Model Compression : Knowledge distillation and Node Sparse modeling

Yeonho Jung
Department of Statistics, Seoul National University, Seoul, South Korea
dusgh9514@snu.ac.kr

Abstract

Deep learning has become a state-of-the-art trend and is growing at a speedy pace when it comes to models, parameters and so on[4]. Basically, image recognition such as human, animal and object is developing themselves and is propagated greatly faster that expected. We generally have known that larger models that contain complex parameters in their structure would perform better than smaller ones, but it is not perfectly right. With the coexisting of 'Artificial Intelligence(AI)', 'Machine Learning(ML)', several kinds of methods are being shown and appealing to us interestingly. To say simple, one of methods is called 'Model Compression' skill that converts complex and larger model into a simple and smaller model. There exists numerous deep learning models in the world to compute accuracy in the process of training data-sets and validating, testing data-sets. Latency as well as prediction accuracy serves as an important indicator which means decreasing computational cost in total training epochs to make a prediction. We regard prediction accuracy and total taken time as one of the important indicators in the performance, but a fairly large model can be a burdensome to the CPU(GPU) in performing our work.

In the main body, two kinds of methods for model compression will be introduced and the principles of the model in addition to the basic structure will be shown with 'CIFAR10' and 'CIFAR100' data-sets. The term 'Model Compression' is already well known to the public, but not that much information have yet been conducted on the comparisons. We basically present a deep residual learning for image recognition[4] training network considerably deeper than previously used one. A residual network can be optimized easily gaining accuracy with increased depth.[7] By applying to the concept of deep convolutional neural network(CNN) in model compression, it may give a meaningful comparison on image classification. As a result, comparing two methods of model compression showed a bit different results in terms of prediction accuracy on data-sets. In addition, when comparing the results, a student model in knowledge distillation and a pruned model in node sparse modeling, the performance was improved compared to 'Wide-ResNet' model, which is a basic model. Both two methods of knowledge distillation[5] and node sparse modeling[3] in the model compression showed similar and superior results. Our proposed methods, 'Knowledge Distillation' and 'Node Sparse modeling' is shown to be 2x or more faster than

Preprint. Under review.

prior methods simultaneously having comparable performance in 'Wide-ResNet' on CIAFR-10 and CIFAR-100 data-sets.

1 Introduction

The model compression skill is said to have a purpose on reducing model size and improving speed simultaneously. A method for 'compression' is a basic term and used for explaining down-sizing from a large and complex model into smaller and simple model without loss of performance surprisingly[2]. Its skill offers several kinds of advantages such as model size reduction, improved inference speed, energy efficiency and maintained accuracy. By reducing the number of parameters in a model to make it simple and decreasing the total taken time, requirement for memory in CPU or GPU can be lighter than before. Two methods for model compression are 'knowledge distillation'[5] and 'Node sparse modeling'[3] applying to 'Wide residual network'[4]. Though well-known many methods exist in deep learning field, the comparative aspects of above two modeling techniques will prove to be more effective in terms of model compression techniques as a result.

Basically, transferring a teacher's knowledge to a student is a basic and main idea in knowledge distillation. In this part, training larger model at first is needed, then we can make a structure for extracting specific values from above model. It is called as a 'teacher' model. To transfer its knowledge from a teacher model to a student model meaning smaller model, this step 'distillation' is done in the next turn. The term 'model compression' is already being used by Rich Caruana in his paper[2] and it shows that extracted knowledge from a teacher model can be transferred to a student model.[5]. We simply need to understand the process how to transfer this ability. It is that instead of using 'hard targets' in the existing common method, we treat the class probabilities from a larger model called 'soft targets' for training smaller model. The paper of knowledge distillation[5] explains in the case of soft targets having high entropy, they would provide more information and get less variance in the gradient. In addition, the main term 'distillation' is to increase the temperature (T) of the final softmax until the larger model makes a soft set of targets.

