Journal of Computer Science Research
https://journals.bilpubgroup.com/index.php/jcsr

ARTICLE

Research and Model Library Construction in Teacher-Student Learning Architectures for Knowledge Transfer

Jiaxiang Chen[*], *Yuhang Ouyang, Zheyu Li*

College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China

ABSTRACT

This paper summarizes and replicates multiple classical and cutting-edge knowledge transfer methods, including Factor Transfer (FT), Knowledge Distillation (KD), Deep Mutual Learning (DML), Contrastive Representation Distillation (CRD), and Born-Again Self-Distillation (BSS). Additionally, we studied three advanced knowledge transfer methods: Relational Knowledge Distillation (RKD), Similarity-Preserving (SP), and Attention-based Feature Distillation (AFD), successfully replicating an optimized version of KD, namely RKD. Based on these methods, a flexible model library was constructed in Pycharm, allowing the quick integration of multiple knowledge transfer strategies. The experimental results are visualized through a user-friendly interface, enabling intuitive comparisons of model training speed and performance across different methods. This research provides valuable insights into the challenge of building a reusable framework that efficiently integrates various knowledge transfer strategies into deep neural networks.

Keywords: Knowledge transfer; Teacher-student architecture; Model library; Visualization; Educational frameworks

1. Introduction

With the rapid development of deep learning, knowledge transfer has gained significant attention as a technique to improve model generalization and training efficiency. By transferring knowledge from a teacher model to a student model, knowledge transfer methods can enhance the performance of smaller models and improve training outcomes, especially in data-limited scenarios. However, the challenge

*CORRESPONDING AUTHOR:
Jiaxiang Chen, College of Electrical Engineering, Zhejiang University, Hangzhou 310027, China; Email: 2530807552@qq.com

ARTICLE INFO
Received: 24 September 2024 | Revised: 15 October 2024 | Accepted: 18 October 2024 | Published Online: 30 October 2024
DOI: https://doi.org/10.30564/jcsr.v6i4.7344

CITATION
Chen, J., Ouyang, Y., Li, Z., 2024. Research and Model Library Construction in Teacher-Student Learning Architectures for Knowledge Transfer. Journal of Computer Science Research. 6(4): 73–81. DOI: https://doi.org/10.30564/jcsr.v6i4.7344

of quickly integrating multiple knowledge transfer strategies in real-world applications remains unresolved[1]. This paper addresses this challenge by replicating classic and advanced knowledge transfer methods and constructing a flexible model library, providing novel insights into solving this problem[2].

2. Methodology

A. *Summary and Replication of Knowledge Transfer Methods:*

KD is relatively simple and transfers knowledge through soft labels. It uses soft labels from a pre-trained teacher model to guide the student model, making it suitable for scenarios where simple and effective knowledge transfer is required.

1) Training the Teacher Model (ResNet-110)

Goal: We need to train a high-performance teacher model, ResNet-110. This model will be trained on a large dataset (e.g., CIFAR-100) to learn rich features and patterns.

Steps:

Obtain the training dataset (e.g., CIFAR-100), including images and their corresponding labels.

Use this data to train the ResNet-110 model, optimizing the model parameters to achieve the highest possible accuracy.

After training is completed, save the trained ResNet-110 model.

Simplified Explanation:

Imagine ResNet-110 as an experienced teacher who has spent years studying and practicing, mastering a vast amount of knowledge and skills.

2) Generating Soft Targets

Goal: Using the trained teacher model (ResNet-110), generate soft targets for each training sample. Soft targets are the predicted probability distributions produced by the teacher model, rather than just the correct answers.

Steps:

Use the trained ResNet-110 model to make predictions on the training data.

For each input sample, obtain the probability distribution output by the model (i.e., soft targets). These distributions represent the model's confidence levels for each class.

