 3 and the accuracy of the knowledge distillation ensemble is represented by KD_E_Y across varying pruning ratios Y. The results depicted in Fig. 4 demonstrate the effectiveness of knowledge distillation after ensembling in enhancing the learning performance of the student model, even as the complexity of the teacher models is reduced through pruning.

6. Conclusion

This study explored the effectiveness of training teacher models from various initialization points, followed by pruning and ensembling them to create a robust teacher model. Through rigorous experimentation, it was demonstrated that this approach yields promising results, showcasing reduced losses and significantly improved performance compared to individual teacher models.

By leveraging the diversity brought by different initialization points and using the ensemble of those pruned models as a teacher, we were able to create a powerful teacher model. Furthermore, distillation from this ensemble model led to even greater enhancements, underscoring the effectiveness of knowledge transfer techniques in improving model efficiency and performance.

Moving forward, further research can delve deeper into understanding the underlying mechanisms driving the effectiveness of ensemble models and distillation techniques. Additionally, exploring the scalability and applicability of

TABLE V: Comparison of test accuracy between individual models and their ensembles for different pruning ratios

Pruning Ratio (Y%)	**Accuracy (%)**			**Ensembling Accuracy (%)**
	M1_Y	**M2_Y**	**M3_Y**	
0	72.37	72.44	72.96	76.29
20	72.02	72.07	71.99	76.07
40	72.37	72.50	71.88	75.94
60	72.27	72.15	72.63	76.14
80	72.72	72.64	72.56	76.32
90	72.10	72.64	72.78	76.42
95	71.13	71.83	71.25	75.68
98	57.73	64.11	54.33	67.67

TABLE VI: Hyperparameters Used in Knowledge Distillation

Hyperparameter	**Value**
Batch Size	128
Number of Epochs	200
Learning Rate Schedule	Epochs 60, 120, 160
Optimizer	Nesterov
Weight Decay	0.0005
Alpha	0.95
Temperature	10

these methods across different domains and datasets would be valuable for expanding their utility in real-world scenarios.

REFERENCES

[1] Hervé Abdi, Dominique Valentin, and Betty Edelman. *Neural networks*. Number 124. Sage, 1999.

[2] Earl B Hunt. *Artificial intelligence*. Academic Press, 2014.

[3] Frank Y Shih. *Image processing and pattern recognition: fundamentals and techniques*. John Wiley & Sons, 2010.

[4] KR1442 Chowdhary and KR Chowdhary. Natural language processing. *Fundamentals of artificial intelligence*, pages 603–649, 2020.

[5] Jianping Gou, Baosheng Yu, Stephen J Maybank, and Dacheng Tao. Knowledge distillation: A survey. *International Journal of Computer Vision*, 129(6):1789–1819, 2021.

[6] Russell Reed. Pruning algorithms-a survey. *IEEE transactions on Neural Networks*, 4(5):740–747, 1993.

[7] Tara N Sainath, Brian Kingsbury, Vikas Sindhwani, Ebru Arisoy, and Bhuvana Ramabhadran. Low-rank matrix factorization for deep neural network training with high-dimensional output targets. In *2013 IEEE international conference on acoustics, speech and signal processing*, pages 6655–6659. IEEE, 2013.

[8] Jiwei Yang, Xu Shen, Jun Xing, Xinmei Tian, Houqiang Li, Bing Deng, Jianqiang Huang, and Xian-sheng Hua. Quantization networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 7308–7316, 2019.

[9] Jinhyuk Park and Albert No. Prune your model before distill it. In *European Conference on Computer Vision*, pages 120–136. Springer, 2022.

[10] Zhi-Hua Zhou, Jianxin Wu, and Wei Tang. Ensembling neural networks: many could be better than all. *Artificial intelligence*, 137(1-2):239–263, 2002.

[11] Solomon Kullback. Kullback-leibler divergence, 1951.

[12] Bolin Gao and Lacra Pavel. On the properties of the softmax function with application in game theory and reinforcement learning. *arXiv preprint arXiv:1704.00805*, 2017.

[13] Zhiwei Hao, Jianyuan Guo, Kai Han, Han Hu, Chang Xu, and Yunhe Wang. Revisit the power of vanilla knowledge distillation: from small scale to large scale. *Advances in Neural Information Processing Systems*, 36, 2024.

[14] Kai Zhang, Hefu Zhang, Qi Liu, Hongke Zhao, Hengshu Zhu, and Enhong Chen. Interactive attention transfer network for cross-domain sentiment classification. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 5773–5780, 2019.

