Delft University of Technology
Master of Science Thesis in Embedded Systems

Resource Efficient Knowledge Distillation on Edge Devices

Ting Liang

Resource Efficient Knowledge Distillation on Edge Devices

Master of Science Thesis in Embedded Systems

Embedded Systems Group
Faculty of Electrical Engineering, Mathematics and Computer Science
Delft University of Technology
Van Mourik Broekmanweg 6, 2628 XE Delft, The Netherlands

Ting Liang

21/08/2024

Author
Ting Liang
Title
Resource Efficient Knowledge Distillation on Edge Devices
MSc Presentation Date
28/08/2024

Graduation Committee

Guohao Lan	Delft University of Technology
Koen Langendoen	Delft University of Technology
Georgios Iosifidis	Delft University of Technology

Abstract

In practical situations, computer vision technique is applied to solve various tasks, including image classification, object detection, image segmentation, and so on. The commonly used supervised learning training paradigm for the network models used to solve these tasks requires training data as well as the ground truth labels specifying the data samples' reference information for the task. However, getting labels for every task would be expensive or even almost impossible, such as medical images due to privacy reasons and expert annotations from medical professionals and facial recognition also because of privacy concerns. Many large-scale general datasets exist, like ILSVRC2012 for both image classification and object detection tasks and COCO also for objection tasks. , and the corresponding pre-trained models whose knowledge can be transferred to other fields. While deeper models typically have better performance and can learn better feature representation from the same tasks, increasing network models introduces difficulties in practical deployment, especially regarding resource limitation and response latency. We hope to explore the learning methods in knowledge distillation to help the smaller student network learn better features from the unlabeled training data and have better transfer performance on downstream tasks. With the remarkable success of contrastive learning, it has become one of the most promising methods of learning from unlabeled data. In this thesis, we proposed an unsupervised knowledge distillation method that applies a contrastive learning method to construct and extract relational knowledge from the feature representations of the intermediate layers as well as the final layer. The evaluation with the ILSVRC2012 dataset proves the effectiveness of the proposed method on feature learning as the method helped the ResNet-18 model achieve a 2% improvement in linear evaluation accuracy compared to the baseline model. Extensive experiments were conducted on eight transfer learning tasks, and the model trained by the proposed method outperformed its baseline model in all the eight classification tasks and also achieved better fine-tuning accuracy in the situation where only a small fraction of the ground truth label was available for fine-tuning.

Preface

This thesis presents my work on the graduate project for the Master of Science in Embedded Systems at TU Delft. The project started with an extra research project on knowledge distillation methods in supervised learning under the guidance of Dr. Guohao Lan. The research topic was to explore applying knowledge distillation in unsupervised learning. Through this project, I gained rich theoretical and practical knowledge and experience with knowledge distillation and contrasive learning. I also developed a more in-depth understanding of deploying deep learning models on resource-limited edge devices.

I would like to express my deepest gratitude to Dr. Guohao Lan for his invaluable guidance, professional advice, and detailed feedback throughout the research project. I am also deeply thankful to Lingyu for all the support, suggestions, and encouragement for my work. In addition, I would also like to thank Dr. Langendoen for his assistance with the graduation matters and for being on my thesis committee, as well as Dr. Iosifidis for being on my thesis committee. Last but not least, I would like to thank my friends and family for supporting me during the project.

Ting Liang

Delft, The Netherlands
2nd September 2024

Contents

Chapter 1

Introduction

1.1 Introduction

The widely applied supervised deep learning algorithms have demonstrated their outstanding capabilities in different tasks like classification [37], object detection [59], and so on. However, supervised learning methods often require large amounts of labeled data to achieve optimal performance [28]. Generally speaking, the performance of deep neural networks heavily relies on the size [58] of the training dataset. More training samples contain more variations of the features and reduce the risk of outfitting the networks on certain patterns to learn more robust and generalizable features. The research of Sun et al. [58] demonstrated that performance on vision tasks increases logarithmically along with the volume increase of training data size. The process of collecting and annotating data is time-consuming and labor-intensive [66]. For example, collecting annotations for medical imaging applications is very costly as such labels require complicated domain expertise that is only available from trained doctors. Therefore, the challenges of obtaining and labeling data have become significant bottlenecks, limiting the practical applicability of supervised deep learning methods.

Recent progress in unsupervised learning methods has greatly alleviated such bottlenecks and demonstrated their potential for efficiently learning from unlabeled data [11]. Among the various unsupervised learning methods, contrastive learning (CL) has demonstrated the potential of utilizing unsupervised representation learning to learn discriminative embedding feature representations from unlabeled data [3],[66]. CL constructs representations of contrasting positive and negative pairs of examples in the embedding spaces and maps similar examples closer together in the feature space while pushing dissimilar examples farther apart to learn from the unlabeled training data. Recent study [8] indicated that models demonstrated better representation learning capabilities as the model capacity increases, and larger models can receive more benefit in performance from self-supervised learning methods.

In applications such as edge computing, due to the consideration of data privacy and computation efficiency, we would like to have the processing and computation take place locally instead of remotely [4]. However, due to the limitations of edge devices, such as computation capability and memory, the most sophisticated models with the best performance may not always be feasible to be deployed in practical situations. These models typically require substantial computation resources to perform calculations and inference [7]. This issue has been addressed using different techniques from different points of view to compress these complex models, such as model quantization [25] and model pruning [13]. Knowledge distillation, via transferring the knowledge from a larger model to a smaller model, is able to achieve the goal of model compression [27].

This thesis project aims to compress a pre-trained large model by transferring the features learned by the larger model from unlabeled training data to a smaller model that is easier to deploy and achieves comparable transfer performance on the downstream tasks with the large model. Specifically, we proposed an unsupervised knowledge distillation method that applies a contrastive learning method to construct similarity relation knowledge for the input augmented data. While other unsupervised knowledge distillation methods focus on making use of the final embedding of the network [24], we hope to also take advantage of the feature embeddings of the intermediate layers for knowledge distillation in an unsupervised learning situation as each of these representations also contains the fruitful feature and semantic information of different levels. This method is inspired by how Contrastive Deep Supervision [72] utilizes the intermediate layer feature embedding to assist the normal supervised learning training of the models. A two-step training and distillation process to realize this method is also proposed accordingly. In this process, a series of auxiliary networks are attached to the pre-trained teacher network and trained to extract the intermediate layer feature embedding in the first step and guide the training of the student model with the intermediate layer and final embedding from the teacher model in the second step.

We conduct extensive evaluations to demonstrate the effectiveness of the proposed method on ILSVRC 2012 [56] dataset and eight transfer learning datasets with ResNet-50 and ResNet-18 networks as the teacher and student pair. The knowledge distillation trained ResNet 18 network achieved a 2% improvement compared to the baseline ResNet 18 network trained with SimCLR. Furthermore, we evaluate the transfer learning performance of the proposed method on eight downstream classification tasks. In all eight tasks, the model trained by the proposed method outperformed the baseline model by up to 3.5%, and the ResNet 18 network trained by the proposed knowledge distillation method also had better performance on the few-shot scenario where only a small fraction of the training set labels are available in the evaluation of the models, performed on two of the tasks.

The main contributions of this thesis are summarized as follows:

- The thesis proposed an unsupervised knowledge distillation method that utilizes contrastive learning to construct and extract knowledge from intermediate layers and the final layer of the network and train the student network model to facilitate better learning of the feature representation from the unlabeled data and achieve a better transfer performance on downstream classification tasks. A corresponding 2-step training and distillation process is also proposed to realize this method.

- Extensive experiments among eight downstream classification tasks exploring the performance of features of the student network learned from the proposed knowledge distillation method.

1.2 Thesis Content

The remainder of the thesis is structured as follows:

- **Related Works**. Chapter 2 provides a comprehensive overview of the related background topics of this thesis.
- **Method**. Chapter 3 presents the design of the proposed unsupervised knowledge distillation method and the step-wise, in-detail introduction of the training process of the smaller model.
- **Evaluation**. The implementation of the knowledge distillation and the evaluation process and results of the distillation are elaborated in Chapter 4.
- **Conclusion**. The conclusion summarizing this thesis and the discussion for future research possibilities are presented in Chapter 5

Chapter 2

Related Work

This chapter describes the theoretical background and related research this project builds upon or relates to. Section2.1 introduces the general pipeline and the principle of self-supervised learning and provides some examples of pretext tasks in self-supervised learning. Section2.2 introduces contrastive learning as a variation of self-supervised learning with its architecture and principle. Section2.3 discusses in detail SimCLR, the specific contrastive learning method applied in this project, with its architecture and algorithm. Section 2.4 introduces the concept of the general model compression technique and two of the major model compression techniques: pruning and model quantization. Section2.5 provides an overview of the knowledge distillation technique and some existing works on combining knowledge distillation and contrastive learning.

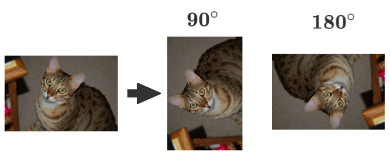
2.1 Self-Supervised Learning

Self-supervised learning is a machine learning paradigm designed to extract meaningful information from unlabeled training data. To address the challenges of lack of expensive labeled data required by supervised learning, self-supervised learning extracts representations from unlabeled data by designing pre-text tasks that automatically generate labels from existing unlabeled data [28]. An effective pretext task requires models to capture visual representations within the unlabeled training data to solve them. The focus of these pretext tasks is to learn intermediate representations that contain rich semantic and structural knowledge of the data [46] and consequently benefit the downstream tasks. Common pretext task examples are described as follows as well as in Fig 2.1.

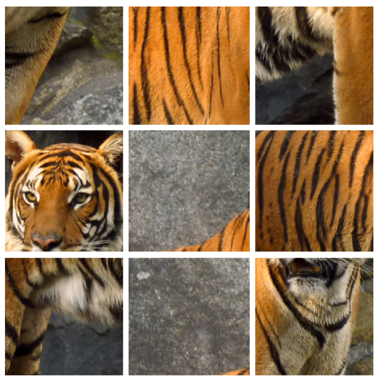
- **Image in-painting [54]**The encoder is trained to reconstruct the missing regions of a picture in a way that is visually coherent with the rest of the image. The encoder network can learn the visual and structural features of the unlabeled data.

- **Rotation prediction [26]**The encoder is trained to predict the degree of the rotation transformation with respect to the original image. Rotation-invariant features can be learned from this task.

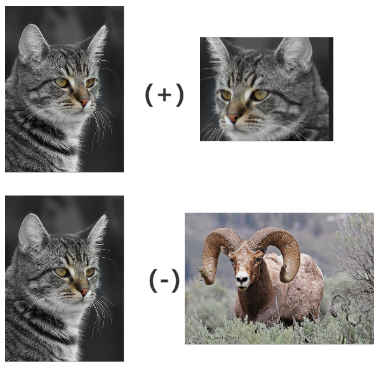
(a) Inpainting [54]



(b) Rotation Prediction



(c) Jigsaw Puzzle [38]



(d) Instance discrimination

Figure 2.1: **Examples of pretext tasks. (a) the network learns to rebuild the picture; (b) the network is trained to predict the rotation angle of the input with respect to the original image; (c) the network is trained to solve the jigsaw puzzle; (d) the network is trained to identify similar and different instances.**

- **Jigsaw puzzles** [**48**]The encoder is trained to recover a jigsaw puzzle to attain the original image. The encoder network can learn the spatial relations of the different parts of the original image and their underlying features.
- **Instance discrimination**[**21**]The encoder is trained to distinguish between instances with different visual features.

As illustrated in Fig.2.2, the general process of self-supervised learning begins with defining a pretext task to generate pseudo labels from the unlabeled data automatically. The network models are then trained using objective loss functions based on these pretext tasks to learn the underlying visual feature representations. These learned task-irrelevant representations can subsequently be transferred to specific downstream tasks [39].

2.2 Constrasive learning

Contrastive learning is a form of self-supervised learning method that achieves state-of-the-art performance [8].

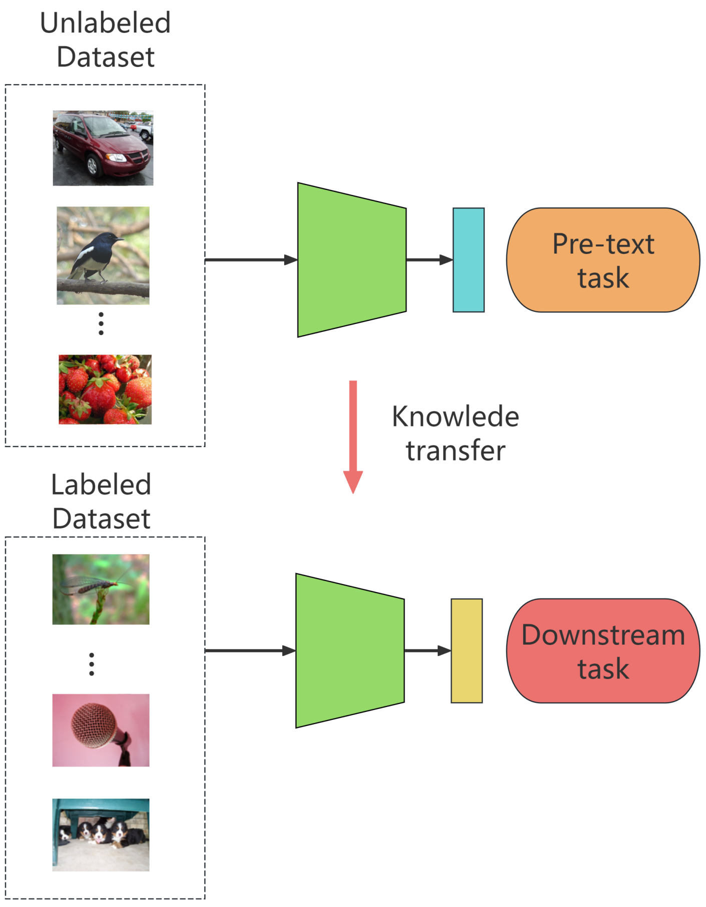


Figure 2.2: **A general pipeline of self-supervised learning. The encoder network is trained with the unlabeled dataset to learn the task-irrelevant feature knowledge, and these can be transferred and applied to specific downstream tasks as pre-trained parameters.**

2.2.1 Overview

Research has demonstrated that contrastive learning, leveraging instance discrimination as a self-supervised learning pretext task, is very promising in visual representation learning [41]. The goal of contrastive learning is to learn the feature representations from the training data via differentiating similar and dissimilar data samples. Specifically, contrastive learning aims to pull the representations of visually similar image pairs, called positive pairs, together while pushing the representation of visually distinct image pairs, known as negative pairs, away.