Simplified Explanation:

This step is like asking the experienced teacher to give detailed explanations for each question, rather than simply providing the correct answer. For example, the teacher not only tells you that the cat is a cat but also explains how similar the cat is to dogs, rabbits, and other animals.

3) Training the Student Model (ResNet-20)

Goal: Train a smaller student model, ResNet-20, to learn from the teacher model's soft targets and achieve similar high performance.

Steps:

Initialize a ResNet-20 model.

Train the ResNet-20 model using the training data along with the soft targets generated by the teacher model.

During training, the student model attempts to mimic the probability distribution output of the teacher model, rather than just matching the hard labels of the training data.

Optimize the student model's parameters so that its outputs closely match the soft targets of the teacher model, while also considering the hard labels of the training data.

Simplified Explanation:

This step is like a novice student (ResNet-20) not just memorizing the answers, but also understanding the essence of the problem and the solution process through the teacher's (ResNet-110) detailed explanations. This way, the student learns more comprehensive and in-depth knowledge.

FT has a slightly higher complexity, requiring the extraction and transfer of intermediate layer features. Knowledge is transferred directly through the feature representations of intermediate layers, making it suitable for scenarios where the student model needs to learn rich feature representations.

1) Training the Teacher Model (ResNet-110)

Goal: We need to train a high-performance teacher model, ResNet-110. This model will be trained on a large dataset (e.g., CIFAR-100) to learn rich features and patterns.

Steps:

Obtain the training dataset (e.g., CIFAR-100), including images and their corresponding labels.

Train the ResNet-110 model using this data, optimizing the model parameters to achieve the highest possible accuracy.

After training, save the trained ResNet-110 model.

Simplified Explanation:

This step is like training an experienced teacher (ResNet-110) by letting them study a vast collection of books and exercises, mastering a large amount of knowledge and skills.

2) Extracting Intermediate Representations (Factors)

Goal: Extract feature representations (factors) from the intermediate layers of the trained teacher model, ResNet-110. These factors contain rich semantic information that is more detailed than the final classification output.

Steps:

Select one or more intermediate layers of the teacher model ResNet-110 (e.g., a certain convolutional layer).

Input the training data into the teacher model and extract the feature representations from the intermediate layers.

These feature representations (factors) will serve as guiding signals for the student model.

Simplified Explanation:

This step is like asking the teacher to share their study notes (intermediate layer feature representations). These notes contain detailed explanations and examples, which are more valuable for learning than just the final answers.

3) Training the Student Model (ResNet-20)

Goal: Train a smaller student model, ResNet-20, by learning and imitating the intermediate feature representations from the teacher model to improve its own performance.

Steps:

Initialize a ResNet-20 model.

Input the training data into the student model, while also providing the teacher model's intermediate feature representations as guiding signals.

During training, optimize the student model to ensure that its intermediate layer outputs are as close as possible to the teacher model's intermediate layer outputs.

Simultaneously, the student model also learns the labels of the training data (hard labels) in a standard supervised learning fashion.

Simplified Explanation:

This step is like a beginner student (ResNet-20) learning by not only listening to the teacher's lectures (learning from the labels of the training data), but also by referring to the teacher's detailed notes (intermediate feature representations). This helps the student better understand key concepts and master problem-solving techniques.

DML has a higher complexity and involves collaborative training of multiple student models without the need for a pre-trained teacher model. It is suitable for resource-constrained scenarios where no pre-trained teacher model is available, and performance is improved through mutual learning among multiple student models [3].

1) Initializing Multiple Student Models (ResNet-20)

Goal: First, we need to initialize multiple student models (let's assume two ResNet-20 models). These models are independent at the start of training, with no prior knowledge.

Steps:

Initialize two ResNet-20 models, naming them Student Model A and Student Model B.

Prepare the training dataset (e.g., CIFAR-100).

Simplified Explanation:

This step is like starting a new semester, where there are two students (Student A and Student B) in the class, both beginners, ready to learn together.