'Node sparsity modeling' is the second model compression skill. Among numerous model compression strategies, the term 'sparse' produces more efficient models by computing the relevance of nodes in each layer of the basic-block and then converting insignificant nodes to 0 values following activation function. Practically, such abilities can be used to minimize complexity by zeroing off subsets of the original model parameters.[6]. We can alter the ratio in a sparse model based on the circumstances, and the basic technique is done as follows. : **'Train basic model - Node pruning - Fine tuning with pruned parameters - Test'**. With pruned model, It is feasible to obtain comparable or improved accuracy compared to the original version while spending less time due to pruning.[2]. Pruning steps are being taken using the convolutional filters, with the goal of improving speed while pursuing accuracy. The primary idea behind node sparse modeling is to remove less important filters based on a pruning ratio, reducing accuracy degradation and enhancing speed using the sum of 'L2-norm'.[2]. Our work begins with a step called 'Train,' which is identical to

the one depicted on the residual network because we are using the same model as 'Wide-ResNet'. For example, in knowledge distillation, a large or student model can be 'wide-ResNet 28x10' while a small or student model can be 'wide-ResNet 28x2'. In terms of 'Node Sparsity modeling', 'Wide-ResNet 28x10' with a pruned ratio of 0.8 can be a comparable comparison target to 'Wide-ResNet 28x2'. The next stage following 'Train' is 'Node pruning', which is a critical step in the process. In each layer of our interested convolutional network(we call it as 'net'), we calculate the sum of L2-norm using 'filter ranks' functions, sort the order of the nodes based on the L2-norm result, and then set a ratio such as 0.5, 0.6 or 0.8. Through the activation function, some nodes are set to zero values as 'ReLu'. For example, if there are 10 integer values ranging from 1 to 10 and the pruning ratio is set to 0.3, three '0's and seven '1's are left after the activation function. This skill is developed using 'learned global ranking(called as LeGR)[3] in Ting-Wu Chine's paper, learning to rank convolutional filters across layers and dropping the bottom-ranked filters. Following the 'Node pruning' stage, the 'Fine tuning with pruned parameters' stage is performed, which is similar to the process of the first stage of the train. When fine tuning in a training session, the main difference is not to cut nodes all at once, but to minimize information loss through pruning and training gradually. We can then evaluate the effectiveness of 'Wide-ResNet' on CIFAR-10 and CIFAR-100 data-sets to the original model in terms of accuracy and latency. We can perform 'Node Sparsity modeling' by considering the principles of ranking filters in each layer globally.[2]

The paper are summarized as follows :

- We compare two methods : *Knowledge distillation* and *Node sparse modeling*.
- A basic model for neural network is 'Wide-ResNet' considering depth and widening factor.
- For *Knowledge distillation*, we select a teacher model as larger model and a student model as smaller model for transferring knowledge.
- For *Node sparsity modeling*, we set a pruning ratio applying to a basic model. In a way of pruning method, keeping the total pruned ratio same, pruning several times as fine-tuning are shown on results

2 Main Framework

Model compression is the technique of shrinking a machine learning model without sacrificing accuracy or performance. This can be accomplished by deleting unnecessary parameters such as weights or nodes, or by using other strategies to make the model considerably sparser. There are various types of model compression skills that may be used to make the model more efficient. We recommend two key concepts: knowledge distillation and node sparsity modeling. On an existing paper[2], two methods of 'Model compression' are shown. This section provides a main framework as well as a design on how to use its theories and procedures.

2.1 Knowledge distillation[5]

It was difficult to imagine how to transfer knowledge from large model to small model. Hinton et al. suggested leveraging probability distributions predicted by large model when small model learns to overcome this difficulty. It is known as 'Soft target', and it involves training a small model with the probability distribution predicted by a larger model.

- Soft-max output layer converting z_i into q_i :

$$q_i = \frac{exp(z_i/T)}{\sum_j exp(z_j/T)}$$

  - z_i : input value of the i-th class
  - q_i : output value of soft-max function for z_i ranging from 0 to 1
  - Temperature : Specified as 'T', Control the 'Softness' of the output probabilities.
  - Higher T : When we set high value T, we get a softer probability distribution since T=1 denotes hard label, amplifying the disparities between the logit values and spreading out the soft-max probabilities.
  - Make a 'Soft label' output from a pre-trained teacher model and distill the knowledge from a teacher model to a student model.
  - Better prediction results can be obtained by smoothing the model prediction probability distribution in the Soft-max output layer.