Fig.2.3 demonstrates a typical architecture for contrastive learning. Each original data sample forms a class itself, and the augmented views of that image inherit such pseudo-classification labels and are regarded as positive pairs. Applying transformations to form positive pairs aims to facilitate the model in

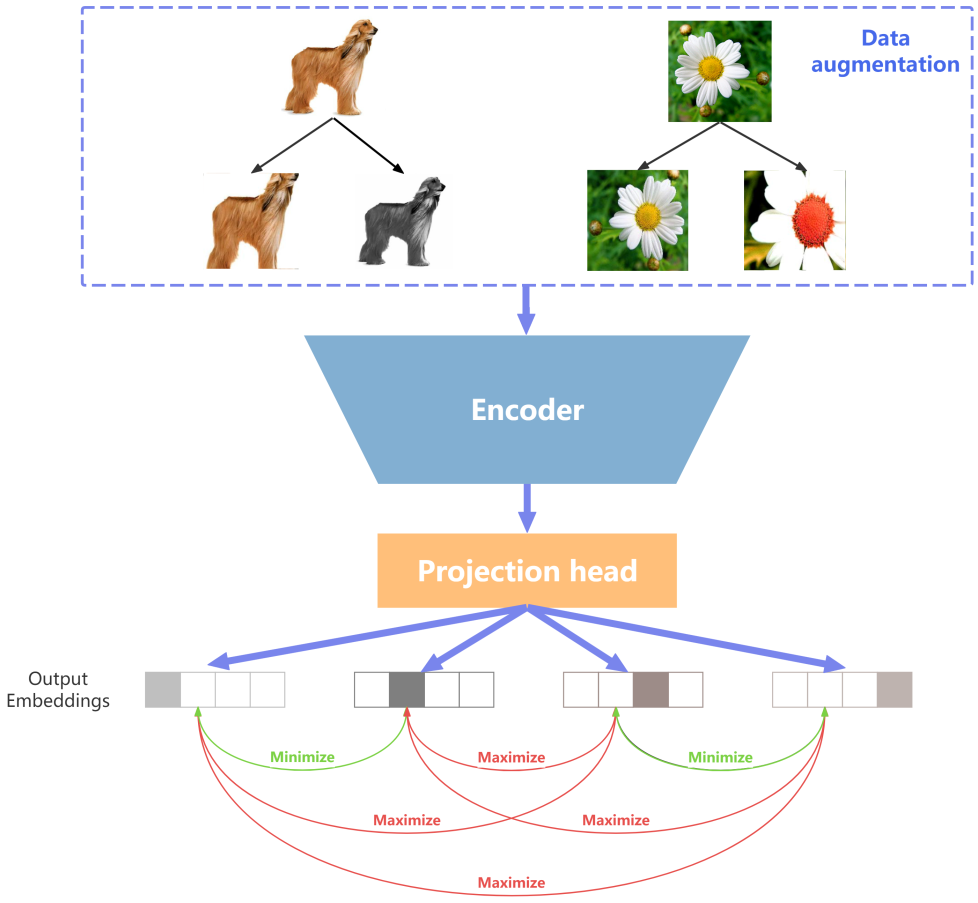


Figure 2.3: **A general architecture for contrastive learning. Augmented views of the same input data sample are regarded as positive pairs and negative pairs are generated from views of different input data within the data batch. The network model is trained to minimize the distance between the output embedding of the positive pairs and maximize those between negative pairs.**

learning transfer-invariant features of the training data. Then, automatically, each augmented sample forms negative pairs with all the samples from different originals. The similarities of these samples are measured in the embedding space with the encoder and projection head network extracting the features from the data samples. The fundamental training target of contrastive learning can be expressed as learning an encoder network f such that for any data point x, the similarity between the anchor point and data points similar to the anchor point is denoted as $Sim(f(x), f(x^+))$ and similarity between the anchor point and data points similar to the anchor point is denoted as $Sim(f(x), f(x^-))$. The score function $Sim()$ is a measurement of the similarity between feature representations. The network is trained with such a loss function that makes the first value much larger than the second similarity term.

There are different variations of contrastive loss as the construction and organization of the positive and negative samples differ[[49],[23],[51],[57]]. A series of work like SimCLR [8] and MoCo [31] has demonstrated that contrastive learning has the capability of achieving state-of-the-art performance in many applications, even surpassing their supervised counterparts in some scenarios [73].

2.2.2 Data transform for contrastive learning

Data transformation is pivotal in contrastive learning [70]. It's indispensable for enabling the model to grasp the underlying invariances and features in training data. We effectively generate augmented training elements by processing input data through transformations, facilitating the model's learning objectives.

There are various transformations applied to different contrastive pretext tasks. Listed below are some of the representative transformations:

- **Geometry transformations**. One common type of transformation in image processing problems is geometry transformation. Images' geometric properties are altered to create variations in the training data while preserving their semantic content. Resizing, cropping, and flipping are some of the most frequently applied transformations.

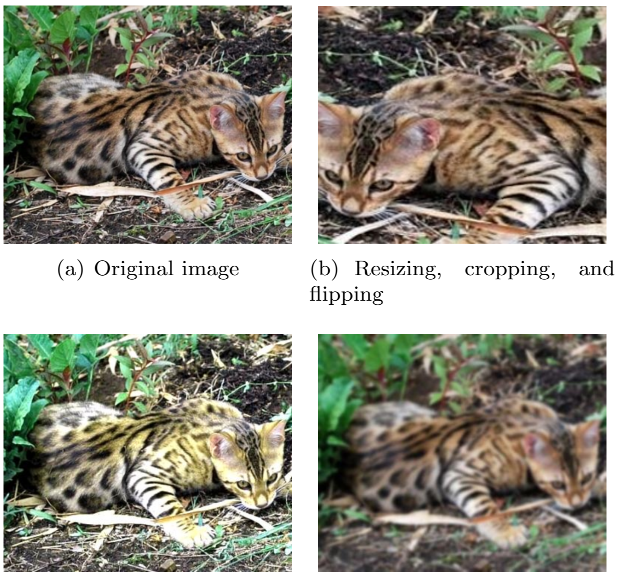
- **Appearance transformation**. Aside from lines, edges, patterns, and textures, colors and lighting are also informative features of an object in the image. Such features can be manipulated with brightness, contrast, saturation, and hue. Transforming a colored picture to a grey scale picture also has the effect of augmenting the original data. Another important form of appearance transformation is Gaussian blur. The Gaussian blur process involves applying a Gaussian filter to the input image, which smooths out pixel intensities based on their proximity to the pixels in the neighborhood. This transformation results in a blurred version of the original image. It can help denoise an image, simulate defocus, and work as regularization in some cases.

To achieve optimal training outcomes, thoughtful consideration should be given to the selection of image augmentations. However, determining the most effective combination of views and transformations remains an ongoing area of study, with varying opinions among researchers. The experiments in SimCLR [8] demonstrated that the training benefited from stronger augmentation combinations.

ReSSL [74] argues that applying aggressive data augmentation to generate positive pairs risks compromising the reliability of the target relations among training data samples [74]. Such transformation can potentially strip away crucial semantic information from the augmented data, resulting in noisy and unreliable associations.

Y. Tian et al. [39] demonstrated that the optimal view for contrastive representation learning is task-dependent, and effective learning requires the preservation of task-relevant information while minimizing irrelevant nuisances.

Some people also question the act of treating all the augmentations equally and believe that some augmentations are more significant than others and certain features learned with strong augmentations may introduce unnecessary


(a) Original image

(b) Resizing, cropping, and flipping

(c) Color jitter

(d) Gaussian blur

Figure 2.4: **Examples of image transforms**

invariance into the trained backbone model, as strong transformations of the model fine-grained information critical for the down-stream tasks.

X. Wang et al. [63] also share the idea with ReSSL that stronger augmentations may have a negative impact on representation learning as induced distortions severely hinder image structure and identity of the original picture. However, they believe that stronger augmentations could still contribute to the model performance with the help of a dedicated training framework.

2.3 SimCLR

Proposed by Chen et al. [8], SimCLR is one of the most applied state-of-the-art contrastive learning frameworks for the computer vision field and this project is developed on this framework. The general workflow of SimCLR is shown as follows: apply transformations on the input images to derive the augmented views, minimize the distance between the augmented views from the same original pictures, and maximize the distance of views from different original pictures. Fig 2.5 provides an overview of the architecture of the method. It consists of the following four primary components:

- **A stochastic data augmentation module.**It defines a random transformation function $\mathcal{T}$ that randomly transforms a given input to two correlated views of the same example. SimCLR sequentially applies random crop with resize and random flip, random color distortion, and random Gaussian blur as the data augmentation. The experiments indicated that the combination of cropping with resize and color distortion is, in particular, beneficial to the training performance.

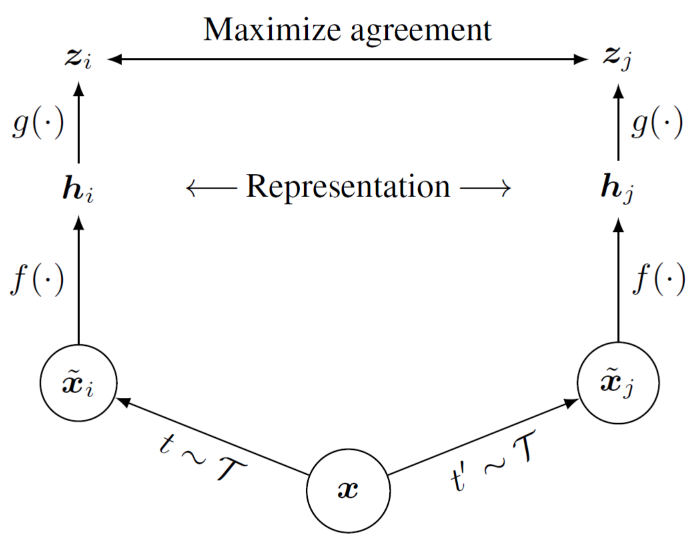


Figure 2.5: **The framework of SimCLR [8]. Each input data is applied with two random transforms to generate a positive pair. The augmented data are fed to the encoder network f to generate feature representations h. They are projected to a latent space with the projection head g, where the agreement of the embedding of the different views z within the same positive pairs is maximized.**

- **The encoder network**.The encoder network extracts the features from the augmented data and derives their representations. SimCLR uses Resnet architecture as the encoder network architecture due to its simplicity. In comparison to standard Resnet networks, the encoder only uses the backbone, the network before the final linear layer, as the final classifier is task-specific and unnecessary for the representation learning.

- **A projection head**. A smaller neural network projection head maps the feature representations extracted by the base encoder network into the latent space where the features will be compared. The non-linear MLP structure projection head has been proven to enhance training performance.

- **Loss function**. It models the relative relations between the feature representations from the input data in the latent space and optimizes the model by minimizing the loss function. Equation 2.1 demonstrates the loss term calculation between a positive pair, and the final loss is performed among all positive pairs.

Given a mini-batch containing N data samples$\{x_i\}_{i=:N}$, and for each element in this batch, we applied two transforms t, t' sampled from the same distribution $\mathcal{T}$ to derive the two augmented views $\widetilde{x_i} = t(x)$ and $\widetilde{x_j} = t'(x)$ and 2N data samples in total. The pair $(\widetilde{x_i}, \widetilde{x_j})$ are regarded as a positive pair, and the other 2(N-1) augmented examples within a mini-batch are negative examples. Then, the augmented views are fed into the encoder network to extract the feature representations $h_i = f(\widetilde{x_i})$ and $h_j = f(\widetilde{x_j})$. Then, the projection head maps the representations for comparison, as demonstrated in Fig.2.5 with $z_i = g(h_i)$ and $z_j = g(h_j)$. Cosine similarity is used here as the pairwise measurement of distances between representations, so the loss function for a positive pair (x_i, x_j) can be expressed as:

$$\ell_{i,j} = -log \frac{exp(sim(z_i, z_j)/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k \neq j]} exp(sim(z_i, z_j)/\tau)} \tag{2.1}$$

where $sim(\boldsymbol{z}_i, \boldsymbol{z}_j)$is the cosine similarity calculated by $\boldsymbol{z}_\mathbf{i}^\top \boldsymbol{z}_j / ||\boldsymbol{z}_i|| ||\boldsymbol{z}_j||$ and $\mathbb{1}_{[k \neq j]}$ is an indicator function evaluating to 1 iff $k \neq i$, and τ is a temperature that works as a scaling coefficient [64] for the distribution of the similarity between the projections. A lower τ value will emphasize the dissimilarity of those contrasting representations. However, there is a risk of introducing noise into the distribution and impairing learning performance as small distances between similar representations are also amplified. On the other hand, a τ value too small also has a negative impact on the training as the distribution will be "smoothed" and lead to the loss of important correlations.

The total loss is computed across all the positive pairs and notice that to include all the positive pairs, the elements should be considered in both sequences. The final loss function is termed as NT-Xent (Normalized temperature-scaled cross-entropy) loss [8].

2.4 Model Compression

While receiving the benefit from the network depth, deep neural network models often require a large number of parameters to achieve optimal performance [32], leading to high computational and storage demands. In practical situations, the deployment of these models may face limitations such as limited computing power and memory on target devices. Additionally, large complex models can have long inference times, which is not ideal for many real-time tasks. Thus, it is essential to develop methods that can compress these large models into smaller neural network models that retain similar accuracy while being applicable with limited resources. Model quantization and model pruning are two primary classic model compression methods.

2.4.1 Model Quantization

Quantization refers to the process of approximating the continuous values of a signal to a finite number of discrete values. It can be regarded as a method of information compression. In current development, most deep neural network models tend to be over-parameterized, so there are opportunities to reduce the bit precision without greatly affecting network accuracy, as many parameters are either redundant or less sensitive to small changes. Such redundancy, combined with the network's robustness to small perturbations, makes it possible to reduce bit precision without a large impact on network accuracy [25]. To be specific, the quantization for deep neural network models is the process of reducing the precision and the number of bits of the network model parameters and calculations so as to reduce the demand for memory and accelerate the forward inference calculations. Generally, 32-bit floating-point numbers are used in deep learning to represent parameters and calculation processes, and studies have demonstrated that the models can be further compressed by using 16-bit, 8-bit, and 4-bit values[[14],[60],[29]]. An extreme case of quantization is binary neural networks, where model weights and activation are represented with only 1-bit binary numbers [68].

The model quantization methods can be categorized into weights quantization and activation quantization [12].

Some of the model quantization methods focus on quantizing model weight parameters to reduce the model size [62, 75, 42, 18][[62],[75],[42],[18]], while other approaches also try to reduce the time from the time-consuming floating point operations, using fix-point values in activations calculation and representations. XNOR-Net [55] extends the binary weight networks and introduces binary into activation calculations, using binary operations to approximate convolution operations to reduce time and memory. The essential target of quantization is to determine a mapping for the network parameters and intermediate values to a finite set of values in a lower precision range. Such range can be symmetric and asymmetric, and the mapped values can be uniformly or non-uniformly spread in the range according to the characteristics of the network [25].

There are different schemes for model quantization in real applications according to the stage to apply this method. In Quantization-Aware Training [25], the quantization is implemented after the original model is pre-trained on the dataset. Consider that quantization will inevitably make the trained parameters deviate from the training results and lead to accuracy degradation. The quantized model is then retrained on the training data to recover model accuracy. However, such re-training could take a relatively long period, especially for low-bit precision quantization. The other scheme is Post-Training Quantization [25], where quantization takes place after the pre-trained model is calibrated with a small subset of training data to compute the clipping ranges and the scaling factors. Then, the model can be directly quantized without further tuning. Fig 2.6 compares the process of these schemes.

