Discrepancy-Based Knowledge Distillation for Image Classification Restoration

Zihao Mo*, Yejiang Yang*†, Weiming Xiang*
*School of Computer and Cyber Sciences, Augusta University, Augusta, GA, 30901, USA
†School of Electrical Engineering, Southwest Jiaotong University, Chengdu, China

***Abstract*—This paper introduces knowledge distillation to restore compressed neural networks on image classification tasks. Rather than focusing on accuracy, it adopts discrepancy as the main metric for compressed neural network performance evaluation. We modify the hard target in the knowledge distillation to address the discrepancy issue during restoration. We utilize MNIST and CIFAR10 datasets to generate compressed neural networks and restore networks using our knowledge distillation method to outperform those using cross-entropy, achieving up to a 5% reduction in performance loss. Furthermore, we discuss the impact of the choice of hyperparameters on discrepancy restoration. Our new knowledge distillation approach brings up a discrepancy-based restoration method that improves the compressed neural network discrepancy performance.**

***Index Terms*—Knowledge Distillation, Network Restoration, Discrepancy, Compressed Network.**

I. Introduction

Image recognition has always been a popular topic in neural networks, with many foundational network structures proposed to tackle this problem, such as feedforward neural networks (FNN) [1] and convolutional neural networks (CNN) [2]. Notable examples include AlexNet [3], VGGNet [4], ResNet [5], and InceptionNet [6], each demonstrating significant achievement in accuracy and performance. Due to the excellent performance of neural networks, they have been applied in various fields, with one major direction being applications on edge devices. While those neural networks perform well, they tend to have large model sizes, high computational demands, and require significant memory, which makes them not the best choices for edge applications. Due to the hardware limitations of edge devices, such as limited processing power, memory constraints, and lower battery life, the deployment of large neural networks is restricted. To cope with these restrictions, various compression methods have been proposed.

Although compression methods can reduce the model complexity, compressed neural networks often exhibit reduced accuracy [7], lower robustness to input variations and adversarial attacks [8], limited capacity for handling complex tasks, and often lack flexibility for unseen data. It is clear that merely compressing neural networks is not sufficient, a compressed neural network restoration approach is necessary. Compressed neural network restoration aims to repair the compressed neural network towards its reference one instead of training a new one. However, only a handful of methods have been proposed. Yang et al. propose a repair method based on the distance to the safe domain to update the compressed neural network parameters, modifying the neural network's reachable domain until it is inside the safe domain [9]. Mo et al. use the reachability analysis method to calculate the discrepancy between the reference neural network and the compressed one and employ the retraining method to lower the discrepancy, restoring the compressed neural network performance [10].

There has been considerable emphasis on improving the accuracy of compressed neural networks, but other key aspects are often neglected. These specifications include latency [11], energy efficiency [12], model size [13], and discrepancy [14], all of which are crucial for practical applications on edge devices. Focusing solely on accuracy while neglecting other neural network properties has hidden potential threats, especially in Cyber-Physical Systems (CPS) applications, where the interaction between computational processes and physical components is critical. For example, learning-enabled CPS with a compressed neural network controller may lead to severe safety issues due to neural network compression [15]. In environment monitor devices, energy inefficiency can result in shorter battery life, compromising the device's reliability during long-term monitoring [16]. In power plants, large model sizes may exceed the memory capacity of embedded systems, leading to system failures [17]. Ignoring those factors can lead to inefficient use of resources and degraded performance.

To address these issues, we propose a novel knowledge distillation method to restore compressed neural networks. We apply the reachability-based method to compute compressed neural network discrepancies and use discrepancy as a metric to repair the compressed neural network by lowering the discrepancy. We modify the hard target loss part from the original knowledge distillation method to address the discrepancy issue. This approach aims to enhance the overall performance of the neural networks, ensuring they meet the prescribed specifications for practical use. We utilize MNIST and CIFAR10 datasets to compare the performance of the original knowledge distillation method and our discrepancy-based knowledge distillation methods. We compare our method with the direct retraining method to show our method guarantees compressed neural network performance.