2) Mutual Learning

Goal: During training, the two student models learn from each other. Each model learns not only from its own loss but also from the outputs of the other model. This helps each model gain more information and better guidance.

Steps:

Feed the training data into both Student Model A and Student Model B.

Student Model A computes its own loss function and predicts the output.

Student Model B computes its own loss function and predicts the output.

The loss function of Student Model A includes two parts: its own prediction error and the difference from the output of Student Model B.

Similarly, Student Model B's loss function includes its own prediction error and the difference from the output of Student Model A.

Using backpropagation, update the parameters of both Student Model A and Student Model B simultaneously.

Simplified Explanation:

This step is like having Student A and Student B both work on the same exercises. Not only do they solve problems on their own, but they also compare answers with each other. If their answers differ, they discuss and compare to find the correct solution, improving their understanding.

3) Joint Optimization

Goal: Through joint optimization, both student models continuously improve during training and ultimately achieve high performance.

Steps:

In each training step, compute the joint loss for both Student Model A and Student Model B.

The joint loss formula is as follows

$Loss_A = Loss_{A,self} + a \bullet Loss_{A,mutual}$

$Loss_B = Loss_{B,self} + a \bullet Loss_{B,mutual}$

By optimizing the joint loss function, update the parameters of both Student Model A and Student Model B, gradually improving their performance.

Simplified Explanation:

This step is like Student A and Student B constantly discussing and comparing answers, correcting mistakes and misunderstandings together, eventually becoming smarter and more capable.

CRD has a high computational complexity and, similar to FT, transfers knowledge through feature representations in intermediate layers. It involves contrastive learning and contrastive loss, optimizing feature representations through contrastive learning. It is suitable for scenarios where the student model needs to learn rich feature representations [4].

1) Train the Teacher Model

Goal: First, train a high-performance teacher model (e.g., ResNet-110). This model is trained on a large dataset (such as CIFAR-100) to learn rich features and patterns.

Steps:

Obtain the training dataset (e.g., CIFAR-100), including images and corresponding labels.

Use this data to train the ResNet-110 model, optimizing its parameters to achieve the highest possible accuracy.

Once training is complete, save the trained ResNet-110 model.

Simplified Explanation:

Imagine you are an experienced teacher who, after years of study and practice, has mastered a large amount of knowledge and skills. Now, you store all this knowledge in a notebook (ResNet-110).

2) Extract Feature Representations

Goal: Extract feature representations from the intermediate layers of the trained ResNet-110 teacher model.

Steps:

Select one or more intermediate layers of the teacher model (e.g., a specific convolutional layer).

Feed the training data into the teacher model and extract the feature representations from the intermediate layers.

Simplified Explanation:

This step is like opening the teacher's notebook and finding some key notes (intermediate feature representations). These notes contain detailed explanations and examples.

3) Initialize the Student Model

Goal: Initialize a student model (e.g., ResNet-20), which is smaller and suitable for faster training and deployment.

Steps:

Initialize the ResNet-20 model and prepare the training dataset.

Simplified Explanation:

Now, a beginner student (ResNet-20) is ready to start learning this knowledge.

4) Contrastive Learning

Goal: Optimize the feature representations of the student model through contrastive learning, so that they become close to the teacher model's feature representations in the embedding space, while being distant from the feature representations of unrelated samples.

Steps:

Feed the training data into the student model and simultaneously input the feature representations from the teacher model.

For each sample, the student model outputs a feature representation (student feature), while the teacher model outputs a corresponding feature representation (teacher feature).

Compute the similarity between the student and teacher feature representations (typically using cosine similarity).

Use a contrastive loss function (such as InfoNCE loss) to optimize the student model's feature representations, making them as close as possible to the teacher model's feature representations while pushing away the feature representations of unrelated samples.