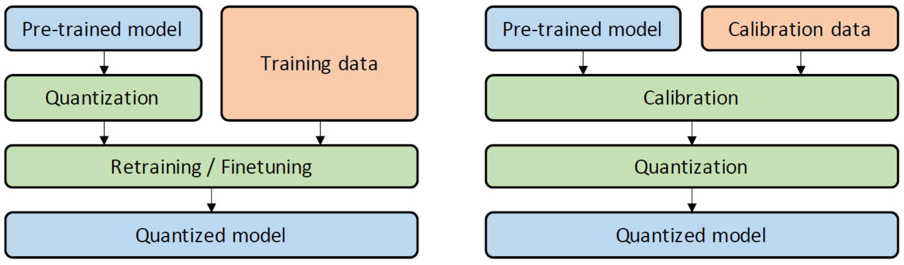


Figure 2.6: **Quantization-Aware Training (Left) and Post-Training Quantization (Right) [25]**

2.4.2 Model Pruning

Research on pruning neural network models is also based on the same assumption that there is significant redundancy in the parametrization of deep learning models [20]. In neural networks, there are many redundant, non-informative parameters in the deep learning network model in the convolutional and fully connected layers [13], and many neuron activation values are close to 0, which indicates that only a fraction of the parameters participate in the main calculation, while the rest not contributing much to the network performance during training. Network pruning removes such parameters from the network that these simplifications will have the least effect on the accuracy of the network [14]. Fig 2.7 provides an example of the pruning process.

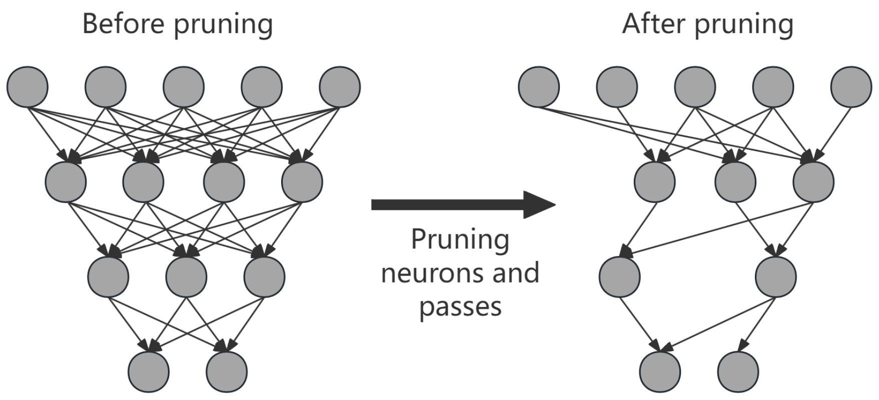


Figure 2.7: **Model pruning. Removing redundant or less important components, such as neurons or weights, without significantly compromising the model's performance**

Given a neural network$f(X, W)$, with X as the input data and W as the set of parameters, model pruning can be formulated as a technique for determining a minimal subset of network parameters W' such that the rest of the parameters in W are removed or set to zero while guaranteeing that the performance of the network model remains to be able to satisfy the required threshold [45].

Most of the methods apply a three-step process for the complete model pruning process[[5],[30]:

- Train the original network connectivity via normal network training for network convergence. However, unlike usual training, this network training step focuses on determining which connections are important.
- Prune the low-weight connections. All the connections with weights below a certain threshold are removed to convert a dense network to a sparse network.
- Retrain the network. Pruning will inevitably have an impact on the model's performance. The sparse network is retrained with the remaining parameters on the target task to recover the accuracy.

The second and the third step can be repeated till the pruned model meets the requirements.

According to the granularity of pruned structures, model pruning methods can be categorized as structured and unstructured pruning. Unstructured pruning, or fine-grained pruning, does not follow a specific geometry or constraint and treats all the parameters equally to perform pruning directly on the weight parameter level [1]. Unstructured pruning has great flexibility and simplicity, but in many cases, it requires specific hardware or library support for realistic acceleration because of the unstructured sparsity [43].

In comparison, structure pruning focuses on modifying networks at a higher granularity, such as entire neurons, kernels [2], filters [33], or channels [34]. In channel-level pruning, for example, all the incoming and outgoing weights to/from a feature map are pruned [1]. Structure pruning will reduce the demand for calculations and generate lightweight intermediate representations, further reducing the demand for memory. However, such methods often require more delicacy in designing as the network models are contacted by different layers, and their inputs and outputs are correlated.

The criteria for determining which parameters or elements of the network are important are also a central question in network pruning. One common scoring method to evaluate whether a weight parameter is important is its magnitude [43]. Under the constraint of weight decay, those weights that do not contribute significantly to the result will shrink in magnitude during training. Applying an L_0 or L_1 regularizer to the model loss function during training can force some of the weight parameters to 0, creating sparsity in network models to achieve pruning [61]. In structure pruning, L_1 and L_2 norms can also be used in structured pruning to measure the importance of pruning units like channels [61].

2.5 Knowledge Distillation

One technique used to address this challenge is knowledge distillation. Knowledge distillation extracts information, also referred to as knowledge, from the larger models and transfers it to a smaller model in order to guide its training process. This enables the smaller model to achieve accuracy and generalization capability comparable to that of the larger model. Knowledge distillation is classified into two main categories, feature-based and logit-based methods, depending on the type of knowledge used [27]. Logit-based methods employ the logit output of the complex model, i.e., the output values that have not undergone the softmax process, as knowledge. In contrast, feature-based methods utilize the information from the intermediate layers of the complex network as knowledge.

The classic knowledge distillation method was proposed by Jeffery Hinton et al. [35]. in 2015. This method involves a novel training process for small neural networks, where the authors used the logit values of a pre-trained large model as transferred knowledge to guide the training of a smaller model to mimic the output of the larger model. This approach enables the smaller model to learn the generalization ability of the pre-trained complex model.

The traditional goal of neural network training is to maximize the output probability value for the target class, using ground truth as the only source of information during training and treating the output probability values for non-target classes equally irrelevant. However, non-target classes' probability output values also contain valuable information [35]. For instance, the model's output probability value for certain non-target classes might be higher than for

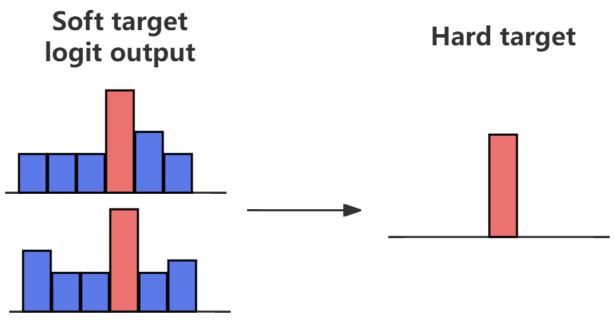


Figure 2.8: **The same hard label result may originate from different soft labels**

others, as demonstrated in Fig 2.8. These differences may correspond to different features in the classifying problem. This information is crucial for improving the model's generalization ability because it includes more comprehensive information about the features used for classification. This part is also referred to as "dark knowledge."

A well-trained and accurate network model is designed to maximize the probability value for the target class, resulting in very small or near-zero probability output values for non-target classes, which makes it difficult to use this information. To solve this issue, the researchers introduced a "softening" approach that incorporates the concept of temperature T into the normal softmax operation. This method alleviates the problem by smoothing the probability distribution, allowing the teacher model to transfer more information about non-target classes [35]. $p(z_i, T)$ represents the softened softmax output under temperature T, and it is calculated with the fraction of $exp(z_i/T)$ and $\sum_j exp(z_i/T)$, where z_i is the original logit value input.

Through the influence of T, the output probability distribution of the softmax output tends to be smoother, and the entropy of the distribution is larger, leading to a greater emphasis on non-target classes as negative labels carry relatively more dark knowledge.

With such soft targets, Hinton et al. [35]. used the teacher-student architecture to train the model, using the well-trained complex network as the teacher to guide the training of the small network as the student. To ensure that the model correctly classifies the target class when learning from the teacher model, ground truth is still used as part of the training target during the training process. The total loss for supervised training with knowledge distillation is the combination of the loss of ground truth labels and soft targets.

As the first knowledge distillation method, the concepts and training configuration introduced by the classic knowledge distillation method become the basis of the later knowledge distillation methods.

The other major category of knowledge distillation is to make use of the features generated by the intermediate layers of the teacher network as guidance signals. For example, a spatial attention map represents the degree of response of each layer of the network to different regions of the input, reflecting the de-

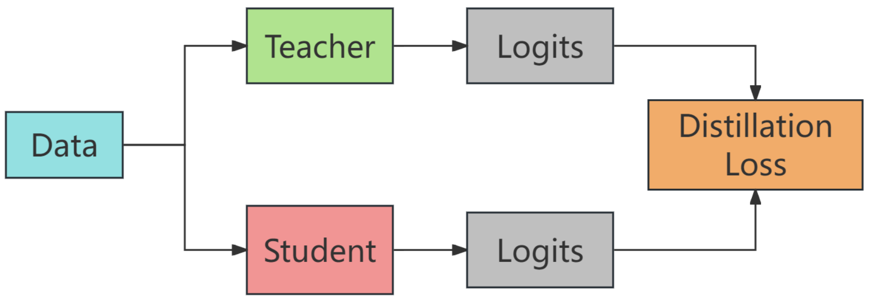


Figure 2.9: **Architecture of logit-based knowledge distillation. The logit-based method uses the logit output of the teacher and student network to perform knowledge distillation.**

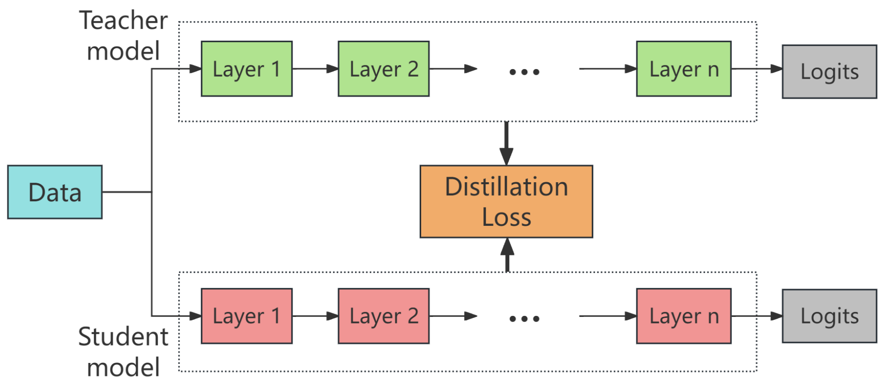


Figure 2.10: **Architecture of feature-based knowledge distillation. Feature-based models use features generated by the intermediate layers to extract knowledge and have the student learn from these guidance signals.**

gree of activation of each neural network layer to different features in the input. The network's response to the input and the predicted object has a spatial correlation, and the more accurate the network is, the stronger the correlation. Attention transfer [69] takes these attention maps to conduct layer-wise knowledge distillation and extract more helpful information from the teacher network of different semantic levels.

Relation-based knowledge distillation methods are sometimes regarded as a variation of feature-based knowledge distillation methods. Instead of focusing only on the final output or class probabilities of the teacher model, this method category emphasizes on exploring the relationships between the layers or data samples [27]. Such relationships include similarities, distances, or other forms of correlations between data points in the feature space. RKD constructs the distance-wise and anglewise relations within one training data batch and calculates the distillation loss accordingly to transfer such relational knowledge to the student model [52].

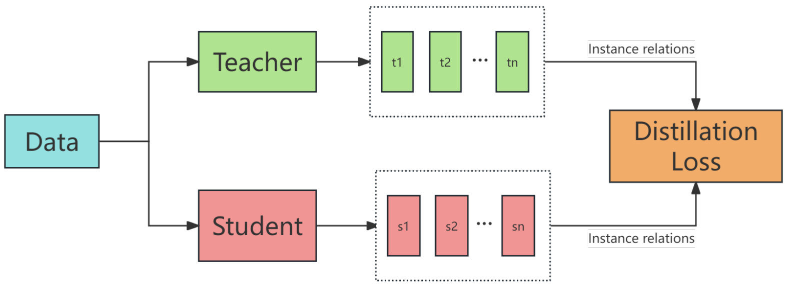


Figure 2.11: **Architecture of relation-based knowledge distillation. Relation-based distillation leverages structural or relational information within data samples, layers, and representations.**

Some existing research studies have tried to apply contrastive learning strategies in knowledge distillation. SSKD [65], as demonstrated in Fig 2.12, proposed using feature representations generated by the teacher model from the augmented input data as secondary knowledge to guide the training of the student network. SimCLR V2 [9] proposed that after the self-supervised retrained model is fine-tuned on the downstream task with a small fraction of labeled data, the teacher network can impute labels for training a student network with the unlabeled data, as demonstrated in Fig 2.13. Based on the MoCo V2 framework [10], SEED [22] models the similarity over the queue of the student and teacher model and has the student mimic the similarity score distribution inferred by the teacher model. DisCo [24] combines the self-supervised learning of a student model and has the student model learn from the pre-trained teacher model by forcing the last embedding of the student to be consistent with that of the teacher.

In comparison, model pruning methods and model quantization may require more training iterations and labeled data for retraining. Knowledge distillation tends to be more stable and less prone to drastic performance loss as it is training a complete network model while pruning and quantization may need a more careful design for hyper-parameters and processes for convergence and to avoid accuracy loss. Knowledge distillation can be applied to a wide range of model architectures and is not limited by the structure of the original model. Furthermore, student models trained by knowledge distillation can be more easily fine-tuned for other tasks or new domains, leveraging the knowledge transferred from the teacher for transfer learning cases. Therefore, an unsupervised knowledge distillation method is more suitable and efficient in this scenario, especially when pre-trained large teacher models on the large-scale training dataset are available.