The structure of this paper is as follows. Section II introduces background and related works. In Section III, we discuss the modification of hard target loss in knowledge distillation. Section IV evaluates the developed approach via experiments. Finally, Section V presents the conclusions.

1946-0759/24/$31.00 ©2024 IEEE
DOI 10.1109/ICMLA61862.2024.00159

II. Related Works

A. Reachability Analysis

Neural network verification aims to ensure the reliability and safety of neural networks through a sound and complete process. As some neural network applications require high robustness against adversarial inputs, compliance with strict specifications, and safe behavior under all possible conditions, it is crucial to perform neural network verification. Reachability analysis is a powerful tool for neural network verification, which uses mathematical techniques to determine the set of all possible states that a system can reach with a given set of inputs. By exploring the range of possible inputs, reachability analysis helps identify all potential outputs, detect unsafe regions, and validate given specifications. There are many proposed reachability analysis methods, such as star set [18], zonotope [19], polytope [20], and face lattice [21]. These methods develop different set representations to handle complex and redundant sets generated during the analysis and try to accelerate the whole process. Even though those methods face challenges in handling high dimensionality inputs, non-linear functions, and complex network structures, they are still powerful and efficient neural network verification approaches.

B. Equivalence Verification

As a branch of neural network verification, equivalence verification also aims to ensure the neural networks' correctness and reliability. However, unlike other neural network verification problems that focus more on the characteristics of a single neural network, equivalence verification pays more attention to the relationship between the two neural networks. It aims to solve the challenges in the deployment and maintenance scenario, such as model updates for performance improvement, cross-platform consistency for platform migration, and especially, model compression for edge device deployment. Equivalence verification involves rigorously proving and evaluating whether the outputs between two neural networks are sufficiently similar or even the same for a given input domain. So far, some different approaches have been proposed for equivalence verification. Xiang et. al. propose approximate bisimulation relations to indicate the distance between two neural networks for model reduction [22]. Mo et. al. propose a reachability-based neural network discrepancy computation approach, giving a concrete value to characterize the maximum output difference between the two neural networks after compression [14]. As a branch of neural network verification, except for the challenges encountered in general verification problems, equivalence verification has other challenges, such as structural differences between two neural networks and acceptable approximate equivalence levels.

III. Discrepancy-Based Knowledge Distillation

As described in the knowledge distillation paper [23] and application in [24], the knowledge embedded in an ensemble of models can be transferred into a single model, where the single model has similar performance as the ensemble of models. Compared to other model training methods, the knowledge distillation method proposes an important concept, referred to as the teacher-student model, which enables the knowledge transfer from a powerful but computationally intensive model to a smaller and more efficient model. To accomplish the goal that restoring the compressed neural network to have a similar output as the reference neural network, knowledge distillation is employed to repair the compressed neural network, where the compressed neural network is viewed as student model $\mathbb{N}_S$, and the reference neural network is the teacher model $\mathbb{N}_T$.

Unlike the general training process, which uses ground truth labels as the only targets, knowledge distillation adopts the soft targets produced by the teacher model to help the student model mimic the teacher model's decision. The loss function of the soft target L_S is defined below

$$L_S = -\textstyle\sum_i p_i \cdot \log(\frac{q_i}{p_i}), \tag{1}$$

where

$$p_i = \frac{\exp(y_{T_i}/T)}{\sum_j \exp(y_{T_j}/T)},\ q_i = \frac{\exp(y_{S_i}/T)}{\sum_j \exp(y_{S_j}/T)}. \tag{2}$$

The softmax function is applied on the neural network output to convert the teacher model logit y_{T_i} and student model logit y_{S_i} into probability p_i, q_i for each label i. T is the temperature that controls the sharpness of probability distribution over classes. A higher T produces a softer distribution. p_i and q_i are the softmax outputs of the teacher and student models at temperature T, respectively. The loss function of hard targets L_H using cross-entropy is defined below

$$L_H = -\textstyle\sum_i c_i \cdot \log(q_i), \tag{3}$$

where

$$c_i = \begin{cases} 1, \text{ corresponding label outputs} \\ 0, \text{ otherwise} \end{cases}, \tag{4}$$

and q_i is the softmax output of the student model at temperature T. Thus, the final loss function is the combination of the soft targets and hard targets as below

$$L = \alpha L_S + (1-\alpha) L_H, \tag{5}$$

where the α is the hyper-parameters to balance the influence of soft targets and hard targets.