Simplified Explanation:

This step is like a novice student (ResNet-20) not only understanding the teacher's notes (teacher feature representations) but also comparing their learning with other classmates (unrelated samples) to ensure their understanding is both correct and unique. The process of contrastive learning is similar to a classroom setting, where you not only need to understand the teacher's explanations but also engage in discussions and comparisons to ensure you've truly mastered the right knowledge.

BSS requires multiple rounds of iterative training, which results in longer training times. Similar to KD, it transfers knowledge through soft labels, but instead of using a pre-trained teacher model, it uses its own iterative process. It is suitable for scenarios where a simple and effective knowledge transfer method is needed[5].

1) Train the Initial Model

Goal: First, train an initial model, which will serve as both the teacher model and the student model in subsequent steps.

Steps:

Obtain the training dataset (e.g., CIFAR-100), including images and corresponding labels.

Use this data to train an initial model (e.g., ResNet-20 or ResNet-110).

Once training is complete, save the trained initial model.

Simplified Explanation:

Imagine you are a freshman learning knowledge for the first time and completing your studies. This initial model is like your grades for the first semester.

2) Self-Distillation

Goal: Use the trained initial model as a new teacher model and train a student model again. This process can be repeated multiple times, with each model iteration referred to as ”Born-Again”.

Steps:

Use the initial model as the teacher model to generate soft targets.

Initialize a new student model (it can be the same model architecture, such as ResNet-20).

Train the student model using the soft targets generated by the teacher model along with the original training data.

Once the student model is trained, it becomes a ”Born-Again” model.

Use the new student model as the new teacher model, and repeat steps 2-4 for multiple iterations.

Simplified Explanation:

This is like using your notes from the first semester (teacher model) to study again in the second semester. Each time you study, your understanding improves, and your grades get better and better. At the end of each semester, your notes become more detailed (the new student model becomes the new teacher model), providing better guidance for the next semester.

B. *Model Library Construction:*

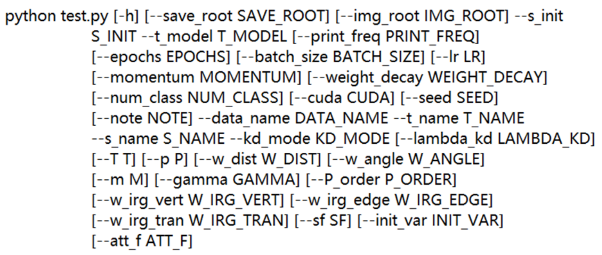
The results of the model library have been packaged into a software package and are planned to be released on PyPI soon. Below is a demonstration of how to use the library for simple model training.

First, use pip to install the package “SRTP-0.1.tar.gz” in the installation directory. (Once the package is published on PyPI, you can directly install it in the virtual environment with the command pip install SRTP).

Next, create a file named test.py and import SRTP to call the six training functions provided in the package.(eg: import SRTP SRTP.train_base())

When using these functions, certain parameters need to be specified (**Figures 1–6**):

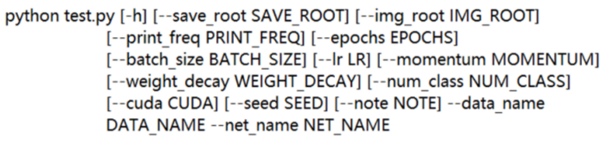
Usage of train_kd():



```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT] --s_init
             S_INIT --t_model T_MODEL [--print_freq PRINT_FREQ]
             [--epochs EPOCHS] [--batch_size BATCH_SIZE] [--lr LR]
             [--momentum MOMENTUM] [--weight_decay WEIGHT_DECAY]
             [--num_class NUM_CLASS] [--cuda CUDA] [--seed SEED]
             [--note NOTE] --data_name DATA_NAME --t_name T_NAME
             --s_name S_NAME --kd_mode KD_MODE [--lambda_kd LAMBDA_KD]
             [--T T] [--p P] [--w_dist W_DIST] [--w_angle W_ANGLE]
             [--m M] [--gamma GAMMA] [--P_order P_ORDER]
             [--w_irg_vert W_IRG_VERT] [--w_irg_edge W_IRG_EDGE]
             [--w_irg_tran W_IRG_TRAN] [--sf SF] [--init_var INIT_VAR]
             [--att_f ATT_F]
```

Figure 1. kd parameters.