- Total loss(L) = $\sum_{(x,y)\in\mathbb{D}} L_{KD}\left(S\left(x,\theta_S,T\right),T\left(x,\theta_T,T\right)\right) + \lambda L_{CE}\left(\widehat{y}_S,y\right)$
  - Calculate both 'Cross Entropy Loss(L_{CE})' and 'Distillation Loss(L_{KD})'.
  - After getting 'soft label' from a teacher model, a student model is simultaneously trained and distilled by 'Total loss'.

2.2 Node Sparsity modeling[8]

Filter ranking function using L2-norm

- We compute the importance-value between nodes within each layer and use it to sparsify the model by deleting nodes with lower importance-values than the pre-set ratio.

- Pruning has two representative methods : 'Weight sparsity'[9] and 'Node sparsity'[3]

- Because some nodes are masked in each layer after pruning, it is more effective as long as performance is maintained.

- Node sparse modeling describes the situation in which specific nodes in a neural network are masked at a certain layer, and then model compression is achieved by masking nodes with a relatively low proportion of relevance.

- Because the nodes are masked within each layer, we achieve a sparse model.

- To compute the importance ratio of nodes in each layer, the expression is given as follows :

  - $I_i = \alpha_{l(i)} \|\Theta_i\|_2^2 + k_{l(i)}$
    - I_i : The importance of filter i is defined .
    - l_i : the layer index for i^{th} filter
    - $\|\Theta_i\|^2$: l_2 norms using weights θ
    - $\alpha, k \in \mathbb{R}^L$: learnable parameters where L denotes the number of layers

  - Filter ranking function[3]
    - Compute **ranking score** in each layer applying to the pre-set ratio.
    - The significance of a feature map is determined by utilizing L2 norms within each layer.
    - Bottom-ranked nodes in each layer are trimmed away due to the relevance of filter and ranking.
    - Pruning ratio : Existing filters are masked as zero values and a norm model is compressed reflecting masking layers.

The process of training 'Knowledge distillation'

Distillation modeling is used in the above 'Main Framework' to treat some of the fundamental functions and notions of generating sparse networks. Because it is impossible to ensemble all of the huge and high-performance models in a given time due to computational expenses, 'distillation' attracts attention for its efficiency. For that reason, the parameters soft label and temperature(T) are used, and a model allows knowledge distillation from a larger model to a smaller model by utilizing soft label. Then, we make an output and get 'soft label' from pre-trained larger model that we name it as 'teacher model'. In Hinton's paper[5], 'soft label' has a different meaning from 'hard label', which means a class label as the answer with the highest probability. Contrary to the hard label, soft label do not have an extreme value or answer, but they do have a probability other than the correct response of extracting more information from an image. It is taking into account the potential that it is the proper response where there is a possibility that it is not the correct answer that we expect. Knowledge distillation by Hinton et al.(2015)[5], 'tempearture(T)' is used as a hyperparameter to govern the degree of knowledge transferring from teacher model to student model. The process of 'Knowledge distillation' is shown as follows.

The process of training 'Node sparsity modeling'

A related paper called 'Weight Pruning[9] ' is one of compression skills that reduces a larger model to get a light network by deleting relatively low important weights. In this method, a complicated and larger network is first trained in the same way as 'Knowledge distillation', then trimmed based on specified standards, and lastly fine-tuned to achieve almost comparable performance with