The hard targets part using the cross-entropy on corresponding outputs in the knowledge distillation indicates that the student model mimics the teacher model's decision strategy and gives higher output probability on the correct class and lower down probability on other classes. However, this strategy isn't consistent with the goal of restoring the student model to give similar outputs as the teacher model, as the student model may pay more attention to the correct class and less to the correlation among all classes. To address the similarity in the output of all classes, we modify the hard targets part with the discrepancy result in $\tilde{\delta}_{max}$, which can be obtained from reachability-based discrepancy computation as shown in [14]. The hard targets part is modified below

$$\hat{\mathbf{y}}_T = \mathbf{y}_T - \beta \tilde{\delta}_{max}, \tag{6}$$

where $\mathbf{y}_T$ is the output vector of the teacher model, and β is a tuning parameter to avoid overfitting issues, normally set as 0.5. The loss function of hard targets is switched to mean squared error instead of cross-entropy loss as mean squared error contains more information for all classes. Thus, the modified loss function for hard targets is

$$\hat{L}_H = -\frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{y}}_T - \mathbf{y}_S\|\,, \tag{7}$$

where $\mathbf{y}_S$ is the output vector of the student model. As we introduce the discrepancy to the teacher model logits, the soft targets part is modified as follows:

$$\hat{L}_S = -\textstyle\sum_i \hat{p}_i \cdot \log(\frac{q_i}{\hat{p}_i}), \tag{8}$$

where

$$\hat{p}_i = \frac{\exp(\hat{y}_{T_i}/T)}{\sum_j \exp(\hat{y}_{T_j}/T)}. \tag{9}$$

Thus, the final loss function is updated as follows

$$\hat{L} = \alpha \hat{L}_S + (1-\alpha)\hat{L}_H. \tag{10}$$

Algorithm 1 summarizes the discrepancy-based knowledge distillation on compressed neural network restoration. For each epoch, it computes the discrepancy between the updated student and teacher models and updates the loss function for student model parameter. If the discrepancy drops below a prescribed threshold, the restoration process is finished, and a timeout counter is adopted to avoid endless updates.

Algorithm 1: Discrepancy-Based Knowledge Distillation for Neural Network Restoration

```
input : Teacher model $\mathbb{N}_T$, student model $\mathbb{N}_S$,
        discrepancy threshold $d$
output: Restored student model $\hat{\mathbb{N}}_S$
1  $\tilde{\delta}_{max} \leftarrow$ discrepancy($\mathbb{N}_T, \mathbb{N}_S$) while True do
2      $\hat{\mathbf{y}}_T \leftarrow \mathbf{y}_T - 0.5 * \tilde{\delta}_{max}$
3      $\hat{p}_i \leftarrow \frac{\exp(\hat{y}_{T_i}/T)}{\sum_j \exp(\hat{y}_{T_j}/T)}$
4      $q_i \leftarrow \frac{\exp(y_{S_i}/T)}{\sum_j \exp(y_{S_j}/T)}$
5      $\hat{L}_S \leftarrow -\sum_i \hat{p}_i \cdot \log(\frac{q_i}{\hat{p}_i})$
6      $\hat{L}_H \leftarrow -\frac{1}{N}\sum_{i=1}^{N} \|\hat{\mathbf{y}}_T - \mathbf{y}_S\|$
7      $\hat{L} \leftarrow \alpha\hat{L}_S + (1-\alpha)\hat{L}_H$
8      $\mathbb{N}_S \leftarrow$ loss backward($\mathbb{N}_S, \hat{L}$)
9      $\tilde{\delta}_{max} \leftarrow$ discrepancy($\mathbb{N}_T, \mathbb{N}_S$)
10     if $\|\tilde{\delta}_{max}\| \leq d$ or timeout then
11         $\hat{\mathbb{N}}_S \leftarrow \mathbb{N}_S$
12         break
13     end
14 end
15 return $\hat{\mathbb{N}}_S$
```

IV. Experiments

A. Datasets

For the experiments, we utilize the MNIST and CIFAR10 datasets for evaluations. The MNIST dataset includes a large number of hand-written digits, and each image is a $28*28*1$ grayscale image. The CIFAR10 dataset contains 10 different classes of images, and each image is a $32*32*3$ RGB image. These two datasets are popularly used benchmarks for image recognition.