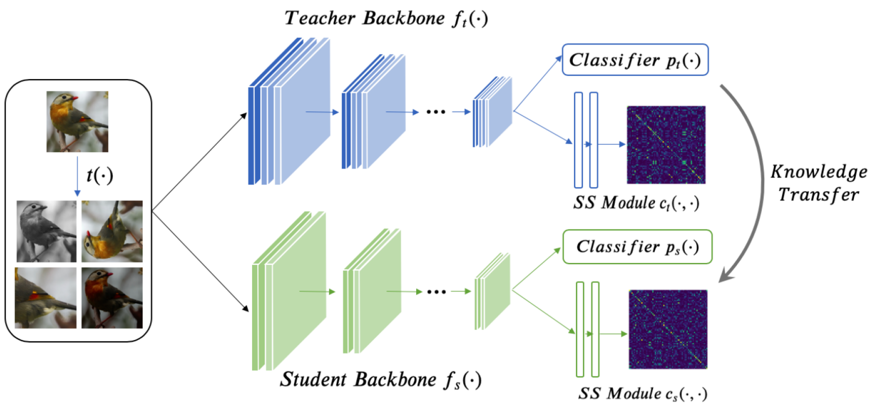

Teacher Backbone $f_t(\cdot)$
Classifier $p_t(\cdot)$
$t(\cdot)$
SS Module $c_t(\cdot,\cdot)$
Knowledge Transfer
Classifier $p_s(\cdot)$
Student Backbone $f_s(\cdot)$
SS Module $c_s(\cdot,\cdot)$

Figure 2.12: **Architecture of SSKD[65]. An SS module is added beside the normal supervised classifier as an auxiliary task. The classifier output of the teacher model, as well as the contrasive output of the SS module with respect to the transformed data, are used as knowledge transferred from the teacher model to guide the training of the student model**

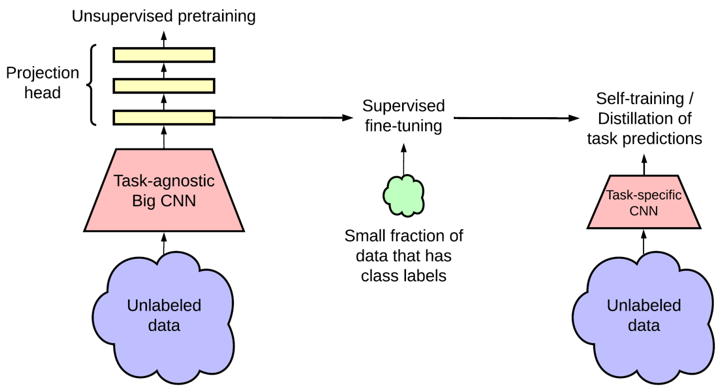


Figure 2.13: **Semi-supervised learning framework the research[9]. The Task-agnostic big CNN network is fine-tuned on a small fraction of the labeled data. Then, the network can be used as a teacher network to input labels for training a student network on an unlabeled dataset.**

Chapter 3

Method

This chapter provides a detailed introduction to the knowledge distillation method proposed in this thesis. Chapter 3.1 first provides an overview of the components and structure of the method. Chapter 3.2 formally formulates the research problem addressed by this knowledge distillation method. Chapter 3.3 discusses the pre-text task utilized of self-supervised learning used in this method and the motivation for extracting such knowledge. Chapter 3.5.1, and Chapter 3.5.2 elaborate in detail the three steps of the training process for this method, with the introduction of the process of each stage, formulation of the training loss function and the pseudo codes.

3.1 Overview

This chapter proposes an unsupervised knowledge distillation method using the networks' intermediate and final embedding outputs under the augmented inputs. The pre-trained teacher network learns to differentiate similar and dissimilar samples in the augmented inputs. When the teacher and student models are fed with the same input data, the teacher can guide the student in determining which samples are essentially similar via similarity relations the teacher generates in the feature embedding from different levels. The overall training scheme is illustrated in Fig 3.1.

The most crucial task of a knowledge distillation process is to determine the appropriate knowledge to be extracted from the teacher model and transferred to the student model for guiding its training. In the cases of supervised learning, feasible knowledge choices could be the logit output of the teacher model [35] or combine it with the features from the intermediate layers[[69] citeromero2015experimental]. However, the task setting of lacking ground truth labels from the training dataset would require task-agnostic feature knowledge to be utilized in training. Also, the specific downstream classification tasks where the knowledge learned from the unlabeled training dataset would be applied remain unknown. The teacher model with a contrasive learning method (SimCLR) learns the instance discrimination knowledge among the training samples in the large unlabeled training set and transfers the relational knowledge to the student model. Therefore, the final layer embedding and the feature embedding from the intermediate layers are chosen to be supervision signals for the knowledge distillation process, and the following components are designed to realize the knowledge distillation process.

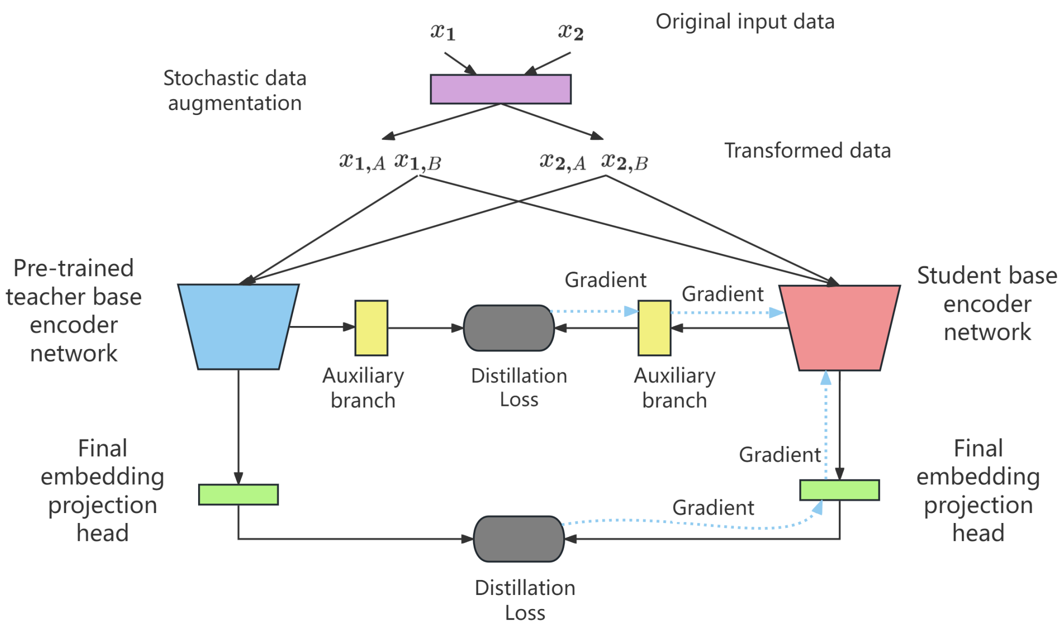


Figure 3.1: **Overview of the architecture of the proposed distillation method.**

- **A pre-trained teacher base encoder network** to provide knowledge for distillation. This network model is trained beforehand on the same unlabeled training set with contrastive learning methods to learn the underlying feature representations of the training data so that it could provide knowledge to guide the training of the smaller network.

- **A stochastic data augmentation module** provides two different random data transform groups for each given input data sample. This module is designed to provide two augmented views of the same original input data sample in order to form positive and negative pairs within each batch of the training data for contrastive learning.

- **Auxiliary branches** are small neural networks designed to perform knowledge distillation between the intermediate layers of the encoder networks. These smaller networks take the intermediate layer feature output of the encoder networks, project the intermediate features to a latent space, and output the embedding of the input intermediate features.

- **A student base encoder network** is the training target of the knowledge distillation process. The student encoder network is updated with the loss calculated from the corresponding knowledge distillation signals.

- **Final embedding projection heads** are smaller neural networks that receive the final feature representation output of the encoder networks and map it to the embedding space to generate the corresponding final embedding. The projection head from the contrastive learning training of the pre-trained teacher base encoder network is preserved and utilized in this method. The projection head of the student is randomly initialized and has the same structure as the projection head of the teacher network.
- **Distillation Loss** is the loss function for comparing the teacher and student network final projection head and auxiliary branches output. The student network is updated with the optimization goal of goal of minimizing this loss function.

3.2 Formulation of the research problem

The research problem of this thesis can be formulated as follows: Given an unlabeled dataset D and a complex network model M trained on dataset D with SimCLR, the research goal is to develop an unlabeled knowledge distillation method that can train a more light-weight network model S so that the model S is able to achieve comparable learning performance on the training dataset D and better transfer performance on the downstream tasks and has less computational resource demands and shorter response time, using the intermediate layer feature embeddings of the complex network and the final embedding trained by SimCLR of the complex model M.

3.3 Pre-text task

In self-supervised learning, the pre-text tasks are pre-defined tasks the network models are trained to solve to extract and learn useful visual features from the attributes of the unlabeled training data [71].

The method proposed in this chapter uses instance discrimination as the pre-text task to perform the knowledge distillation process. Without the truth labels, instance discrimination trains the network models based on the standard that embedding features of the different data augmentations originating from the same instance should be invariant. It encourages the network to maximize the similarity between augmented views of the same instance while the features of different instances should be spread out [66].

The teacher encoder network and its final projection head are trained with contrastive learning with instance discrimination. Before performing knowledge distillation, the auxiliary branches attached to the teacher network also need to be trained on top of the trained teacher encoder network with the same instance discrimination. Therefore, when receiving augmented data inputs, the teacher network and the attaching auxiliary branches are capable of using the previously learned features knowledge to generate intermediate and final embedding outputs that differentiate the similar and dissimilar data samples based on their intrinsic features.

Trained with transform-invariant features with contrasive learning, these embedding outputs are more robust to variations in the input data. They are able to help the student network learn better feature representations from the training data for generalization and to be transferred to downstream tasks. While it is generally believed that the final embedding of the projection head output contains the most fruitful knowledge [24], features from intermediate layers could also be helpful in learning from the training data, and we hope to explore their performance in unsupervised knowledge distillation. Intermediate layers can capture a wider range of features, from low-level details to mid-level abstractions, some of which may not be visible to the final layers [50]. The shallow layers learn low-level features such as colors and edges and are more general features, while the deeper layers tend to learn more high-level task-related semantic features such as categorical knowledge for specific tasks [76]. Research of A also indicates that in contrasive learning using instance discrimination, low-level and mid-level representations make great contributions in transfer learning to downstream tasks [73].. Using these intermediate layer embeddings can provide a richer and more detailed representation of the data.

As demonstrated in Fig 3.2, the positive-concentrated and negative-separated features in the final embedding and intermediate layer embedding of the trained teacher encoder network obtained from the contrasive learning pre-training can be modeled by the pairwise-instance similarity relations of the augmented views within the same data batch. Such relations can be regarded as a form of knowledge distilled from the teacher and transferred to guide the training of the student model. By forcing the student model to mimic the similarity matrices of the teacher model, the student model acquires the ability to discriminate different instances with their intrinsic features.

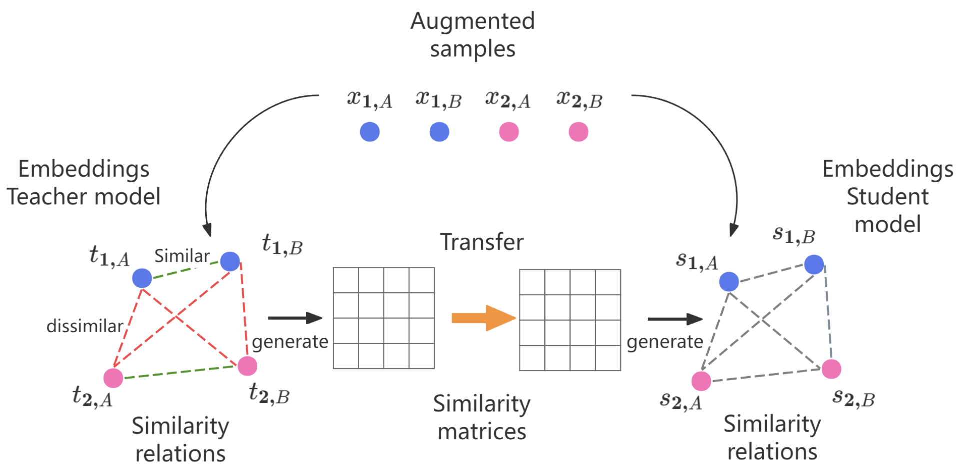


Figure 3.2: $x_{1,A}$ **and** $x_{1,B}$ **are two augmented views of** x_1 **and** $x_{2,A}$ **and** $x_{2,B}$ **are two augmented views of** x_2**. The knowledge is transferred in the form of forcing the student network to match the similarity matrices of the teacher model with respect to the same input data so as to generate embedding containing similarity relations of the augmented data instances similar to the teacher model**

3.4 Auxiliary branches

In order to extract feature embedding from the intermediate layers of the trained teacher model, a group of several small sub-networks called auxiliary branches are attached to the intermediate layers of the encoder networks to acquire their outputs. Such subnetworks have similar functions as the projection head in contrasive learning. Each auxiliary branch is made up of multiple building blocks, and each building block is formed by two separable depth-wise convolution modules [36]. The separable depth-wise convolution process divides the convolution process into two steps, depthwise and pointwise convolution, and reduces the number of parameters and operations to lower the computing complexity. A second separable depth-wise convolution is used in the building block to increase the non-linearity of the auxiliary branches. The auxiliary branches are attached after the end of each convolution stage of the network models. As the features extracted from different depths come with different semantic features, deeper feature representations contain more features from the training data of a higher abstract level, and such disparity also follows as the features are projected into the latent space. Therefore, the depth of the auxiliary branches reduces as the depth of the intermediate features increases.

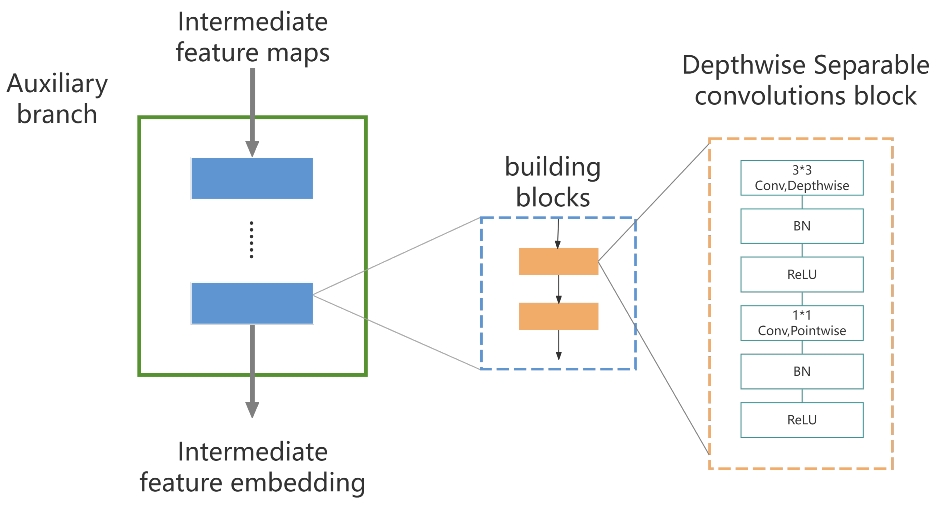

Intermediate feature maps
Auxiliary branch
Depthwise Separable convolutions block
building blocks
3*3 Conv,Depthwise
BN
ReLU
1*1 Conv,Pointwise
BN
ReLU
Intermediate feature embedding

Figure 3.3: **Structure of the auxiliary branches**

3.5 Training Details

This section describes the detailed process of the proposed knowledge distillation method training.

3.5.1 Training of the Teacher Auxiliary Branches

In order to carry out the knowledge distillation, one indispensable element is a well-trained teacher model. The teacher model is trained by the SimCLR learning method, following the procedure introduced in the previous chapter 2.3. Different from the normal SimCLR training process, where the projection head is usually disposed of after the training, this method preserves it for use in the subsequent stages.