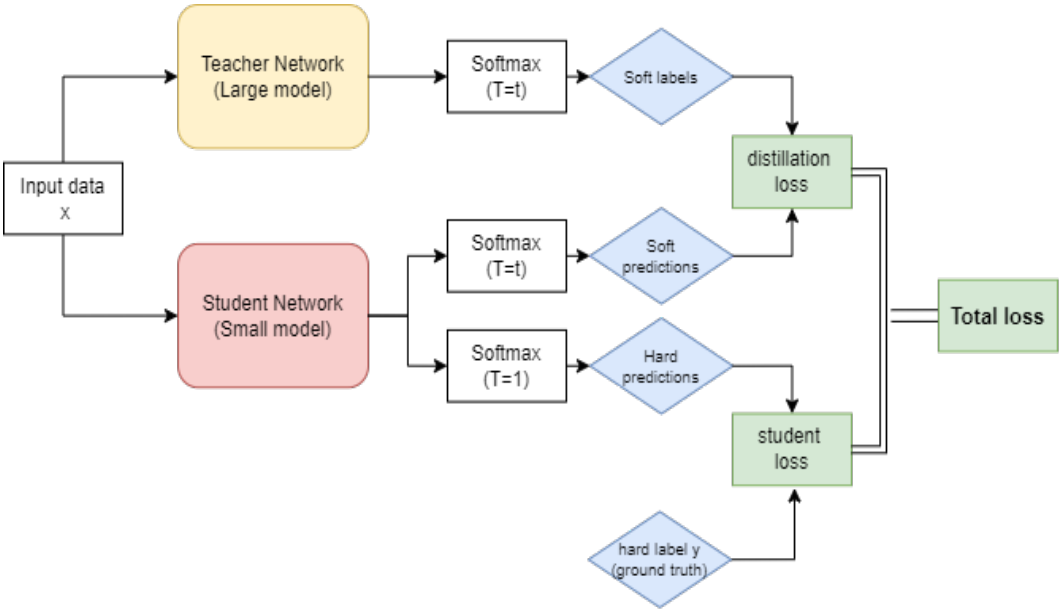

Figure 1: (Knowledge distillation model) After obtaining softmax prediction values, the training stage is completed by using distillation loss and student loss to obtain 'Total loss' for training student network.

pruned parameters. Instead of weight pruning, irrelevant nodes in each layer are deleted using the filter ranking function which is function of calculating the node's importance in a layer. This strategy can shrink the model by eliminating nodes from a fully connected layer. It is difficult to create an absolute comparison criterion when comparing the performance of knowledge distillation and node sparsity modeling, thus we will evaluate performance as test accuracy by setting several hyper-parameters through CIFAR-10 and CIFA-100 data-sets using same size models.

In the context of model compression, 'Node sparsity modeling' refers to the process of lowering the number of nodes in a neural network by pruning away less significant nodes in each layer. As previously stated, the core idea is to first select the nodes that contribute with the least importance in each layer, using filter ranking and L2 norm techniques given in the main framework [3]. Then, these nodes are pruned away from the network, reducing the overall number of nodes and making the network sparse.

Each node in a layer has a unique importance in the 'filter ranking' stage. In an experiment, a pruning ratio is set, and the bottom ranked nodes are pruned away step by step. In addition, to maintain performance, we use a fine-tuning strategy of pruning and learning nodes frequently rather than cutting a huge percentage of nodes at once. We therefore anticipate that this learning and tuning stage will aid in performance maintenance. 'Node sparsity modeling' aims to lower a neural network's computational and memory requirements while minimizing the influence on its accuracy.

It is feasible to obtain large compression without losing performance by finding and removing less important nodes. When pruning a neural network utilizing its modeling technique, there is a risk of losing existing information during the activation function pruning process. This is because prun-

ing entails eliminating specific neurons or connections from the network, which might reduce the network's capacity to represent information. If too many neurons or connections are removed, the network may lose vital information and have lower accuracy than planned. The problem and difficulty with pruning technologies is information loss. More aggressive pruning may result in a greater danger of losing crucial information, thus we must consider this while determining a pruning ratio. The process of 'Knowledge distillation'is shown as follows.