B. Neural Network Implementation

For the experiments, we implement three different neural networks as the teacher model for different datasets.

- **MNIST FNN:** It is a feedforward neural network with a total of 4 linear layers, each of which is followed by the ReLU function except for the output layer.
- **MNIST CNN:** It is a convolution neural network with two convolution layers, a max pooling layer, and two linear layers. Except for the output layer, each layer is followed by the ReLU function.
- **CIFAR10 VGG:** It is a deep convolution neural network with 16 layers [4]. It contains five groups of convolution layers, where the first two groups have two layers, and the last three groups contain three layers. A max pooling layer is added between two convolution groups. Three linear layers follow the last convolution group. The ReLU function follows each layer.

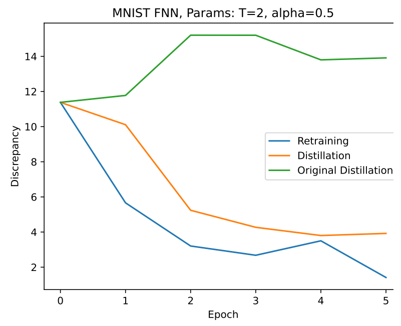
We adopt the quantization method for each student model to compress each teacher model without changing the network structure. Table I shows that the teacher model and the student model have the same number of parameters, as we apply the same network structure. After quantization, all student models' size drops down to one-quarter the size of their teacher models while maintaining a high accuracy.

TABLE I: Overview of neural network implementation.

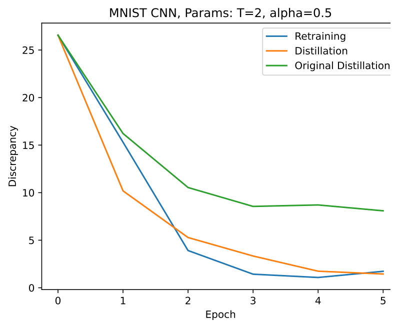
		Parameters	Size (KB)	Accuracy (%)
MNIST	Teacher	472,042	1,847	96
FNN	Student	472,042	469	96
MNIST	Teacher	1,199,882	4,690	98
CNN	Student	1,199,882	1,179	98
CIFAR10	Teacher	7,879,818	30,791	80
VGG	Student	7,879,818	7,725	80

C. Discrepancy Computation

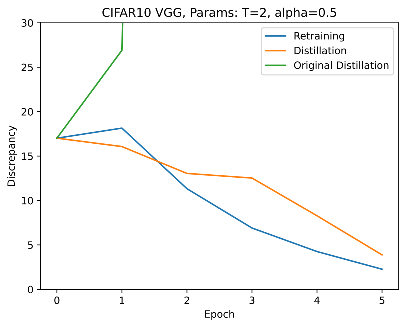
As our method is based on discrepancy results, we employ the exact reachability analysis method to calculate the maximum vector $\tilde{\delta}_{max}$. Exact reachability analysis guaranteed the maximum discrepancy between the two neural networks with a given input domain without any conservativeness, unlike over-approximation, which provides conservative results due to approximation on non-linear function. For simplicity, we randomly choose an image from both datasets as our experiment sample and set a pixel on the sample with a ± 0.05 variation after normalization and standardization.



(a) MNIST FNN



(b) MNIST CNN



(c) CIFAR10 VGG

Fig. 1: Discrepancy performance restoration comparison among different methods.