Now, the teacher backbone model is trained with the aim of maximizing the agreement between positive pair elements in the latent space to learn the features of the training set. In this step, intuitively, we would like to preserve the outcomes of learning from the training and extract the informative representations from the trained model. Accordingly, the teacher backbone model is frozen and was not updated during the training at this stage. The training only includes the auxiliary branches. We optimized the auxiliary branches with the same target as the previous step. Similarly, we take the mini-batch of N data

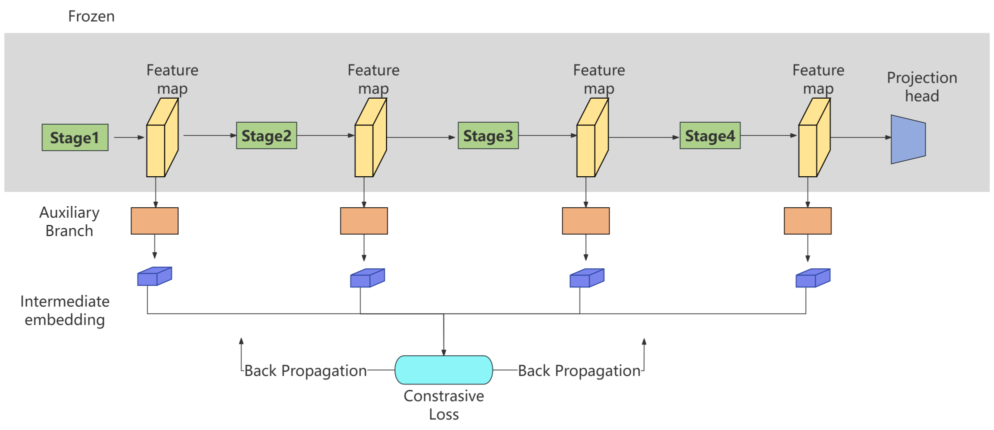


Figure 3.4: **Training of the teacher auxiliary branches**

samples $\{x_i\}_{i=:N}$, and 2 transforms t, t' sampled from a random distribution T to derive the two augmented views $\widetilde{x_i} = t(x)$ and $\widetilde{x_j} = t'(x)$ and 2N input images. Each pair of augmented vies originating from the same input image is defined as a positive pair, and each transformed data sample forms negative pairs with the rest of the 2(N-1) data samples. With such data organization, we use the instance-discrimination style input to train the auxiliary branches.

Define the M output feature representations from auxiliary branches attached to the teacher network as $\{v_i\}_{i=:M}$. The derivation of these outputs can be expressed as follows:

$$v_k(x_i) = c_k(f_k(x_i)) \tag{3.1}$$

$$v_k(x_j) = c_k(f_k(x_j)) \tag{3.2}$$

where $f_k(x)$ is the output feature of the frozen backbone network up to the k^{th} convolutional stage and $c_k(x)$ is the k^{th} auxiliary branch. Although there are different measurements for the similarity between feature representations, we stick to the cosine similarity as in equation 2.2 for consistency. For the k^{th} auxiliary branch, the contrasive loss function with respect to the positive pair (x_i, x_j) can be expressed as:

$$\ell_{k,i,j} = -log\frac{exp(sim(v_{ki}, v_{kj})/\tau)}{\sum_{k=1}^{2N} \mathbb{1}_{[k\neq j]} exp(sim(v_{ki}, v_{kj}))} \tag{3.3}$$

where

$$sim(\boldsymbol{v}_{kj}, \boldsymbol{v}_{kj}) = \boldsymbol{v}_{ki}{}^{\top}\boldsymbol{v}_{kj}/||\boldsymbol{v}_{ki}||||\boldsymbol{v}_{kj}|| \tag{3.4}$$

and $\mathbb{1}_{[k \neq j]}$ is an indicator function evaluating to 1 iff $k \neq i$. The temperature coefficient τ has identical effects as that in equation 2.1, either smoothing or enhancing the contrasts between negative paired samples.

Again, the total contrasive loss for each auxiliary branch is computed across all the positive pairs.

$$\mathcal{L}_k = \frac{1}{2N}\sum_{k=1}^{N}[\ell_k(2k-1, 2k) + \ell_k(2k, 2k-1)] \tag{3.5}$$

The final loss is the sum of the losses from each individual branch:

$$\mathcal{L} = \sum_{k=1}^{M}\mathcal{L}_k \tag{3.6}$$

The detailed algorithm for the knowledge distillation process is elaborated in 1.

Algorithm 1 learning algorithm of auxiliary branches

Input: N: batch size, τ:temperature, f: base encoder, c: auxiliary branches, T: distribution of augmentations.
Output: trained auxiliary branches c($\cdot$)

```
1: for the input mini-batch $\{x_k\}_{k=1}^{N}$ do
2:   for $k \in \{1, \ldots, N\}$ do
3:     sample two augmentation functions $t \sim T$ and $t' \sim T$
4:     $\tilde{\mathbf{x}}_{2k-1} = t'(\mathbf{x}_k)$
5:     $\tilde{\mathbf{x}}_{2k} = t(\mathbf{x}_k)$
6:     for $m \in \{1, \ldots, M\}$ do
7:       $v_{2k-1,m}(\tilde{\mathbf{x}}_{2k-1}) = c_m(f_m(\tilde{\mathbf{x}}_{2k-1}))$
8:       $v_{2k,m}(\tilde{\mathbf{x}}_{2k}) = c_m(f_m(\tilde{\mathbf{x}}_{2k}))$
9:     end for
10:  end for
11:  for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ do
12:    $\mathbf{s}_{i,j} = \mathbf{v}_i{}^{\top}\mathbf{v}_j / \|\mathbf{v}_j\| \|\mathbf{v}_j\|$
13:  end for
14:  define $\ell_m(i,j) = -log\frac{exp(\mathbf{s}_{i,j}(v_{i,m}, v_{j,m})/\tau)}{\mathbb{1}_{[k \neq j]}exp(\mathbf{s}_{i,j}(v_{i,m}, v_{j,m}))}$
15:  $\mathcal{L}_m = \frac{1}{2N}\sum_{k=1}^{2N}[\ell_m(2k-1, 2k) + \ell_m(2k, 2k-1)]$
16:  $\mathcal{L} = \sum_{m=1}^{M}\mathcal{L}_m$
17:  update the auxiliary branches c wrt. the loss
18: end for
19: return trained auxiliary branches c($\cdot$)
```

3.5.2 Training of the Student Model

With the complete teacher model and auxiliary modules for knowledge distillation trained and prepared, we can now proceed to the most significant step of the method: knowledge distillation, where the teacher transfers what he has learned from the training data to guide the student model.

The knowledge we would like to distill from the teacher model is the similarity responses for the augmented data inputs at different depths of the teacher network, and it's achieved in the form of a similarity matrix of the samples in the mini-batch calculated by the feature representations of the intermediate layers and final projection head in the latent space of the teacher network. The similarity responses indicate how the teacher network recognizes similar augmented samples and differentiates distinct ones and, correspondingly, the underlying features in the training data.

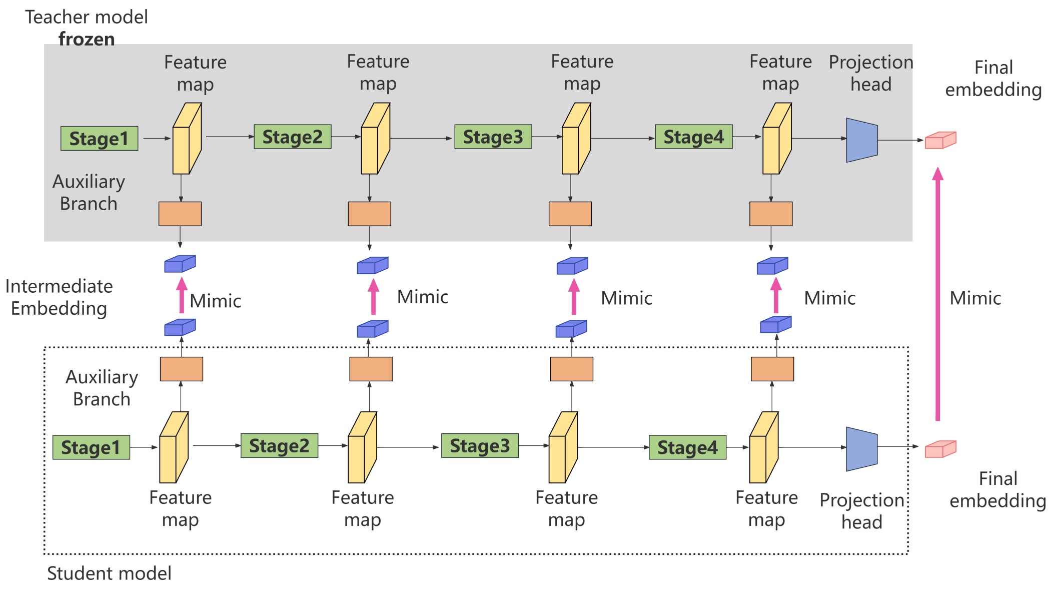


Figure 3.5: **Knowledge distillation training of the student model**

the training goal was to have the student model mimic these guidance signals to learn the teacher model behaviors and responses under the same input. Thus, the same configuration of the auxiliary branches was also applied to the student model accordingly to extract the corresponding feature representations. The same batch of augmented pictures is fed to the teacher as well as the student network, after which the feature representation outputs from the teacher and student model can be collected for comparison. The final optimization goal is to minimize the difference between the corresponding feature representations from the student and teacher network.

Once again, the input data samples are augmented by the two random transforms to form positive pairs and negative pairs, as described in chapter 2.5. The teacher and student model share the same input all the time. Let z_i and z_j be the output of a random auxiliary branch or a projection head from either the teacher model or the student model with respect to a random pair of data samples (x_i, x_j) in a data batch. Then, the output similarity matrix of this auxiliary branch or projection head is defined as follows:

$$\mathcal{A}_{i,j} = \boldsymbol{z}_{\mathbf{i}}^{\top} \boldsymbol{z}_j / ||\boldsymbol{z}_i|| \, ||\boldsymbol{z}_j|| \tag{3.7}$$

Apply softmax on the similarity matrix $\mathcal{A}$ with temperature scale τ to the similarity matrix along the row dimension, leading to a probability matrix $\mathcal{B}$. The loss between a similarity matrix of an auxiliary branch or the projection head of the teacher model $\mathcal{B}^T$ and the matching similarity matrix of the student matrix $\mathcal{B}^S$ is computed with the KL divergence:

$$\ell = -\tau^2 \sum_{i,j} \mathcal{B}^T_{i,j} log(\mathcal{B}^S_{i,j}) \tag{3.8}$$

So, the loss between the M auxiliary branch pairs:

$$\mathcal{L}_{aux} = \textstyle\sum_{k=1}^{M} \ell_k \tag{3.9}$$

where ℓ_k is the KL divergence between the similarity matrix of the k^{th} auxiliary branches of the teacher and student model.

The total loss is calculated as the sum of the losses from auxiliary branches and that from the projection head:

$$\mathcal{L} = \mathcal{L}_{aux} + \ell_{proj} \tag{3.10}$$

where ℓ_{proj} is the KL divergence between the similarity matrix of the projection heads of the teacher and student model. The detailed algorithm for the knowledge distillation process is elaborated in 2.

Algorithm 2 learning algorithm of student model

Input: N: batch size, τ:temperature, f^T: base encoder of teacher model, c^T: auxiliary branches of teacher model,g^T: projection head of the teacher model, f^S: base encoder of student model, c^S: auxiliary branches of student model,g^S: projection head of the student model, T: distribution of augmentations.

Output: trained student model encoder model f^S

```
 1: for the input mini-batch $\{x_k\}_{k=1}^N$ do
 2:   for $k \in \{1, \ldots, N\}$ do
 3:     sample two augmentation functions $t \sim T$ and $t' \sim T$
 4:     $\tilde{\mathbf{x}}_{2k-1} = t(\mathbf{x}_k), \tilde{\mathbf{x}}_{2k} = t'(\mathbf{x}_k)$
 5:     for $m \in \{1, \ldots, M\}$ do
 6:       $v^T_{2k-1,m}(\tilde{\mathbf{x}}_{2k-1}) = c^T_m(f^T_m(\tilde{\mathbf{x}}_{2k-1})), v^T_{2k,m}(\tilde{\mathbf{x}}_{2k}) = c^T_m(f^T_m(\tilde{\mathbf{x}}_{2k}))$
 7:       $v^S_{2k-1,m}(\tilde{\mathbf{x}}_{2k-1}) = c^S_m(f^S_m(\tilde{\mathbf{x}}_{2k-1})), v^S_{2k,m}(\tilde{\mathbf{x}}_{2k}) = c^S_m(f^S_m(\tilde{\mathbf{x}}_{2k}))$
 8:     end for
 9:     $\mathbf{z}^T_{2k-1} = g^T(f^T(\tilde{\mathbf{x}}_{2k-1})), \mathbf{z}^T_{2k} = g^T(f^T(\tilde{\mathbf{x}}_{2k}))$
10:     $\mathbf{z}^S_{2k-1} = g^S(f^S(\tilde{\mathbf{x}}_{2k-1})), \mathbf{z}^S_{2k} = g^S(f^S(\tilde{\mathbf{x}}_{2k}))$
11:   end for
12:   for all $i \in \{1, \ldots, 2N\}$ and $j \in \{1, \ldots, 2N\}$ do
13:     $\mathbf{A}_{i,j} = \mathbf{v}_i{}^\top \mathbf{v}_j / \|\mathbf{v}_j\| \|\mathbf{v}_j\|$ for $\mathbf{v}^T$ from teacher network and $\mathbf{v}^S$ from
        student models.
14:     $\mathbf{A}_{i,j} = \mathbf{z}_i{}^\top \mathbf{z}_j / \|\mathbf{z}_j\| \|\mathbf{z}_j\|$ for $\mathbf{z}^T$ from teacher network and $\mathbf{z}^S$ from stu-
        dent models.
15:   end for
16:   for $m \in \{1, \ldots, M\}$ do
17:     $\mathcal{B}^T_m = softmax(\mathcal{A}^T_m, \tau), \mathcal{B}^S_m = softmax(\mathcal{A}^S_m, \tau)$
18:   end for
19:   $\mathcal{B}^T_{proj} = softmax(A^T_{proj}, \tau), \mathcal{B}^T_{proj} = softmax(A^S_{proj}, \tau)$
20:   $\mathcal{L}_{aux} = \sum_{m=1}^{M} KL\_div(\mathcal{B}^T_m, \mathcal{B}^S_m)$
21:   $\ell_{proj} = KL\_div(\mathcal{B}^T_{proj}, \mathcal{B}^S_{proj})$
22:   $\mathcal{L} = \ell_{proj} + \mathcal{L}_{aux}$
23:   update the student encoder network, student auxiliary branch, and stu-
      dent projection head wrt. the loss
24: end for
25: return trained student encoder network and discard student auxiliary
    branches and student projection head
```

Chapter 4

Evalutaion

Chapter 4 demonstrates the implementation details of the evaluations and the results. Chapter 4.1 demonstrates the datasets used for the evaluation process. Chapter 4.2 elaborates on the detailed implementations of the experiment models and specific experiments. Chapter 4.3 introduces the evaluation protocols used for the experiments. Chapter 4.4 demonstrated all the compared methods. The evaluation results of linear evaluation on the evaluation datasets are presented in Chapter 4.5. In Chapter 4.6, the results of transfer learning on the downstream tasks are presented. The ablation study in Chapter 4.7 explored the influences of the hyperparameters in the method. The system performance of the tested models is analyzed on multiple devices in Chapter 4.8.