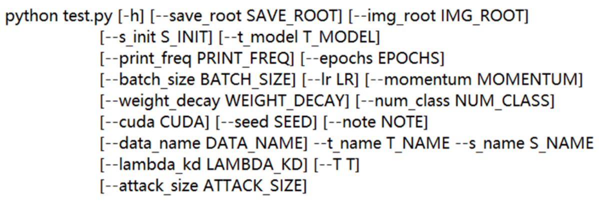
Usage of train_base():



```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT]
               [--print_freq PRINT_FREQ] [--epochs EPOCHS]
               [--batch_size BATCH_SIZE] [--lr LR] [--momentum MOMENTUM]
               [--weight_decay WEIGHT_DECAY] [--num_class NUM_CLASS]
               [--cuda CUDA] [--seed SEED] [--note NOTE] --data_name
               DATA_NAME --net_name NET_NAME
```

Figure 2. base parameters.

Usage of train_bss():



```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT]
              [--s_init S_INIT] [--t_model T_MODEL]
              [--print_freq PRINT_FREQ] [--epochs EPOCHS]
              [--batch_size BATCH_SIZE] [--lr LR] [--momentum MOMENTUM]
              [--weight_decay WEIGHT_DECAY] [--num_class NUM_CLASS]
              [--cuda CUDA] [--seed SEED] [--note NOTE]
              [--data_name DATA_NAME] --t_name T_NAME --s_name S_NAME
              [--lambda_kd LAMBDA_KD] [--T T]
              [--attack_size ATTACK_SIZE]
```

Figure 3. bss parameters.

Usage of train_crd():

```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT]
              --s_init S_INIT --t_model T_MODEL
              [--print_freq PRINT_FREQ] [--epochs EPOCHS]
              [--batch_size BATCH_SIZE] [--lr LR] [--momentum MOMENTUM]
              [--weight_decay WEIGHT_DECAY] [--num_class NUM_CLASS]
              [--cuda CUDA] [--seed SEED] [--note NOTE] --data_name
              DATA_NAME --t_name T_NAME --s_name S_NAME
              [--lambda_kd LAMBDA_KD] [--feat_dim FEAT_DIM]
              [--nce_n NCE_N] [--nce_t NCE_T] [--nce_mom NCE_MOM]
              [--mode {exact,relax}]
```

Figure 4. crd parameters.

Usage of train_dml():

```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT]
              --net1_init NET1_INIT --net2_init NET2_INIT
              [--print_freq PRINT_FREQ] [--epochs EPOCHS]
              [--batch_size BATCH_SIZE] [--lr LR] [--momentum MOMENTUM]
              [--weight_decay WEIGHT_DECAY] [--num_class NUM_CLASS]
              [--cuda CUDA] [--seed SEED] [--note NOTE] --data_name
              DATA_NAME --net1_name NET1_NAME --net2_name NET2_NAME
              [--lambda_kd LAMBDA_KD]
```

Figure 5. dml parameters

Usage of train_ft():

```
python test.py [-h] [--save_root SAVE_ROOT] [--img_root IMG_ROOT] --s_init
              S_INIT --t_model T_MODEL [--print_freq PRINT_FREQ]
              [--epochs EPOCHS] [--batch_size BATCH_SIZE] [--lr LR]
              [--momentum MOMENTUM] [--weight_decay WEIGHT_DECAY]
              [--num_class NUM_CLASS] [--cuda CUDA] [--seed SEED]
              [--note NOTE] --data_name DATA_NAME --t_name T_NAME
              --s_name S_NAME [--lambda_kd LAMBDA_KD] [--k K]
```

Figure 6. ft parameters.