TABLE II: Discrepancy restoration performance

Model	Method	Ori. Disc.	Re. Disc.	Reduction
MNIST FNN	Disc. KD	11.3810	3.9215	34.5%
	Ori. KD	11.3810	13.9124	122.2%
	Retrain	**11.3810**	**1.4141**	**12.4%**
MNIST CNN	Disc. KD	**26.5594**	**1.4474**	**5.4%**
	Ori. KD	26.5594	8.0912	30.5%
	Retrain	26.5594	1.7304	6.5%
CIFAR10 VGG	Disc. KD	17.0158	3.8791	22.8%
	Ori. KD	17.0158	1092.3517	$\geq$1000%
	Retrain	**17.0158**	**2.2670**	**13.3%**

D. *Knowledge Distillation Evaluation*

In this experiment, we mainly focus on comparing the discrepancy results for employing different restoration methods. Here, we compare our discrepancy-based knowledge distillation method with the original knowledge distillation method on the discrepancy restoration performance, and we take the retraining method proposed in [10] as the reference, which has been proved effective in performance restoration. For the hyperparameters in knowledge distillation, we generally set temperature $T = 2$ and $\alpha = 0.5$. We run for five epochs to observe the trends of different methods. Fig.1 shows the result for these methods' performance on discrepancy restoration. Only in Fig.1b the original knowledge distillation method can help reduce the discrepancy after several epochs. Besides, the original knowledge distillation method can't help restore the neural network's discrepancy but may even increase the discrepancy, which is not expected. On the opposite, our discrepancy-based knowledge distillation method can effectively reduce the discrepancy. Compared to the retraining method, our method can reduce the discrepancy to as low as the retraining method. Table II shows the detail of discrepancy restoration performance for different restoration methods. "Disc. KD" means the discrepancy-based knowledge distillation, "Ori. KD" means the original knowledge distillation, "Ori. Disc." means the originally calculated discrepancy, "Re. Disc." means the discrepancy after restoration, and "Reduction" means the ratio of restored discrepancy to the original discrepancy. "Ori. KD" incredibly increases the discrepancy over ten times after the restoration process, which goes against our purpose. It is interesting to notice that our "Disc. KD" can perform better than retraining in the MNIST CNN model, which shows the potential for further improvement in discrepancy restoration purposes.

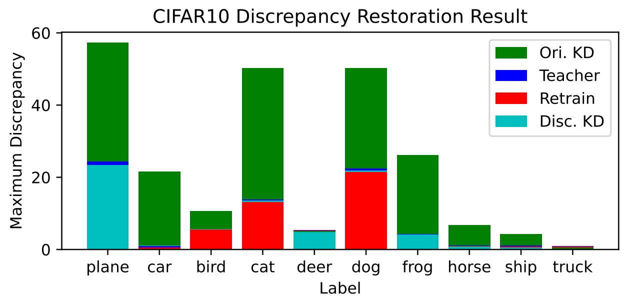
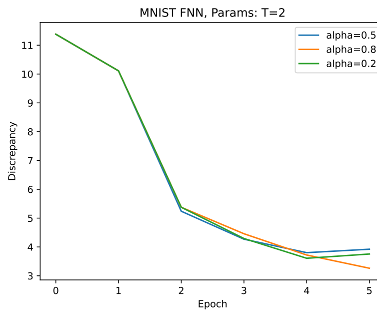
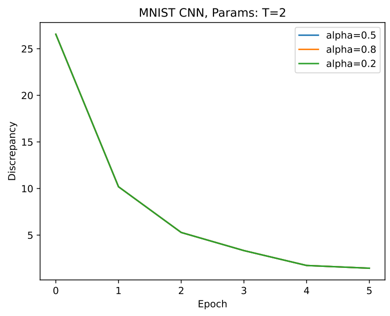


Fig. 2: Output result comparison among different restoration methods with CIFAR10 sample.