4.1 Evalution datasets

In this chapter, the datasets used in the evaluation are introduced.

4.1.1 STL-10

The STL-10 [17] dataset is an image recognition dataset dedicated to self-supervised learning and unsupervised feature learning. In this dataset, items are split into three parts. There are 5000 pictures in the labeled training set and 8000 labeled pictures in the test set, which are uniformly distributed among ten classes, including airplane, bird, car, cat, deer, dog, horse, monkey, ship, and truck. The third split, the unlabeled set, contains 100000 unlabeled pictures from the ten classes mentioned above, plus other vehicle and animal categories. All the items in this dataset are with a resolution of 96×96 and colored pictures. This dataset strikes a balance between dataset size and complexity, and the dedicated unlabeled set makes it ideal to conduct experiments to test the proposed unsupervised knowledge distillation method.

4.1.2 ILSVRC2012

In this experiment, we use the dataset from ImageNet Large Scale Visual Recognition Challenge 2012 [56] as the task dataset. This is a subset of the largest picture database, Imagenet [19], with over 1.2 million pictures from 1000 categories. The ILSVRC2012 dataset is one of the most widely used datasets in the field of image recognition. It contains three target tasks: classification, classification with localization, and fine-grained classification, but this experiment only focuses on the classification task. The dataset has a training set of 1281167 pictures, a validation set of 50000 pictures, and a test set of 100000 items. The elements in the dataset come in different sizes, so dedicated pre-processing is necessary.

4.1.3 Transfer learning datasets

We evaluate our representation on other classification datasets to assess how well the features learned on the ILSVRC2012 dataset learned with different methods can be transferred to other downstream classification tasks. The evaluated transfer learning datasets include: Food-101 [6], Flowers [47], FGVC Aircraft [44], the Describable Textures Dataset[16], Oxford-IIIT Pets [53],CIFAR 10 [40], CIFAR 100 [40] and Leaf[15].The leaf dataset has two different classification tasks: binary classification on whether a leaf is healthy and leaf classification. For the DTD dataset, the dataset creators defined multiple train/test splits, and only the first split is used here in this evaluation. The leaf dataset has no specific train/test split, and in this evaluation, a randomly selected portion of 80% of the original data is used for the training set and the rest for the testing set.

Dataset	Classes	Size(train/test)
Food-101	101	75750/25250
CIFAR-10	10	50000/10000
CIFAR-100	100	50000/10000
Oxford-IIIT Pets	37	3680/3369
Oxford 102 Flowers	102	2040/6149
Describable Textures (DTD)	47	3760/1080
Leaf binary health classification	2	3600/903
Leaf classification	12	3600/903

Table 4.1: **Datasets examined in transfer learning**

4.2 Implementation

The evaluation of the method was implemented in a PyTorch environment. The standard ResNet architecture is used in this experiment for the base encoder networks. For the teacher model, ResNet-50 forms the backbone of the teacher network, and the student network contains the ResNet-18 network. Detailed information on the model structure is listed in Fig 4.2 and table 4.2.

Network	Stage 1	Stage 2	Stage 3	Stage 4
ResNet-18	3	3	3	3
ResNet-50	3	4	6	3

Table 4.2: **Number of building blocks of ResNet models at each stage**

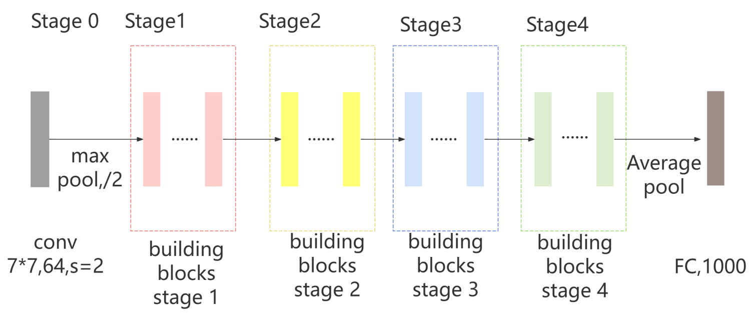
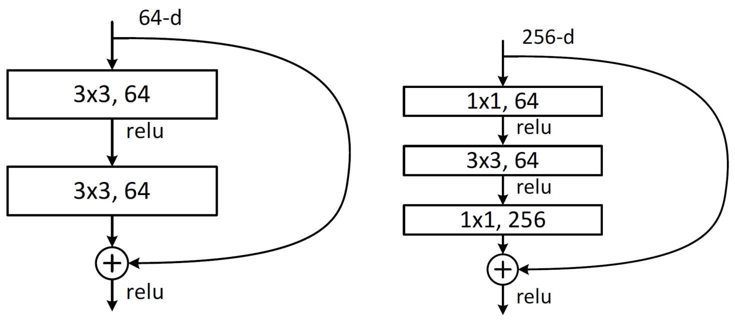


Figure 4.1: **The general architecture of ResNet-18 and ResNet-50 models**



(a) Building block ResNet-18[32] (b) Building block ResNet-50[32]

Figure 4.2: **Building blocks of ResNet models**

4.2.1 Implementation of the experiment models

In terms of the training of the baseline models via SimCLR, the 2-layer non-linear multi-layer perceptron projection heads are applied in the training processes in this paper, which project the feature representations from the backbone networks into a 128-dimension latent space. Other configurations will not be applied here as the original paper indicates that the dimension of the projection head output does not have a substantial influence on the training results[8].

4.2.2 Implementation of the auxiliary branches

The auxiliary branches assisting the knowledge distillation process share the same architecture, while the construction becomes simpler as the intermediate feature of the branch corresponding to it comes from a deeper stage of the network. On top of each auxiliary branch for the ResNet-50 teacher model, a 1x1 convolution layer is applied to reduce the input dimension to match the number of channels of the intermediate feature maps and auxiliary branches.

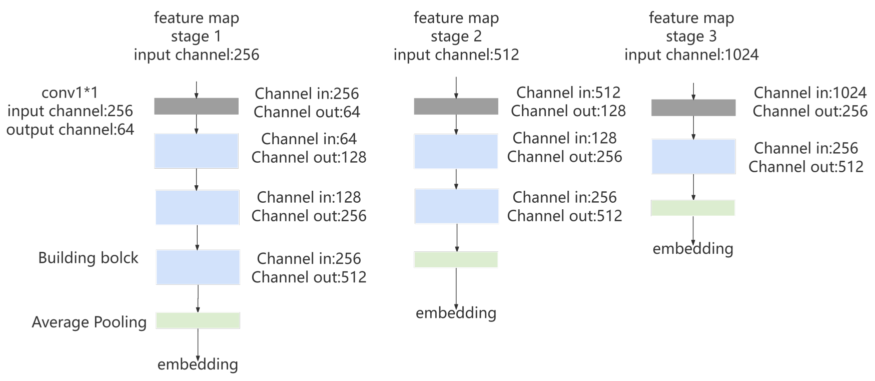


Figure 4.3: **Auxiliary branches for the teacher model**

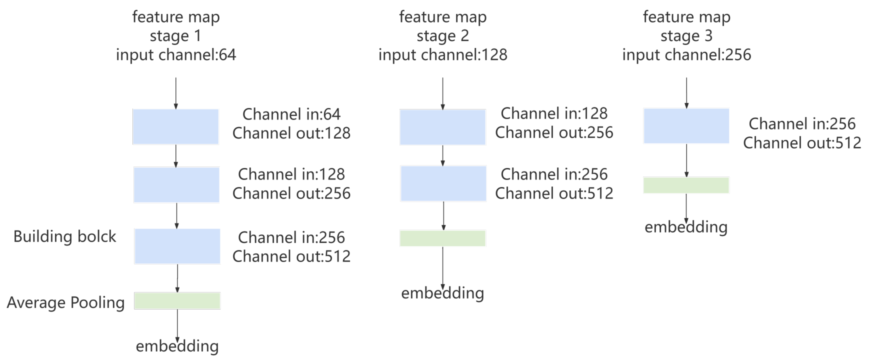


Figure 4.4: **Auxiliary branches for the student model**

4.2.3 Experiment settings of the training on STL-10 data-set

In this experiment, a ResNet-50 teacher model and a ResNet-18 baseline model were trained on an STL-10 unlabeled set with the SimCLR method. With the pre-trained teacher backbone model, the student ResNet-18 model was trained following the process described in Chapter 3, including the training for auxiliary branches of the teacher model. Then, the learned feature representations are evaluated with the linear evaluation protocol [8], extensively used in this scenario for feature learning.

Both SimCLR training for baseline models and knowledge distillation used the same group of data augmentation configurations to process the input data.

- **Random crop, resize, and horizontal flip**. The input picture is randomly resized with the default scale setting (0.08,1) and ratio (3/4,4/3). The final crop will be resized to 96×96 before being fed to the model, followed by a random horizontal flip of the probability of 0.5.

- **Color Jitter**. The transform consists of a color jittering of probability

0.8 and a color dropping of 0.2.

- **Gaussian blur**. Gaussian blur is applied to the input image 50% of the time, and the kernel size is set to be 10% of the image height/weight.

4.2.4 Experiment settings of the training on ILSVRC2012 dataset

This experiment also uses the teacher-student model pair of ResNet50 and ResNet18 networks. A ResNet50 and a ResNet18 model are first trained by SimCLR as baseline models on the ILSVRC2012 training set. Similarly, the auxiliary branches of the teacher model are then trained with the backbone model fixed before the teacher model can be used to carry out the knowledge distillation process. The training of auxiliary branches and knowledge distillation process is also conducted on the ILSVRC2012 training set.

This experiment uses the same data augmentation combination as the previous experiment described in Chapter 4.2.2, with the only change being that the output size of the training data is 224×224.

In the SimCLR training for the baseline models, the first step of the experiment, each training batch contains 256 pictures, and the training continues for 100 epochs. With reference to the results of the paper of SimCLR [8], the training process uses a LARS [67] optimizer with an initial learning rate of 0.6 and weight decay of 10^{-6}. The scheduling of the learning rate follows a linear warm-up strategy, where the learning rate linearly increases to the starting learning rate within the set epochs, 10 in this experiment. After that, it is adjusted by a CosineAnnealingLR scheduler. The temperature coefficient in the loss function is set as 0.1. This set of training settings is also applied in the second step of the experiment, the training for the auxiliary branches.

4.3 Evaluation protocol

In this section, two evaluation protocols are introduced.

4.3.1 Linear evaluation

To evaluate the feature representation learned, we use the linear evaluation protocol. To evaluate the feature learned, the backbone model is frozen, and a linear classifier is trained on top of the frozen encoder network. The linear evaluation accuracy is the classification accuracy of the linear classifier.

4.3.2 Fine-tuning

In the transfer learning experiment, since we are evaluating the features learned from the unlabeled training dataset to be transferred to the specific downstream tasks, the evaluation policy applied in such experiments is fine-tuning. Similarly, a linear classifier for the specific classification task is applied to the backbone network. In the case of fine-tuning, the whole network is trained and updated together for the classification task.

4.4 Compared methods

The target method for evaluation is the multi-layer contrasive knowledge distillation method proposed in this thesis, which will be abbreviated as MLCKD in the following sections. In this method, the pre-trained ResNet-50 network acts as the teacher model and transfers the knowledge learned from the unlabeled training dataset to the ResNet-18 student model, and the learning performance is evaluated subsequently.

The baseline method for comparison is the contrasive learning method SimCLR. The method is applied to both the ResNet-50 network model to evaluate the performance that the teacher model can achieve and also the ResNet-18 network model to evaluate the student model performance when the student model learns from the unlabeled training dataset directly by itself.

4.5 Evaluation results on liner evaluation

In this section the liner evaluation results on the two datasets are demonstrated.

4.5.1 Experiment settings on STL-10 dataset

The linear evaluation training uses a learning rate of 0.0003 and a batch size of 256, and the linear classifier is optimized by an SGD optimizer with momentum set to 0.9. Each linear evaluation process takes 100 epochs of training. At training time, the training data will go through a transformation combination of random crops with resize to 96×96 and random flips. For test time, the test images were resized to 96×96.

4.5.2 Experiment settings on ILSVRC2012 dataset

A linear evaluation protocol is applied to this experiment to evaluate the effectiveness of feature learning. The linear evaluation training uses a learning rate of 0.8 and a batch size of 2048, and the linear classifier is optimized by an SGD optimizer with momentum set to 0.9. Each linear evaluation process takes 100 epochs of training. At training time, the training data will go through a transformation combination of random crops with resize to 224×224 and random flips. For test time, the test images were resized to 256 pixels along the shorter side and took a 224×224 center crop. The temperature coefficient for the training of the teacher auxiliary branches is set to 0.1, and the temperature coefficient for knowledge distillation is set to 0.5.

4.5.3 Liner evaluation results

Comparing the ResNet18 model trained by the proposed multi-layer contrasive knowledge distillation method and the baseline method, the model trained by knowledge distillation achieves a better performance on both the STL-10 dataset and ILSVRC2012 dataset on the linear evaluation with an improvement in the accuracy of 3.24% and 2% respectively, which indicates that the knowledge distillation method is more effective than the model learning the features of the unlabeled training data by the student model learning the unlabeled data itself using SimCLR.

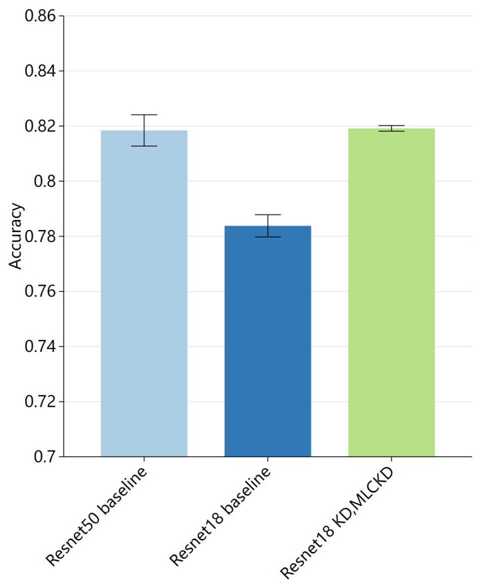


Figure 4.5: **The linear evaluation results on STL-10 dataset.**

4.6 Evaluation results on transfer learning

In this chapter, the results from different experiments for transfer learning are presented.

4.6.1 Experiment settings of the fine-tuning on transfer learning datasets

For the training of the fine-tuning, we train the network with the linear classifier for 200 epochs at a batch of 256 using SGD with Nesterov momentum with a momentum parameter of 0.9. The learning rate was selected from a grid search of logarithmically spaced learning rates between 0.001 to 0.1 and 4 logarithmically spaced values of weight decay between $10^{-}6$ and $10^{-}3$.