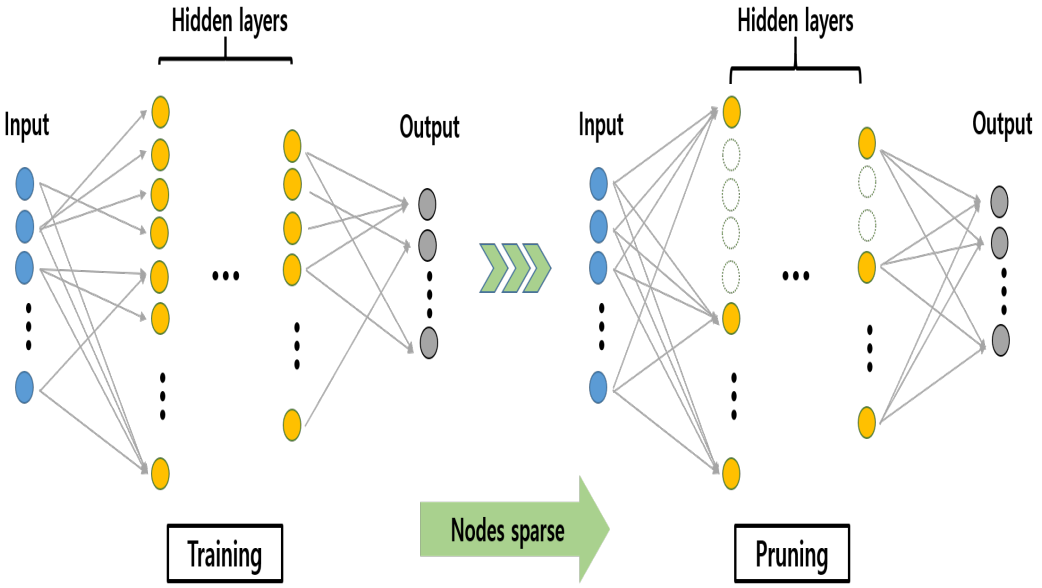

Figure 2: (Node sparsity modeling)Training step is similar with the basic model 'Wide-ResNet', but with node sparse step of pruning, filter ranking fuction by L2-norm are applied to its pruning and nodes in each layer are pruned away reflecting a pruning ratio.

3 Experiment

Our experiment makes use of Python's pytorch and the 'CIFAR-10 and CIFAR-100' data-sets to demonstrate the effectiveness of model compression in preserving original accuracy while lowering computation costs. To begin, we used 'Deep Residual Learning' to train 'Wide-ResNet'[10] to see how well our models functioned. Although the two approaches to model compression that we anticipate cannot be identical, experimental settings are carried out with minimal variation. Knowledge distillation, like model ensemble, is used to educate student models by transferring the knowledge of the teacher model, while sparsity modeling is a means of reducing model size when compared to conventional models. We can adjust the pruning ratio in the sparsity modeling all at once, but this degrades performance. Given that pruning may be applied gradually, similar to fine-tuning, a method of fine-tuning gradually pruning at a certain pace can improve the model's performance even more. Each method is effective at reducing model size, and the most effective way is determined by tuning parameters such as initial learning rate, optimizer, scheudler, and temperature, among others.

3.1 Data-sets : CIFAR-10 / CIFAR-100

In their most basic form, data-sets consist of 60,000 images divided into 10 or 100 classes. These data sets are extensively used in computer vision fields for model construction and evaluation. We used three random seeds to make the experiments more precise and relevant for this study: 10, 100, and 777. Furthermore, we used the 'transform' package to manipulate the data during training, such as 'transforming, random cropping, and horizontal flip', which improved performance. As a result, we assessed the performance of the CIFAR-10 and CIFAR-100 data-sets by analyzing the accuracy and error achieved by averaging the results from the three random seeds.

The 'Knowledge distillation' test uses the same fundamental experimental setup as 'Wide-ResNet'. However, the temperature parameter ('T') is altered at 4, 10, and 20 during distillation from the teacher model to the student model. The temperature parameter is a hyper-parameter that influences student learning by regulating the softness of the teacher's output. The temperature parameter is modified to examine how it affects the performance of the student model. By distilling knowledge from a teacher model with different temperature parameters, it was possible to observe how the softness of the teacher's output affected the student's learning. The results of these experiments shed light on how knowledge distillation might be improved for various models and applications.

The 'Node sparsity modeling' experiments provide two types of experimental settings. First, we design two distinct starting learning rates. Throughout the fine-tuning stage, we take into account both the final learning rate of 0.004 learned by the decay of the original 'Wide-ResNet' model and the previous learning rate of 0.02. We also tried with two different scheduling approaches, MultiStepLR and CosineAnnealingLR. A goal of the second experiment is to identify the ideal model that is lighter and has the least amount of performance loss by varying the parameters of the learning rate, scheduler, and making the pruning ratio the most significant aspect.