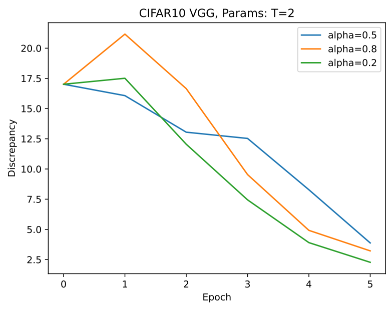
As the "Ori. KD" provides a very high discrepancy result after the restoration process, we want to dig a little deeper to find out the reason. We input the sample to the three student models and the teacher model and plot the comparison for all label outputs in Fig. 2. The results of the teacher model are the target outputs. It is clear that the "Retrain" and the "Disc. KD" methods can narrow down the difference between the student model's output and the teacher model's output. However, the "Ori. KD" methods increase the difference between them. Although the logit at the correct label is still the highest one, the logit itself rises too high. As the original knowledge distillation uses Eq. (3) as the hard target part, it raises too much attention on the correct label, which leads to the increment of the correct label logit. Meanwhile, Eq. (1) updates the student model to provide a similar output distribution as the teacher model, which forces all other label logits to increase as the correct label logit rises too high. To testify, we evaluate the student model's accuracy performance. "Ori. KD" drops the accuracy down to 66%, while the "Disc. KD" keeps the accuracy at 70%, which further proves our discrepancy-based knowledge distillation method outperforms the original knowledge distillation in neural network restoration.



(a) MNIST FNN

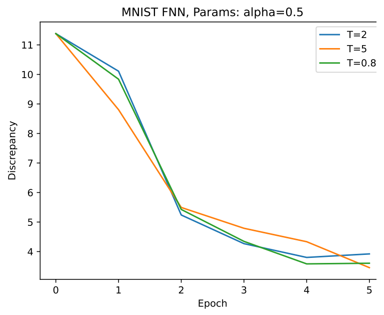


(b) MNIST CNN

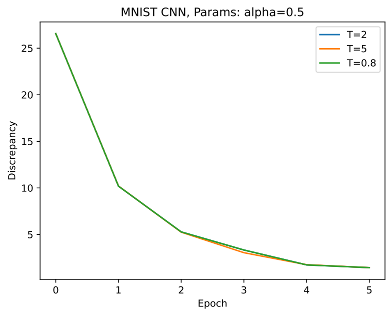


(c) CIFAR10 VGG

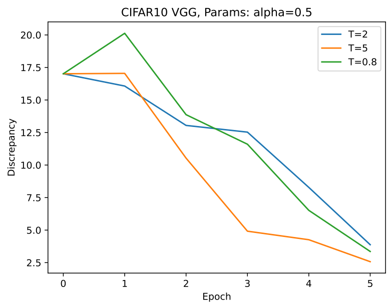
Fig. 3: Discrepancy performance restoration comparison with different α.



(a) MNIST FNN



(b) MNIST CNN



(c) CIFAR10 VGG

Fig. 4: Discrepancy performance restoration comparison with different temperature T.

E. Hyperparameters Tuning for Discrepancy Based Knowledge Distillation

As shown in section three, the knowledge distillation method has two hyperparameters that are not pre-set. The combination of different temperatures T and α can affect the loss function and further affect the optimization of neural networks. In the above experiment, the temperature is set to 2, and α is set to 0.5 as an initial setting. To further explore the influence of different choices of T and α, we design two controlled variable experiments to have a glance at the effect of temperature T and α.

1) α-Tuning Experiment: To explore the effect of α, we control the temperature $T = 2$ and test it with three different α, where $\alpha = 0.2,\ 0.5$, and 0.8. We test our three models and run them with five epochs to gain insight into the discrepancy reduction trend. The results are shown in Fig. 3 and Table III. From the results, we notice that in small models, different α have almost the same reduction slope for the first few epochs. In Fig. 3a, $\alpha = 0.8$ has a trend of continued decline, but $\alpha = 0.5$ and $\alpha = 0.2$ are more inclined to start converging. In Fig. 3b, all three α give the same result. After inspection, we find that the hard target loss far outperforms the soft target loss, which results in that the updated direction of the student model mainly relies on the hard target part. In Fig. 3c, $\alpha = 0.2$ and $\alpha = 0.8$ may increase the discrepancy at the first epoch but lately drop rapidly and reach a lower discrepancy compared to $\alpha = 0.5$. After 5 epochs, all of these three α still have a downward trend to reduce the discrepancy further.