[15] Ryusuke Murata, Fumiya Okubo, Tsubasa Minematsu, Yuta Taniguchi, and Atsushi Shimada. New perspective on input feature analysis for early feedback by student performance prediction considering the future effect. In *Companion proceedings of the 12th international conference on learning analytics & knowledge*, pages 95–97, 2022.

[16] Frederick Tung and Greg Mori. Similarity-preserving knowledge distillation. In *Proceedings of the IEEE/CVF international conference on computer vision*, pages 1365–1374, 2019.

[17] Mingi Ji, Seungjae Shin, Seunghyun Hwang, Gibeom Park, and Il-Chul Moon. Refine myself by teaching myself: Feature refinement via self-knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10664–10673, 2021.

[18] Yuang Liu, Wei Zhang, and Jun Wang. Adaptive multi-teacher multi-level knowledge distillation. *Neurocomputing*, 415:106–113, 2020.

[19] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Dark knowledge. *Presented as the keynote in BayLearn*, 2(2), 2014.

[20] Xianjing Han, Xuemeng Song, Yiyang Yao, Xin-Shun Xu, and Liqiang Nie. Neural compatibility modeling with probabilistic knowledge distillation. *IEEE Transactions on Image Processing*, 29:871–882, 2019.

[21] Sajid Anwar, Kyuyeon Hwang, and Wonyong Sung. Structured pruning of deep convolutional neural networks. *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, 13(3):1–18, 2017.

[22] Jian-Hao Luo, Jianxin Wu, and Weiyao Lin. Thinet: A filter level pruning method for deep neural network compression. In *Proceedings of the IEEE international conference on computer vision*, pages 5058–5066, 2017.

[23] Yihui He, Xiangyu Zhang, and Jian Sun. Channel pruning for accelerating very deep neural networks. In *Proceedings of the IEEE international conference on computer vision*, pages 1389–1397, 2017.

[24] Hao Li, Asim Kadav, Igor Durdanovic, Hanan Samet, and Hans Peter Graf. Pruning filters for efficient convnets. *arXiv preprint arXiv:1608.08710*, 2016.

[25] Zhu Liao, Victor Quétu, Van-Tam Nguyen, and Enzo Tartaglione. Can unstructured pruning reduce the depth in deep neural networks? In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 1402–1406, 2023.

[26] Vinnie Ko, Stefan Oehmcke, and Fabian Gieseke. Magnitude and uncertainty pruning criterion for neural networks. In *2019 IEEE International Conference on Big Data (Big Data)*, pages 2317–2326. IEEE, 2019.

[27] Praveen Kulkarni, Joaquin Zepeda, Frederic Jurie, Patrick Pérez, and Louis Chevallier. Learning the structure of deep architectures using l1 regularization. In *British Machine Vision Conference, 2015*, 2015.

[28] Advait Gadhikar and Rebekka Burkholz. Masks, signs, and learning rate rewinding. *arXiv preprint arXiv:2402.19262*, 2024.

[29] Leo Breiman. Bagging predictors. *Machine learning*, 24:123–140, 1996.

[30] Robert E Schapire et al. A brief introduction to boosting. In *Ijcai*, volume 99, pages 1401–1406. Citeseer, 1999.

[31] Bohdan Pavlyshenko. Using stacking approaches for machine learning models. In *2018 IEEE second international conference on data stream mining & processing (DSMP)*, pages 255–258. IEEE, 2018.

[32] Hanan G Ayad and Mohamed S Kamel. On voting-based consensus of cluster ensembles. *Pattern Recognition*, 43(5):1943–1953, 2010.

[33] Leo Breiman. Random forests. *Machine learning*, 45:5–32, 2001.

[34] Ury Naftaly, Nathan Intrator, and David Horn. Optimal ensemble averaging of neural networks. *Network: Computation in Neural Systems*, 8(3):283, 1997.