Usage notes:

When using the library for the first time or when you need to verify a specific knowledge transfer method, users must first train a base model using train_base(). This base model will then serve as the teacher model for applying other knowledge transfer methods.

The datasets we used for testing are CIFAR-10 and CIFAR-100, and we do not guarantee that the code will work successfully with other datasets. To ensure the best performance, we recommend that users download CIFAR-10 and CIFAR-100 and input the dataset directory name as the IMG_ROOT parameter.

The model library is built on PyTorch, so users must ensure that all necessary dependencies are installed in their environment.

C. *Visualization of Results:*

A visualization tool was developed to intuitively display the training process and results of different models. This tool allows users to compare key metrics such as training speed, accuracy, and loss across various knowledge transfer methods, making it easier to evaluate model performance.

We use the third-party library bar_chart_race_cn to visualize the training results in GIF format. The code is included in the attachment.

3. Experiments and results

It can be observed that after training for 10 epochs, the student networks trained using all methods reached the accuracy level of the teacher network on the test set. Some methods, such as BSS, FT, and KD, produced student networks that performed even better than the teacher network (**Figure 7**). This also confirms the goal of knowledge transfer in the teacher-student architecture, where the student model, with fewer parameters, can achieve performance comparable to the teacher model.

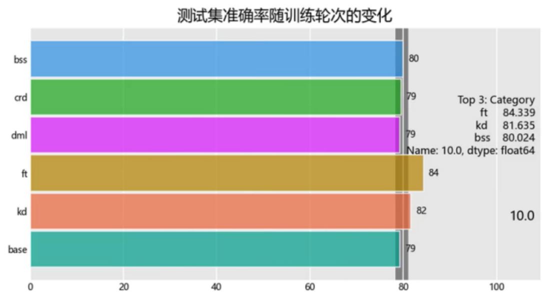


Figure 7. The accuracy of the test set varies with training epochs visual display.

Impact of Networks

Taking the CIFAR-10 dataset as an example with FT, we found that the performance of Teacher ResNet-110 and Student ResNet-110 (92.98%) is greater than that of Teacher ResNet-110 and Student ResNet-20 (93.01%), which in turn is greater than that of Teacher ResNet-20 and Student ResNet-20 (94.30%).

DML shows a similar trend overall (**Figure 8**).

Net1	Net2	Name	CIFAR10	CIFAR100
-	resnet-20	baseline	92.37%	68.92%
-	resnet-110	baseline	93.86%	73.15%
resnet20	resnet20	DML	93.07%/93.37%	70.39%/70.22%
resnet110	resnet20	DML	94.45%/92.92%	74.53%/70.29%
resnet110	resnet110	DML	94.74%/94.79%	74.72%/75.55%

Figure 8. Overall results of DML.

This indicates that deeper networks can learn more complex and abstract feature representations, making them suitable for handling complex tasks and large-scale datasets. As a result, the performance is more accurate.

Influence of Datasets

CIFAR-10: With fewer categories, the task is relatively simple, making it suitable for beginners and experiments with simpler models.

CIFAR-100: With more categories, the task complexity is higher, making it suitable for studying more complex models and methods.

For all five algorithms, we find that as the dataset becomes more complex, the final accuracy decreases. In other words, the performance on the CIFAR-10 dataset is better than that on the CIFAR-100 dataset.

Comparison of Results from Different Methods

We found that the final performance of student models learned through different knowledge distillation methods is relatively similar. However, DML performs better on both CIFAR-10 and CIFAR-100 compared to other algorithms. This may be because DML effectively leverages the collaboration between models, enhancing their learning ability and generalization performance.