From this α experiment, we can briefly conclude that the choice of α will affect the discrepancy descent rate. A lower α brings a more aggressive decline curve but converges early, and a higher α allows a slower but steadier decline. However, it requires the soft target loss and hard target loss to be in the same order of magnitude.

2) T-Tuning Experiment: To explore the effect of temperature T, we control the $\alpha = 0.5$ and test it with three different T, where $T = 0.8,\ 2$, and 5. Same as the α-tuning process, the results are shown in Fig. 4 and Table III. In Fig. 4a, $T = 2$ and $T = 0.8$ have similar decline curves, both of which have a sharp drop at the second epoch and start converging at later epochs. While $T = 5$ performs differently, where the discrepancy reduces steadily and goes lower than the other two. In Fig. 4b, $T = 2$ and $T = 0.8$ have the same curve, and the $T = 5$ has a very similar curve, which means that in this model, the hard target loss far exceeds the soft target loss, which led almost the same update direction. But, it is interesting to notice that the $T = 5$ is a little different, which brings us a hint that a higher temperature will increase the soft target loss ratio in the total loss and decrease the effect from the hard target part. In Fig. 4c, taking $T = 2$ as a reference,

TABLE III: Hyperparameters tuning result for discrepancy-based knowledge distillation

Model	Ori. Disc.	$\alpha = 0.5, T = 2$	$\alpha = 0.2, T = 2$	$\alpha = 0.8, T = 2$	$\alpha = 0.5, T = 0.8$	$\alpha = 0.5, T = 5$
MNIST FNN	11.3810	3.9215	3.7550	**3.2620**	3.6056	3.4563
MNIST CNN	26.5594	**1.4474**	**1.4474**	**1.4474**	**1.4474**	1.4488
CIFAR10 VGG	17.0159	3.8791	3.2217	**2.2759**	3.3584	2.5741

$T = 0.8$ has a noticeable rise at the first epoch and later drops down to a lower discrepancy. $T = 5$ still gives a steady decline curve and reaches the lowest point among these three curves. From these three model comparisons, discrepancy converges earlier with a simple neural network, and a more complicated neural network seems to restore to a lower discrepancy level, as MNIST FNN seems to start converging while CIFAR10 VGG keeps decreasing.

From this temperature T experiment, we can briefly summarize the effect of the temperature T in the restoration process. A higher temperature T helps increase the soft target loss proportion in the total loss and provides a steady decline curve. A lower T can bring a sharp decline curve and drop down to the low discrepancy point earlier. As explained in [23], a higher temperature T provides a soft and smooth output distribution, fuzzing the relation between label logits. A lower temperature T, especially when $T < 1$, addresses each label, bringing more attention to each label logit.

V. Conclusions

This paper introduces our discrepancy-based knowledge distillation method for neural network restoration in image recognition problems. Compared to the classic knowledge distillation, we apply discrepancy results on teacher logits and modify hard target loss to be consistent with the target of lowering the student model's discrepancy. The experiment proves that our knowledge distillation outperforms the original one in discrepancy restoration performance. And we discuss the effect of α and temperature T. With different combinations of α and T, it has great potential to have better discrepancy restoration performance on image recognition problems and outperform the retraining method.

Acknowledgment

This work was supported by the National Science Foundation, under NSF CAREER Award 2143351, NSF CNS Award 2223035, and NSF IIS Award 2331938.

References

[1] G. Arulampalam and A. Bouzerdoum, "A generalized feedforward neural network architecture for classification and regression," *Neural networks*, vol. 16, no. 5-6, pp. 561–568, 2003.

[2] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," *Advances in Neural Information Processing Systems*, vol. 25, 2012.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 770–778, 2016.

[6] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, pp. 448–456, 2015.

[7] Y. Zhang, W. Ding, and C. Liu, "Summary of convolutional neural network compression technology," in *2019 IEEE International Conference on Unmanned Systems (ICUS)*, pp. 480–483, 2019.

[8] A. W. Wijayanto, J. J. Choong, K. Madhawa, and T. Murata, "Towards robust compressed convolutional neural networks," in *2019 IEEE International Conference on Big Data and Smart Computing (BigComp)*, pp. 1–8, IEEE, 2019.