TABLE VII: Knowledge Distillation Results

Teacher Model Type	Pruning Ratio (Y%)	Student Model Type	Without Ensembling			With Ensembling
			KD_M1_Y	KD_M2_Y	KD_M3_Y	KD_E_Y
VGG-19	0	VGG-11	72.39	72.67	72.83	72.64
VGG-19	20	VGG-11	72.64	72.65	72.71	73.21
VGG-19	40	VGG-11	72.56	72.95	72.51	73.42
VGG-19	60	VGG-11	72.75	72.72	72.43	72.90
VGG-19	80	VGG-11	72.96	72.64	72.77	73.03
VGG-19	90	VGG-11	72.61	72.82	73.06	73.60
VGG-19	95	VGG-11	72.78	73.22	72.64	74.01
VGG-19	98	VGG-11	63.31	65.50	60.99	69.22

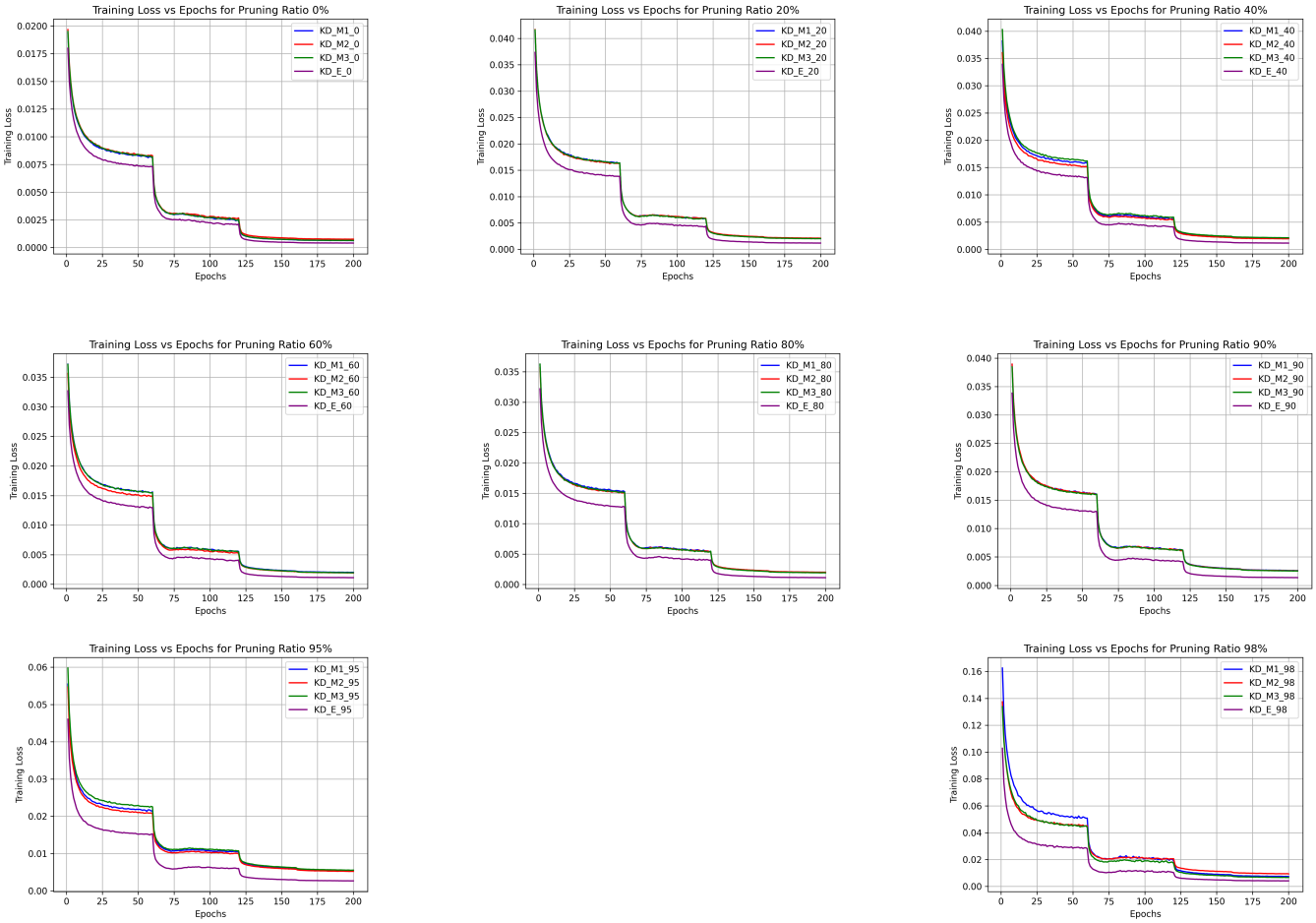


Fig. 3: Training loss versus epochs for student models obtained from different pruning ratios. Models distilled from ensembles of pruned models exhibit lower losses compared to those distilled directly from pruned models.