4. Notes

Networks:

In the study of knowledge distillation algorithms, networks play multiple important roles, specifically including the teacher model and the student model. The roles of each network are as follows:

Teacher Model:

Knowledge Provider: The teacher model is typically a large and high-performance model that has been thoroughly trained on a large dataset, capable of providing accurate predictions and rich feature representations.

Generating Soft Targets: In standard knowledge distillation, the soft targets generated by the teacher model contain information about the class probability distribution, which offers more insights into the data distribution than hard labels (one-hot labels).

Feature Extraction: In some knowledge distillation methods (such as Factor Transfer), the intermediate layer feature representations of the teacher model are used to guide the training of the student model.

Student Model:

Knowledge Receiver: The student model is a smaller and lightweight model designed to retain the knowledge of the teacher model while reducing model complexity and the number of parameters.

Model Compression: By learning from the teacher model, the student model can significantly reduce its size without a substantial decrease in performance, making it suitable for deployment in resource-constrained environments.

Faster Inference: Since the student model is smaller, it requires less computation and has lower latency during inference, making it suitable for real-time applications.

The difference between ResNet-20 and ResNet-110:

ResNet (Residual Network) is a deep neural network architecture that addresses the vanishing gradient problem in deep networks by introducing residual connections. ResNet-20 and ResNet-110 are two different variants within the ResNet family, and their primary difference lies in the depth of the network, specifically the number of layers.

ResNet-110 is often used as a teacher model because its depth and high performance can provide richer knowledge and feature representations for the student model to learn from.

ResNet-20 is commonly used as a student model due to its smaller architecture, making it more suitable for learning from the teacher model during the distillation process, achieving a balance between performance and complexity.

Differences between the CIFAR-10 and CIFAR-100 Datasets:

CIFAR-10 and CIFAR-100 are two commonly used image classification datasets created by Alex Krizhevsky and Geoffrey Hinton at the University of Toronto. The main differences between them lie in the number of classes and the level of granularity.

Number of Classes:

CIFAR-10: Contains 10 classes, which are airplane, automobile, bird, cat, deer, dog, frog, horse, ship, and truck.

CIFAR-100: Contains 100 classes, with fewer samples per class compared to CIFAR-10.

Class Hierarchy:

CIFAR-10: Each class is standalone with no hierarchical structure.

CIFAR-100: Each class belongs to a superclass, and there are 20 superclasses in total. Each superclass contains 5 fine-grained classes.

Data Volume:

CIFAR-10 and CIFAR-100: Both datasets contain the same amount of data, with 50,000 images in the training set and 10,000 images in the test set. Each image is a 32x32 color image.

Sample Complexity:

CIFAR-10: Due to the fewer number of classes, the tasks are relatively simple, making it suitable for beginners and simple model experiments.

CIFAR-100: The larger number of classes increases task complexity, making it more suitable for studying more complex models and methods.

Specific Use Cases:

Model Training:

CIFAR-10: Used for rapid prototyping and testing of simple image classification models. Due to the fewer classes and larger number of samples per class, it allows faster training and quicker experiment iterations.

CIFAR-100: Used for testing and validating the performance of complex models. It is very useful for research on fine-grained classification tasks.

Model Evaluation:

Test Sets: Both CIFAR-10 and CIFAR-100 can be used to evaluate the final performance of models. Testing on unseen data helps assess the model's generalization ability.

5. Advanced methods introduction

RKD (Relational Knowledge Distillation)

RKD emphasizes the preservation of relationships between the student model and the teacher model. It guides the training of the student model by maintaining the relationships between input samples. RKD is a method of knowledge distillation that focuses on preserving the relationships between samples. Compared to traditional knowledge distillation, which directly distills the output results, RKD pays more attention to the geometric relationships between samples, using these relationships to guide the training of the student model[6].

SP (Similarity-Preserving Knowledge Distillation)

SP performs knowledge distillation by preserving the similarity between samples, ensuring that the student model can learn the structural information from the teacher model. SP emphasizes that during the learning process, the student model should retain the sample similarities in the feature space as much as possible. This way, the student model not only learns the classification capabilities of the teacher model but also captures the teacher model's deeper understanding of the data.