For data augmentation, the same transformation policy as the transfer learning experiments of SimCLR [8] was applied. At training time, only random crops with resize to 224×224 and random flips were used. For test time, the test images were resized to 256 pixels along the shorter side and took a 224×224 center crop.

4.6.2 Results on transfer learning datasets

The MLCKD-trained ResNet-18 models were able to achieve better fine-tuning performance than the baseline ResNet-18 model among 6 out of 8 transfer classification tasks, which indicated that the MLCKD knowledge method indeed assists the student model in learning better features from the source domain and has better transfer performance on the target tasks. In some tasks, the student model attained comparable accuracy performance, such as the leaf classification task. The knowledge distillation method also outperforms the supervised

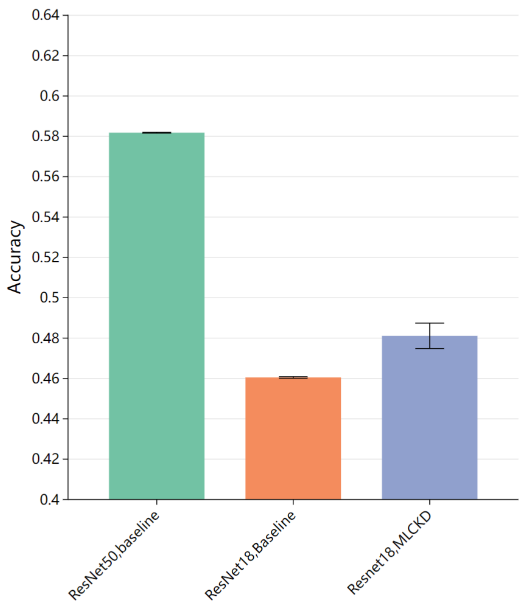


Figure 4.6: **Linear evaluation results on ILSVRC2012 dataset**

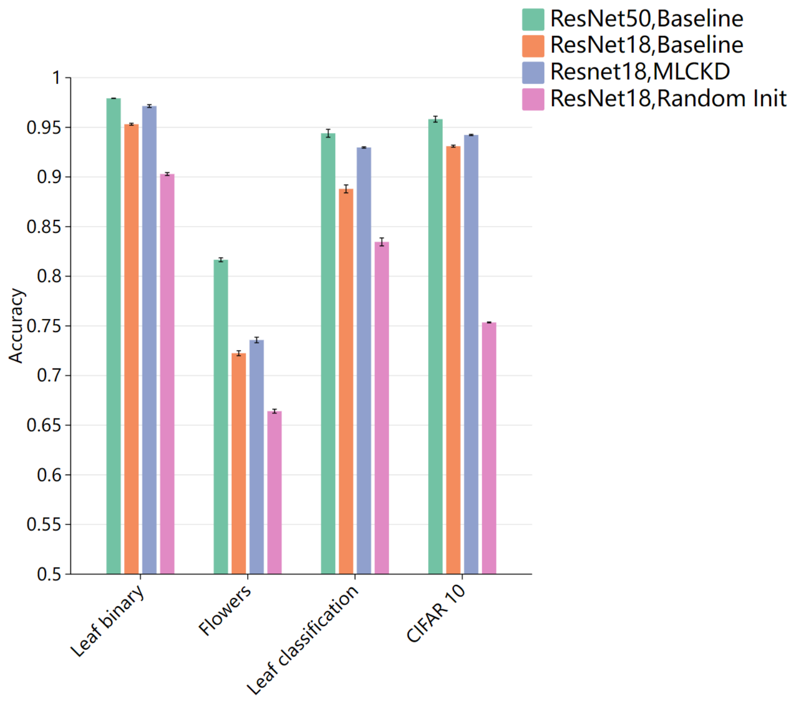


Figure 4.7: **Fine tuning results on Flowers, CIFAR 10, leaf binary task, and leaf classification task**

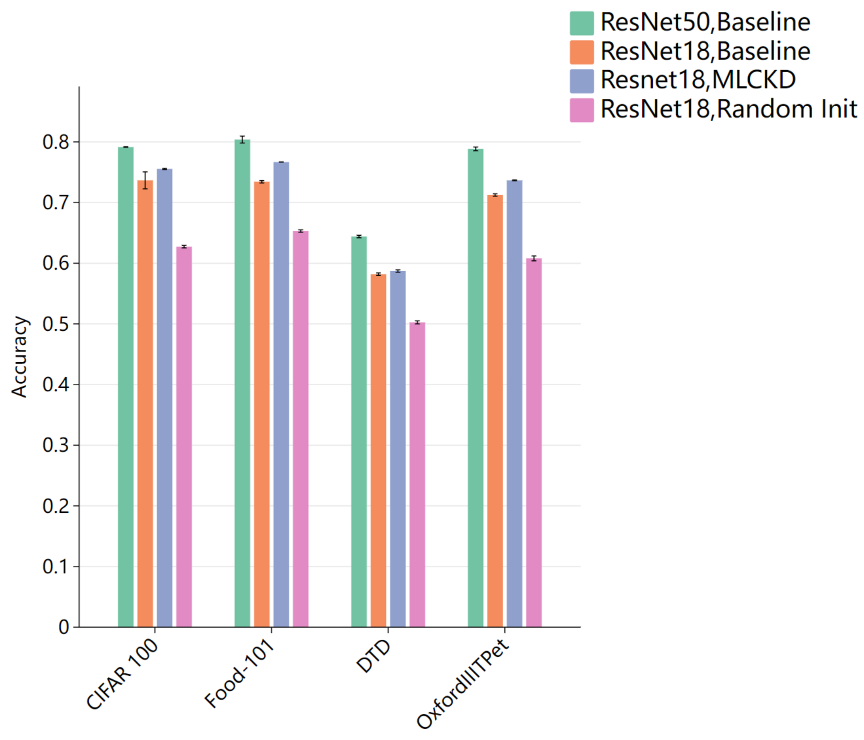


Figure 4.8: **Fine tuning results on DTD, CIFAR 100, Food-101, and Pet dataset**

training with random initialization. The features transferred from the larger teacher model facilitate the ResNet-18 model to reach a better accuracy on the transfer learning tasks within shorter training epochs and improve the training efficiency. The results are demonstrated in figure 4.7 and figure 4.8.

4.6.3 Results on transfer learning datasets in few-shot scenarios

In the case of the CIFAR 100 classification task, the MLCKD-trained ResNet-18 model was outperformed by the baseline ResNet-18 model, showing that the feature representations taught by the deeper teacher model have worse transfer performance in situations with very few labels. For the Food 101 dataset, the baseline ResNet-18 model surpasses the MLCKD method at 10% label scenario by a marginal number. All the models' accuracy improves as the training label increases, and the features learned from the deeper teacher model gain advantages on transfer performance as the number of ground truth labels increases. The results are demonstrated in figure 4.9 and 4.10.

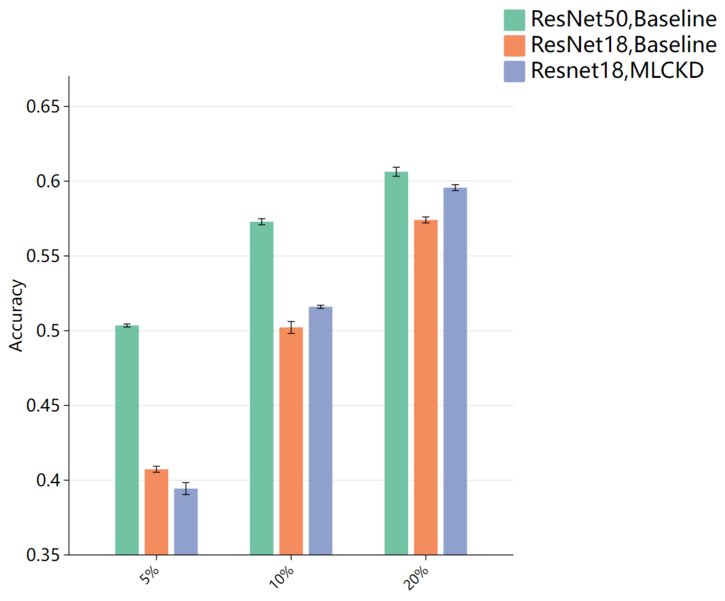


Figure 4.9: **Fine tuning results CIFAR 100 dataset with 5%,10% and 20% of the truth label**

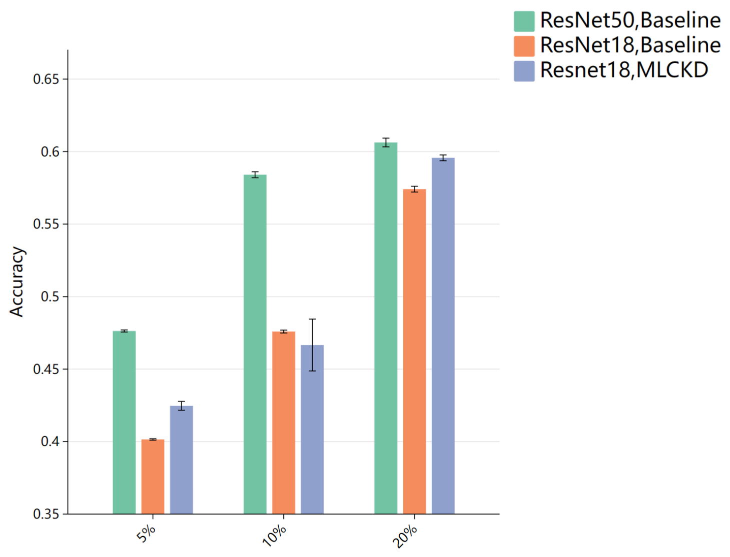


Figure 4.10: **Fine tuning results Food 101 dataset with 5%,10% and 20% of the truth label**

4.7 Ablation study

In this section, we demonstrate the results of the ablation study with different hyper-parameters.

4.7.1 Influences of temperature hyperparameters

In this experiment, different temperature settings for the two training steps in the proposed method were tested using linear evaluation protocol on the STL-10 dataset. The temperature coefficients $\tau 1$ and $\tau 2$ in the training of auxiliary branches and knowledge distillation have a major impact on the feature learning performance of the student network. Auxiliary branches typically need a smaller temperature coefficient in the contrastive loss function. Enhancing and focusing on the distributions of the samples with large contrast brings benefit to the training effect in the end. In comparison, a larger temperature is more beneficial to the knowledge distillation training of the student model, which allows the student to better mimic the feature relations of the teacher model on the input data. With proper temperature coefficient, the ResNet18 model trained by the MLCKD method surpasses the ResNet18 baseline model in linear evaluation accuracy, therefore demonstrating that the MLCKD method is capable of helping the student learn better features from the source domain data. The results are demonstrated in figure 4.11.

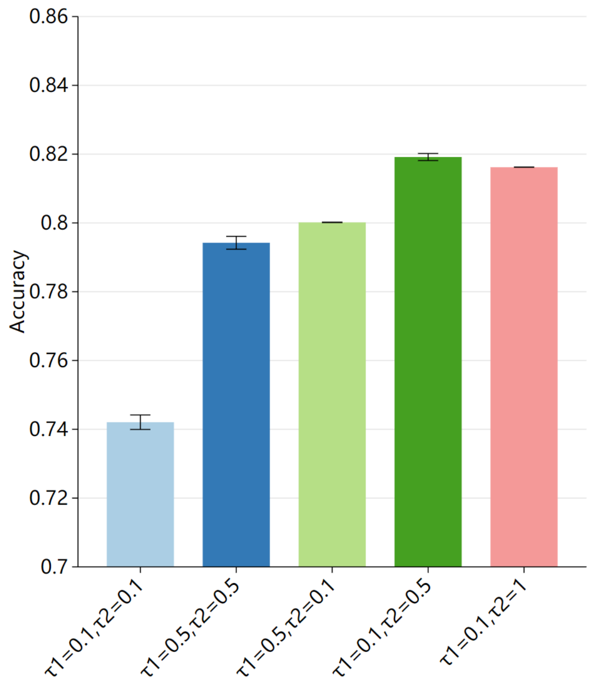


Figure 4.11: **Linear evaluation results with different training temperature hyperparameters, $\tau 1$ and $\tau 2$ represent the temperature coefficient of the training of the second and third stage**

4.8 System performance

In this chapter, the resource consumption of the evaluated models and their performance on edge devices are discussed.

4.8.1 Model Information

Listed in the table4.3 is the information of the models used in this experiment.

Model	FLOPS/G	Parameters/M
ResNet-18	1.82	11.6895
ResNet-50	4.09	25.5570

Table 4.3: **FLOPS and number of parameters of the models**

Floating point operations per second and number of parameters are major measurements of the model complexity. Listed in Table 4.3 is the information of the models used in this thesis. The FLOPS is calculated with standard (3,224,224) input. The ResNet50 model requires 2.24 times more floating point operations than the ResNet18 model when processing the same input and 54.2% more parameters.

4.8.2 Deployment devices

Listed in the table.4.4 below is the basic information for the target devices.

Device	CPU		RAM	GPU	
	Cores	Speed		Cores	Speed
Jetson Nano	4	1.43GHz	4GB	128	0.912GHz
Jetson Xavier NX 16GB	6	1.9Ghz	16GB	384	1.1GHz
Jetson AGX orin	12	2.2GHz	32GB	2048	1.3GHz

Table 4.4: **Information of the deployment devices**

4.8.3 Inference Time

This latency measures the time the network needs to carry out one forward inference with a single input of size (3,224,224). The results are measured with a batch of 50 samples, and the average latency is calculated. On a Jetson Nano device, a ResNet-50 model takes 2.67 times the inference latency of that for a ResNet-18 model. When deployed on a Jetson Xavier NX device, a ResNet-50 will require 52.4% more time to process the same (3,224,224) dimension input data. On the Jetson AGX Orin device, using a ResNet-18 model to replace a ResNet-50 model can achieve a speedup of 1.66.