[9] X. Yang, T. Yamaguchi, H.-D. Tran, B. Hoxha, T. T. Johnson, and D. Prokhorov, "Neural network repair with reachability analysis," in *International Conference on Formal Modeling and Analysis of Timed Systems*, pp. 221–236, Springer, 2022.

[10] Z. Mo, Y. Yang, S. Lu, and W. Xiang, "Compression repair for feedforward neural networks based on model equivalence evaluation," *arXiv preprint arXiv:2402.11737*, 2024.

[11] S. Yao, J. Li, D. Liu, T. Wang, S. Liu, H. Shao, and T. Abdelzaher, "Deep compressive offloading: Speeding up neural network inference by trading edge computation for network latency," in *Proceedings of the 18th Conference on Embedded Networked Sensor Systems*, pp. 476–488, 2020.

[12] S. Han, X. Liu, H. Mao, J. Pu, A. Pedram, M. A. Horowitz, and W. J. Dally, "Eie: Efficient inference engine on compressed deep neural network," *ACM SIGARCH Computer Architecture News*, vol. 44, no. 3, pp. 243–254, 2016.

[13] R. Pilipović, P. Bulić, and V. Risojević, "Compression of convolutional neural networks: A short survey," in *2018 17th International Symposium INFOTEH-JAHORINA (INFOTEH)*, pp. 1–6, IEEE, 2018.

[14] Z. Mo and W. Xiang, "Maximum output discrepancy computation for convolutional neural network compression," *Information Sciences*, vol. 665, p. 120367, 2024.

[15] W. Xiang, H.-D. Tran, J. A. Rosenfeld, and T. T. Johnson, "Reachable set estimation and safety verification for piecewise linear systems with neural network controllers," in *2018 Annual American Control Conference (ACC)*, pp. 1574–1579, IEEE, 2018.

[16] T. Reddy, S. P. RM, M. Parimala, C. L. Chowdhary, S. Hakak, W. Z. Khan, *et al.*, "A deep neural networks based model for uninterrupted marine environment monitoring," *Computer Communications*, vol. 157, pp. 64–75, 2020.

[17] A. Saeed and A. Rashid, "Development of core monitoring system for a nuclear power plant using artificial neural network technique," *Annals of Nuclear Energy*, vol. 144, p. 107513, 2020.

[18] H.-D. Tran, D. Manzanas Lopez, P. Musau, X. Yang, L. V. Nguyen, W. Xiang, and T. T. Johnson, "Star-based reachability analysis of deep neural networks," in *Formal Methods–The Next 30 Years: Third World Congress, FM 2019, Porto, Portugal, October 7–11, 2019, Proceedings 3*, pp. 670–686, Springer, 2019.

[19] G. Singh, T. Gehr, M. Mirman, M. Püschel, and M. Vechev, "Fast and effective robustness certification," *Advances in Neural Information Processing Systems*, vol. 31, pp. 10802–10813, 2018.

[20] G. Singh, T. Gehr, M. Püschel, and M. Vechev, "An abstract domain for certifying neural networks," *Proceedings of the ACM on Programming Languages*, vol. 3, pp. 1–30, 2019.

[21] X. Yang, H.-D. Tran, W. Xiang, and T. Johnson, "Reachability analysis for feed-forward neural networks using face lattices," *arXiv preprint arXiv:2003.01226*, 2020.

[22] W. Xiang and Z. Shao, "Approximate bisimulation relations for neural networks and application to assured neural network compression," in *2022 American Control Conference (ACC)*, pp. 3248–3253, IEEE, 2022.

[23] G. Hinton, O. Vinyals, and J. Dean, "Distilling the knowledge in a neural network," *arXiv preprint arXiv:1503.02531*, 2015.

[24] R. Gaire, S. Tabrizchi, and A. Roohi, "Encode: Enhancing compressed deep learning models through feature—distillation and informative sample selection," in *2023 International Conference on Machine Learning and Applications (ICMLA)*, pp. 633–638, IEEE, 2023.