AFD (Attention-based Feature Distillation)

AFD utilizes an attention mechanism to select important features for distillation, enhancing the distillation effectiveness through the introduction of attention modules. AFD is a method of knowledge distillation that leverages attention mechanisms to choose significant features for distillation, thereby improving the performance of the student model[7].

6. Discussion

The model library constructed in this study simplifies the implementation of knowledge transfer methods while the visualization tool enhances the efficiency of result analysis. By comparing the effectiveness of different strategies, it is evident that each knowledge transfer method has its strengths depending on the specific task. Future work could involve expanding the model library to support more transfer strategies and custom parameter tuning, further increasing its applicability.

1. **Difference in Knowledge Source and Transfer Method:**

KD (Knowledge Distillation) and BSS (Born-Again Self-Distillation): Knowledge is transferred through soft labels. KD uses a pre-trained teacher model, while BSS iterates using its own progressively trained models.

FT (Feature Transfer) and CRD (Contrastive Representation Distillation): Knowledge is transferred through intermediate feature representations. FT directly uses feature representations, while CRD optimizes them through contrastive learning.

DML (Deep Mutual Learning): Multiple student models learn from each other without the need for a pre-trained teacher model.

2. **Difference in Training Complexity and Implementation Difficulty:**

KD: Relatively simple, with the teacher model's soft labels guiding the student model.

FT: Requires extracting and transferring intermediate feature representations, which increases complexity.

DML: Involves collaborative training of multiple student models, making it more complex.

CRD: Involves contrastive learning and contrastive loss, leading to higher computational complexity.

BSS: Requires multiple rounds of iterative training, resulting in longer training times.

3. **Difference in Applicable Scenarios:**

KD and BSS: Suitable for scenarios requiring a simple and effective knowledge transfer method.

FT and CRD: Suitable for scenarios where the student model needs to learn rich feature representations.

DML: Suitable for resource-constrained scenarios where no pre-trained teacher model is available, and performance is improved through mutual learning among student models.

7. Conclusion

This paper provides a detailed summary and replication of five classical knowledge transfer methods and three cutting-edge techniques, culminating in the successful construction of a flexible model library. Through extensive experimentation, this research offers new insights into knowledge transfer and provides a valuable framework for future developments. The model library supports the rapid integration of multiple strategies and facilitates result visualization, offering significant convenience for researchers and developers.

References

[1] Hinton, G., Vinyals, O., Dean, J., 2015. Distilling the knowledge in a neural network. Advances in Neural Information Processing Systems. 28, 1–9.

[2] Yosinski, J., 2014. Transfer learning via sparse fine-tuning. Proceedings of the 31st International Conference on Machine Learning (ICML); Beijing, China; 22–24 June 2014. pp. 370–378.

[3] Zhang, Y., Xiang, T., Hospedales, T. M., et al., 2018. Deep mutual learning. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Salt Lake City, UT, USA; 18–22 June 2018. pp. 1–9.

[4] Tian, Y., Krishnan, D., Isola, P., 2020. Contrastive representation distillation. Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR); Seattle, WA, USA; 14–19 June 2020. pp. 1–10.

[5] Furlanello, T., Lipton, Z., Tschannen, M., et al., 2018. Born-again neural networks. Proceedings of the 35th International Conference on Machine Learning (ICML); Stockholm, Sweden; 10–15 July 2018. pp. 1603–1612.

[6] Park, W., Kim, D., Lu, Y., et al., 2019. Relational knowledge distillation. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR); Long Beach, CA, USA; 16–20 June 2019. pp. 1–10.

[7] Wang, Q., 2021. Attention-based feature distillation for image classification. IEEE Transactions on Neural Networks and Learning Systems. 32(5), 2040–2050.