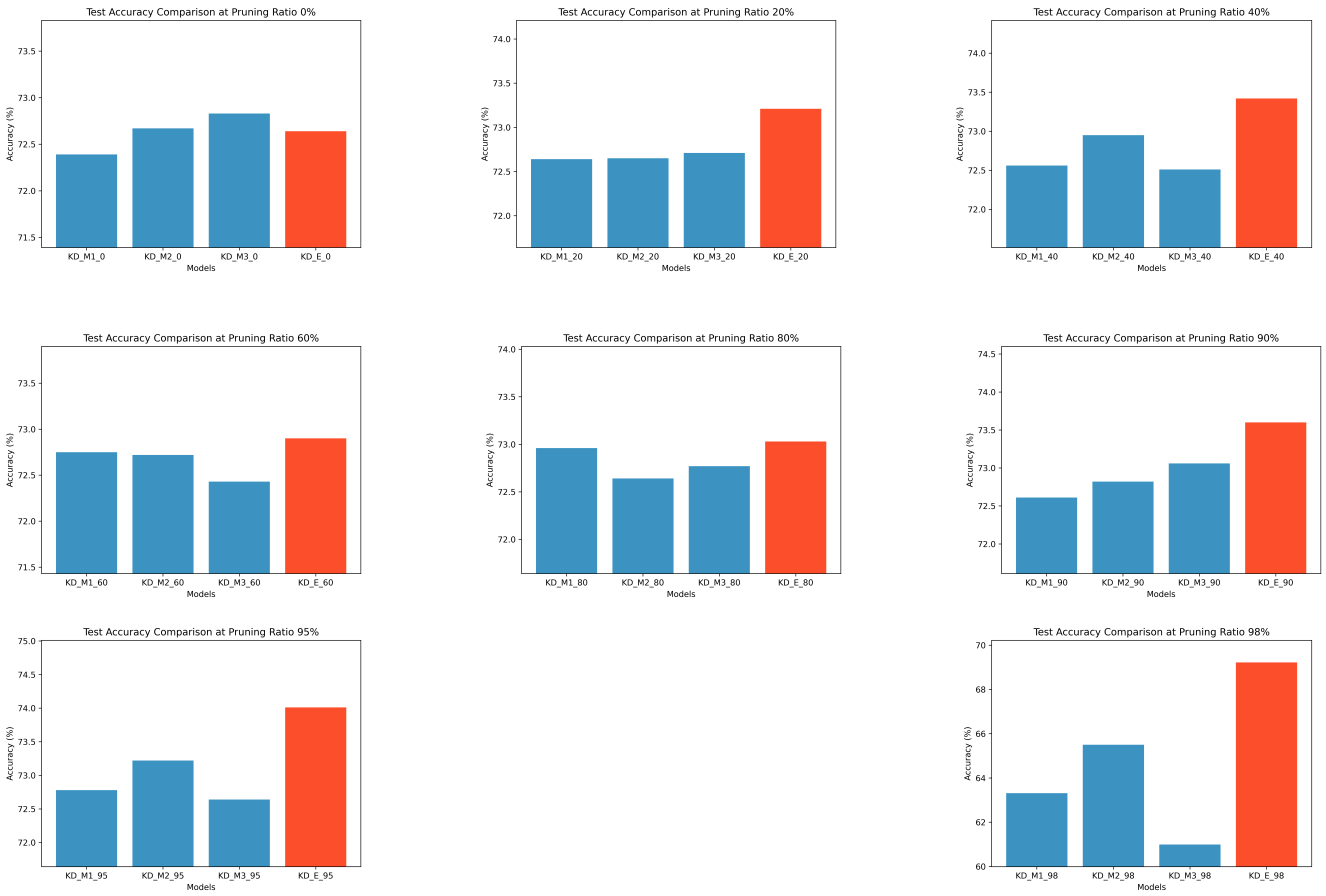


Fig. 4: Test accuracy comparison of student models after knowledge distillation.

[35] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006.

[36] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[37] Shangquan Sun, Wenqi Ren, Jingzhi Li, Rui Wang, and Xiaochun Cao. Logit standardization in knowledge distillation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 15731–15740, 2024.

[38] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[39] Antti Tarvainen and Harri Valpola. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results. *Advances in neural information processing systems*, 30, 2017.

[40] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32, 2019.

[41] Tommaso Furlanello, Zachary Lipton, Michael Tschannen, Laurent Itti, and Anima Anandkumar. Born again neural networks. In *International conference on machine learning*, pages 1607–1616. PMLR, 2018.

[42] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society Series B: Statistical Methodology*, 58(1):267–288, 1996.