Dataset	**batch size**	**epoch**	**learning rate**	**decay epoch**
CIFAR10/100	128	200	0.1	[60,120,160]

Table 1: Data-sets and settings of hyper-parameters

3.2 Wide-ResNet model

To focus on our major model compression notions, we start with a basic model named 'Wide-ResNet' with a specific depth and widening factor. The number of widening factors in 'Node sparsity modeling' is proportional to a pruning ratio that we choose. The depth of the model reflects its complexity. A higher depth suggests that the model has more learnable parameters and is more complicated, so model compression can assist reduce model size and complexity while enhancing performance. The widening factor regulates the connection between the model's number of parameters and its accuracy. A larger network contains more parameters and can achieve higher accuracy,

but it is more computationally expensive, which is a significant issue when using model compression approaches. When working with 'Wide-ResNet', it is critical to consider both the depth and width of the model.

- 'Wide-ResNet' models used : Refer to it as 'WR' in our experiment.

  A. WR 28x10 **B.** WR 28x4

  C. WR 28x2 **D.** WR 28x1

- As a fundamental model, 'WR' on CIFAR-10 / CIFAR-100

Model	**Depth x Widening factor**	**Test Acc**	
		CIFAR 10	**CIFAR 100**
Wide Resnet	28 x 10	95.72	80.02
	28 x 4	95.38	77.24
	28 x 2	94.91	73.71
	28 x 1	93.07	68.93

Table 2: The results of the 'WR' model

- A Loss and Accuracy Graph Example : 'WR 28x10'

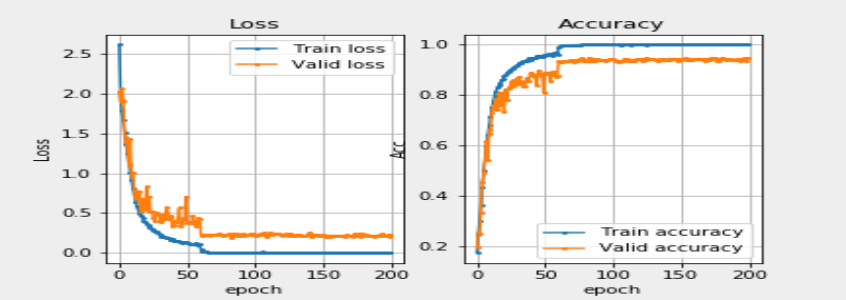

Figure 3: (WR model) 'Loss' and 'Accuracy': With 200-epochs on CIFAR-10, all models are trained to the same standards

.

3.3 Knowledge Distillation

In 'Knowledge distillation,' the teacher model works as a larger model, while the student model works as a smaller model. We employ the 'Wide-ResNet' architecture and four different distillation models. The experiment was carried out to ensure the reliability of the experimental results by using three random seeds in each model, and the result performance shown below indicates the average value of the corresponding values. The hyper-parameter 'Temperature' used in this model comes in three varieties. The best performance will be highlighted, with the others listed in the appendix.
The models we used in our experiment are listed below :

	Teacher model	Student model
1	WR 28x10	WR 28x4
2	WR 28x10	WR 28x4
3	WR 28x4	WR 28x2
4	WR 28x2	WR 28x1

- **Four types of distillation models**

  The knowledge of a teacher model is distilled to a student model using soft-max prediction value. A simple distillation notion is transferring knowledge from a larger model to a smaller one, where the teacher model is larger than the student model.

  The results are as follows: :

	Model		Test Acc (std.error)
	Teacher	**Student**	
CIFAR 10	WR 28x10	WR 28x4	95.68(0.0570)
	WR 28x10	WR 28x2	95.08 (0.0493)
	WR 28x4	WR 28x2	95.13 (0.1099)
	WR 28x2	WR 28x1	93.86 (0.0491)
CIFAR 100	WR 28x10	WR 28x4	79.53 (0.1424)
	WR 28x10	WR 28x2	76.01 (0.1472)
	WR 28x4	WR 28x2	76.31 (0.2233)
	WR 28x2	WR 28x1	71.64 (0.2303)

Table 3: The results of Knowledge distillation model

3.4 Node sparsity modeling

There are two techniques of 'Node sparsity modeling': pruning once and pruning several times such as 5 or 10, but the overall pruned ratio as a result is the same. As previously stated, we prune in the following manner: **'Train - Pruning - Train with pruned parameters - Test'**. The experiment was carried out to ensure the reliability of the experimental results by using three random seeds in each model, and the result performance shown below indicates the average of the corresponding values. The hyper-parameters 'scheduler' and 'learning rate' are associated with two distinct types. The best performance is noted, and the others are listed in the appendix.

- **Node sparsity modeling : fine tuning**

  It employs a method of fine-tuning with a predetermined pruning ratio. Instead of cutting nodes all at once, the approach gradually prunes nodes by a defined ratio at certain epochs, allowing for gradual learning while reducing the number of nodes. For hyper-parameters, the optimizers 'MultiStepLR' and 'CosineAnnealingLR' are employed, and the initial learning rates are 0.004 and 0.02.

  The following results are presented: :

	Model	Total pruning ratio	Test Acc(std.error)
CIFAR 10	WR 28x10	0.8	95.46(0.0811)
	WR 28x10	0.6	95.73(0.0458)
	WR 28x4	0.5	95.20(0.0917)
	WR 28x2	0.5	94.20(0.0784)
CIFAR 100	WR 28x10	0.8	78.38(0.2809)
	WR 28x10	0.6	79.37(0.1225)
	WR 28x4	0.5	75.96(0.0882)
	WR 28x2	0.5	72.23(0.2058)

Table 4: The results of Node sparisty modeling

3.5 Comparison of three models and their results

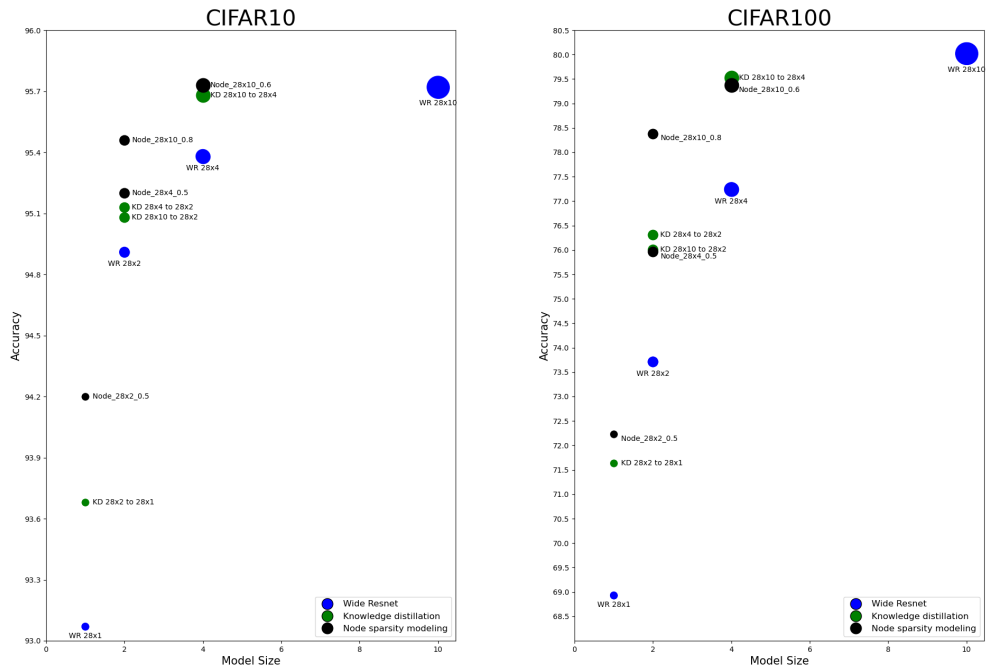

Figure 4: (Comparison of three models on CIFAR-10 and CIFAR-100 data-sets based on the same model size)

In the case of the 'Wide-ResNet' model, as the 'widening factor' increases when 'depth = 28' is fixed, computing cost rises, and test accuracy rises as well. As expected, the largest model in an experiment, WR 28x10, takes the longest time, and test accuracy appears to be the highest. We compare the performance of three models when we utilize the same number of nodes as the standard. First, taking WR 28x4 as the basic model with 1,792 nodes, we can compare the performance in 'Knowledge distillation' through the teacher model as WR 28x10 and student model as WR 28x4,

and in ‘Node Sparsity Modeling’ through a pruning ratio of 0.6 in WR 28x10. Consistent criteria were required to compare the strategies mentioned above. In the paper, depth(=28) is a fixed value, and the widening factor(=10, 4, 2, 1) varies. It is evident that the computing cost cannot be neglected, hence model compression is required to reduce model size.
In terms of test accuracy and standard error on CIFAR-10 and CIFAR-100, performance may vary based on data-sets and other types of factors, but in general, ‘Node sparsity modeling’ performed somewhat better than ‘Knowledge distillation’ under the same criteria.

4 CONCLUSION AND FUTURE WORK

Model compression strategies are used to reduce model size and computational cost while retaining model performance. Due to their small size and hardware capacity, small devices such as drones and autonomous detection robots require much lighter models, which can be accomplished by applying model compression techniques.[1]. Furthermore, applying these skills to the small size of detecting equipment necessitates image, video, and object detection, all of which must be processed in a real-world battlefield setting.High-performance deep learning models and model compression techniques that preserve the same performance can be utilized to progress the military’s object detection areas in order to keep up with the quickly changing technological trends required to continue research and development of new innovation.[11] Image classification on small size devices, in particular, is quite useful for acquiring information about the situation on the ground. Model compression techniques can thus be used to categorize and interpret images collected by small unmanned devices like drones. Similarly, to evaluate images, terrain contours, and target detection, a lightweight model that can be run on mobile devices can be employed. Signal classification in military communication is another example. Different types of signals must be classified. Deep learning models can be used to classify signals automatically. As a result, compression techniques can be used to minimize model size while increasing speed and accuracy. Furthermore, because the ’Node sparse modeling’ technique outperformed the ‘knowledge distillation’ technique in previous experiments on CIFAR-10 and CIFAR-100, it would be preferable to use it. Furthermore, ‘Node sparsity modeling’ enables good performance when utilizing larger models with a pruning ratio, as opposed to using only larger models with significantly lower speed. In the future, we need to explore more efficient and effective methods of merging multiple model compression approaches, as well as develop new compression techniques customized to the special needs of the military. More study on how to maximize the performance of compressed models under combat settings, such as limited processing resources and threatening surroundings, is required. Even though we cut the nodes with low importance, some crucial information may be lost while compressing the model, so it is critical to preserve the performance of compressed models by choosing the suitable indicators and conducting the experiment.

References

[1] Umar Asif, Jianbin Tang, and Stefan Harrer. Ensemble knowledge distillation for learning improved and efficient networks, 2019.

[2] 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.

[3] Ting-Wu Chin, Ruizhou Ding, Cha Zhang, and Diana Marculescu. Towards efficient model compression via learned global ranking. In Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pages 1518–1528, 2020.

[4] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 770–778, 2016.

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

[6] Torsten Hoefler, Dan Alistarh, Tal Ben-Nun, Nikoli Dryden, and Alexandra Peste. Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks, 2021.

[7] Weigui Li, Wenyu Sun, Yadong Zhao, Zhuqing Yuan, and Yongpan Liu. Deep image compression with residual learning. Applied Sciences, 10(11), 2020.

[8] Yigit Oktar and Mehmet Turkan. A review of sparsity-based clustering methods. Signal Processing, 148, 02 2018.

[9] Md Aamir Raihan and Tor M. Aamodt. Sparse weight activation training, 2020.

[10] Sergey Zagoruyko and Nikos Komodakis. Wide residual networks. In Edwin R. Hancock Richard C. Wilson and William A. P. Smith, editors, Proceedings of the British Machine Vision Conference (BMVC), pages 87.1–87.12. BMVA Press, September 2016.

[11] X. Zeng and T. R. Martinez. Using a neural network to approximate an ensemble of classifiers. Neural Process. Lett., 12(3):225–237, dec 2000.