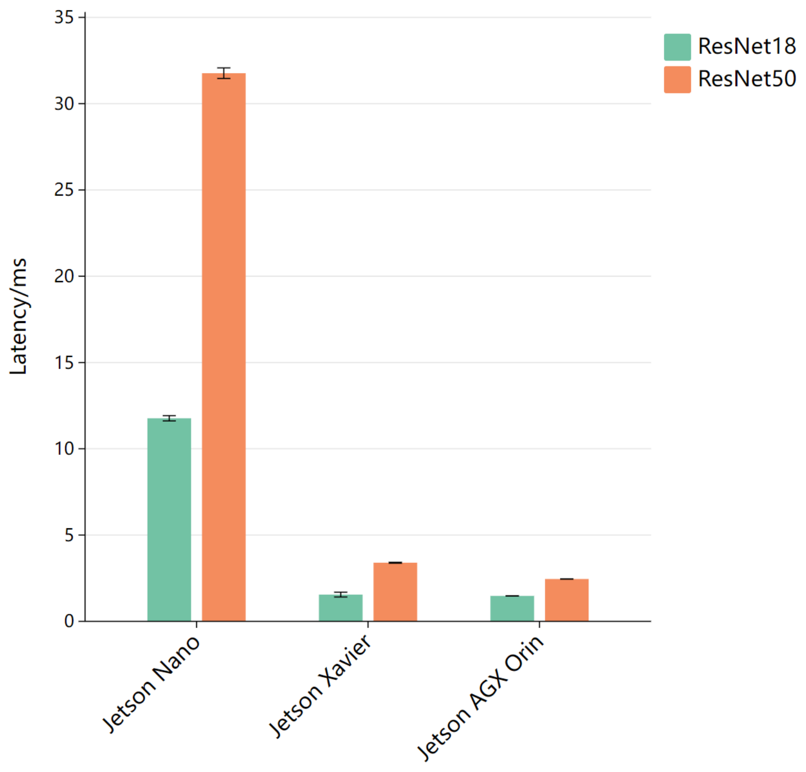


Figure 4.12: **Forward inference time of the models on different devices**

Chapter 5

Conclusion and discussion

5.1 Conclusion

In this work, we explored the effectiveness of utilizing contrastive learning techniques to construct relational knowledge in the teacher network and verify its validity on a small-scale dataset, STL-10. We proposed a training method that, with the help of auxiliary subnetworks, extracts the contrastive feature knowledge from intermediate layers and the final layer of the teacher network and transfers it to guide the training of the study network.
The knowledge distillation combining contrastive learning is able to improve the student network learning outcome. On the source domain, the linear evaluation result has demonstrated that a deeper network is capable of learning better feature knowledge from the unlabeled training data, and training the knowledge distilled from the teacher via the proposed method indeed improves the learning result. The learned features are tested among eight smaller or fine-grained transfer learning classification datasets, and the model trained by the proposed knowledge distillation method has better fine-tuning accuracy on all eight tasks and consequently demonstrates the effectiveness of the feature learned from the proposed knowledge distillation method. In comparison to the random-initialized model, the knowledge distillation pre-trained model also takes shorter training epochs to achieve convergence. In the scenario, only a small fraction of the label is available, and the MLCKD method enables the model to perform better than the baseline model trained by SimCLR on two of the transfer learning tasks.

5.2 Future Work

- **Adaptation on heterogeneous network**. The current method takes advantage of the ResNet architecture of the teacher and student network models to match the intermediate feature embedding. Further research on how to utilize the intermediate feature embedding of the heterogeneous network can be conducted to extend the generalizability of this knowledge distillation method.

- **Expoloration on other contrasive learning methods**. The current method is based on SimCLR, and this method benefits from larger training batches to construct larger sample space for better training performance, which imposes great demand on calculation resources. Research can be done to explore if other contrastive learning methods can also be applied here.

Bibliography

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

[2] Sajid Anwar and Wonyong Sung. Compact deep convolutional neural networks with coarse pruning. *arXiv preprint arXiv:1610.09639*, 2016.

[3] Sanjeev Arora, Hrishikesh Khandeparkar, Mikhail Khodak, Orestis Plevrakis, and Nikunj Saunshi. A theoretical analysis of contrastive unsupervised representation learning. *arXiv preprint arXiv:1902.09229*, 2019.

[4] Mengnan Bi, Yingjie Wang, Zhipeng Cai, and Xiangrong Tong. A privacy-preserving mechanism based on local differential privacy in edge computing. *China Communications*, 17(9):50–65, 2020.

[5] Davis Blalock, Jose Javier Gonzalez Ortiz, Jonathan Frankle, and John Guttag. What is the state of neural network pruning? *Proceedings of machine learning and systems*, 2:129–146, 2020.

[6] Lukas Bossard, Matthieu Guillaumin, and Luc Van Gool. Food-101–mining discriminative components with random forests. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part VI 13*, pages 446–461. Springer, 2014.

[7] Jiasi Chen and Xukan Ran. Deep learning with edge computing: A review. *Proceedings of the IEEE*, 107(8):1655–1674, 2019.

[8] Ting Chen, Simon Kornblith, Mohammad Norouzi, and Geoffrey Hinton. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, pages 1597–1607. PMLR, 2020.

[9] Ting Chen, Simon Kornblith, Kevin Swersky, Mohammad Norouzi, and Geoffrey E Hinton. Big self-supervised models are strong semi-supervised learners. *Advances in neural information processing systems*, 33:22243–22255, 2020.

[10] Xinlei Chen, Haoqi Fan, Ross Girshick, and Kaiming He. Improved baselines with momentum contrastive learning. *arXiv preprint arXiv:2003.04297*, 2020.

[11] Yanbei Chen, Massimiliano Mancini, Xiatian Zhu, and Zeynep Akata. Semi-supervised and unsupervised deep visual learning: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 46(3):1327–1347, 2022.

[12] Jian Cheng, Pei-song Wang, Gang Li, Qing-hao Hu, and Han-qing Lu. Recent advances in efficient computation of deep convolutional neural networks. *Frontiers of Information Technology & Electronic Engineering*, 19:64–77, 2018.

[13] Yu Cheng, Duo Wang, Pan Zhou, and Tao Zhang. A survey of model compression and acceleration for deep neural networks. *arXiv preprint arXiv:1710.09282*, 2017.

[14] Tejalal Choudhary, Vipul Mishra, Anurag Goswami, and Jagannathan Sarangapani. A comprehensive survey on model compression and acceleration. *Artificial Intelligence Review*, 53:5113–5155, 2020.

[15] Siddharth Singh Chouhan, Uday Pratap Singh, Ajay Kaul, and Sanjeev Jain. A data repository of leaf images: Practice towards plant conservation with plant pathology. In *2019 4th International Conference on Information Systems and Computer Networks (ISCON)*, pages 700–707. IEEE, 2019.

[16] Mircea Cimpoi, Subhransu Maji, Iasonas Kokkinos, Sammy Mohamed, and Andrea Vedaldi. Describing textures in the wild. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3606–3613, 2014.

[17] Adam Coates, Andrew Ng, and Honglak Lee. An analysis of single-layer networks in unsupervised feature learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, pages 215–223. JMLR Workshop and Conference Proceedings, 2011.

[18] Matthieu Courbariaux, Yoshua Bengio, and Jean-Pierre David. Binaryconnect: Training deep neural networks with binary weights during propagations. *Advances in neural information processing systems*, 28, 2015.

[19] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255, 2009.

[20] Misha Denil, Babak Shakibi, Laurent Dinh, Marc'Aurelio Ranzato, and Nando De Freitas. Predicting parameters in deep learning. *Advances in neural information processing systems*, 26, 2013.

[21] Alexey Dosovitskiy, Jost Tobias Springenberg, Martin Riedmiller, and Thomas Brox. Discriminative unsupervised feature learning with convolutional neural networks. *Advances in neural information processing systems*, 27, 2014.

[22] Zhiyuan Fang, Jianfeng Wang, Lijuan Wang, Lei Zhang, Yezhou Yang, and Zicheng Liu. Seed: Self-supervised distillation for visual representation. *arXiv preprint arXiv:2101.04731*, 2021.

[23] Nicholas Frosst, Nicolas Papernot, and Geoffrey Hinton. Analyzing and improving representations with the soft nearest neighbor loss. In *International conference on machine learning*, pages 2012–2020. PMLR, 2019.

[24] Yuting Gao, Jia-Xin Zhuang, Shaohui Lin, Hao Cheng, Xing Sun, Ke Li, and Chunhua Shen. Disco: Remedying self-supervised learning on lightweight models with distilled contrastive learning. In *European Conference on Computer Vision*, pages 237–253. Springer, 2022.

[25] Amir Gholami, Sehoon Kim, Zhen Dong, Zhewei Yao, Michael W Mahoney, and Kurt Keutzer. A survey of quantization methods for efficient neural network inference. In *Low-Power Computer Vision*, pages 291–326. Chapman and Hall/CRC, 2022.

[26] Spyros Gidaris, Praveer Singh, and Nikos Komodakis. Unsupervised representation learning by predicting image rotations. *arXiv preprint arXiv:1803.07728*, 2018.

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

[28] Jie Gui, Tuo Chen, Jing Zhang, Qiong Cao, Zhenan Sun, Hao Luo, and Dacheng Tao. A survey on self-supervised learning: Algorithms, applications, and future trends. *arXiv preprint arXiv:2301.05712*, 2023.

[29] Suyog Gupta, Ankur Agrawal, Kailash Gopalakrishnan, and Pritish Narayanan. Deep learning with limited numerical precision. In *International conference on machine learning*, pages 1737–1746. PMLR, 2015.

[30] Song Han, Jeff Pool, John Tran, and William Dally. Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, 28, 2015.

[31] Kaiming He, Haoqi Fan, Yuxin Wu, Saining Xie, and Ross Girshick. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 9729–9738, 2020.

[32] 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.

[33] Yang He, Yuhang Ding, Ping Liu, Linchao Zhu, Hanwang Zhang, and Yi Yang. Learning filter pruning criteria for deep convolutional neural networks acceleration. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2009–2018, 2020.

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

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

[36] Andrew G Howard, Menglong Zhu, Bo Chen, Dmitry Kalenichenko, Weijun Wang, Tobias Weyand, Marco Andreetto, and Hartwig Adam. Mobilenets: Efficient convolutional neural networks for mobile vision applications. *arXiv preprint arXiv:1704.04861*, 2017.

[37] Gao Huang, Zhuang Liu, Laurens Van Der Maaten, and Kilian Q Weinberger. Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4700–4708, 2017.

[38] Ashish Jaiswal, Ashwin Ramesh Babu, Mohammad Zaki Zadeh, Debapriya Banerjee, and Fillia Makedon. A survey on contrastive self-supervised learning. *Technologies*, 9(1):2, 2020.

[39] Longlong Jing and Yingli Tian. Self-supervised visual feature learning with deep neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 43(11):4037–4058, 2020.

[40] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. Technical Report 0, University of Toronto, Toronto, Ontario, 2009.

[41] Phuc H Le-Khac, Graham Healy, and Alan F Smeaton. Contrastive representation learning: A framework and review. *Ieee Access*, 8:193907–193934, 2020.

[42] Fengfu Li, Bin Liu, Xiaoxing Wang, Bo Zhang, and Junchi Yan. Ternary weight networks. *arXiv preprint arXiv:1605.04711*, 2016.

[43] Tailin Liang, John Glossner, Lei Wang, Shaobo Shi, and Xiaotong Zhang. Pruning and quantization for deep neural network acceleration: A survey. *Neurocomputing*, 461:370–403, 2021.

[44] Subhransu Maji, Esa Rahtu, Juho Kannala, Matthew Blaschko, and Andrea Vedaldi. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*, 2013.

[45] Gaurav Menghani. Efficient deep learning: A survey on making deep learning models smaller, faster, and better. *ACM Computing Surveys*, 55(12):1–37, 2023.

[46] Ishan Misra and Laurens van der Maaten. Self-supervised learning of pretext-invariant representations. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6707–6717, 2020.

[47] Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pages 722–729, 2008.

[48] Mehdi Noroozi and Paolo Favaro. Unsupervised learning of visual representations by solving jigsaw puzzles. In *European conference on computer vision*, pages 69–84. Springer, 2016.

[49] Hyun Oh Song, Yu Xiang, Stefanie Jegelka, and Silvio Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4004–4012, 2016.

[50] Chris Olah, Alexander Mordvintsev, and Ludwig Schubert. Feature visualization. *Distill*, 2017. https://distill.pub/2017/feature-visualization.

[51] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748*, 2018.

[52] Wonpyo Park, Dongju Kim, Yan Lu, and Minsu Cho. Relational knowledge distillation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 3967–3976, 2019.

[53] Omkar M Parkhi, Andrea Vedaldi, Andrew Zisserman, and C. V. Jawahar. Cats and dogs. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3498–3505, 2012.

[54] Deepak Pathak, Philipp Krahenbuhl, Jeff Donahue, Trevor Darrell, and Alexei A Efros. Context encoders: Feature learning by inpainting. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2536–2544, 2016.

[55] Mohammad Rastegari, Vicente Ordonez, Joseph Redmon, and Ali Farhadi. Xnor-net: Imagenet classification using binary convolutional neural networks. In *European conference on computer vision*, pages 525–542. Springer, 2016.

[56] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115:211–252, 2015.

[57] Florian Schroff, Dmitry Kalenichenko, and James Philbin. Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 815–823, 2015.

[58] Chen Sun, Abhinav Shrivastava, Saurabh Singh, and Abhinav Gupta. Revisiting unreasonable effectiveness of data in deep learning era. In *Proceedings of the IEEE international conference on computer vision*, pages 843–852, 2017.

[59] Christian Szegedy, Alexander Toshev, and Dumitru Erhan. Deep neural networks for object detection. *Advances in neural information processing systems*, 26, 2013.

[60] Vincent Vanhoucke, Andrew Senior, Mark Z Mao, et al. Improving the speed of neural networks on cpus. In *Proc. deep learning and unsupervised feature learning NIPS workshop*, volume 1, page 4, 2011.

[61] Huan Wang, Can Qin, Yulun Zhang, and Yun Fu. Neural pruning via growing regularization. *arXiv preprint arXiv:2012.09243*, 2020.

[62] Peisong Wang and Jian Cheng. Fixed-point factorized networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4012–4020, 2017.

[63] Xiao Wang and Guo-Jun Qi. Contrastive learning with stronger augmentations. *IEEE transactions on pattern analysis and machine intelligence*, 45(5):5549–5560, 2022.

[64] Zhirong Wu, Yuanjun Xiong, Stella X Yu, and Dahua Lin. Unsupervised feature learning via non-parametric instance discrimination. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3733–3742, 2018.

[65] Guodong Xu, Ziwei Liu, Xiaoxiao Li, and Chen Change Loy. Knowledge distillation meets self-supervision. In *European conference on computer vision*, pages 588–604. Springer, 2020.

[66] Mang Ye, Xu Zhang, Pong C Yuen, and Shih-Fu Chang. Unsupervised embedding learning via invariant and spreading instance feature. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 6210–6219, 2019.

[67] Yang You, Igor Gitman, and Boris Ginsburg. Large batch training of convolutional networks. *arXiv preprint arXiv:1708.03888*, 2017.

[68] Chunyu Yuan and Sos S Agaian. A comprehensive review of binary neural network. *Artificial Intelligence Review*, 56(11):12949–13013, 2023.

[69] Sergey Zagoruyko and Nikos Komodakis. Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer. *arXiv preprint arXiv:1612.03928*, 2016.

[70] Junbo Zhang and Kaisheng Ma. Rethinking the augmentation module in contrastive learning: Learning hierarchical augmentation invariance with expanded views. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16650–16659, 2022.

[71] Kexin Zhang, Qingsong Wen, Chaoli Zhang, Rongyao Cai, Ming Jin, Yong Liu, James Y Zhang, Yuxuan Liang, Guansong Pang, Dongjin Song, et al. Self-supervised learning for time series analysis: Taxonomy, progress, and prospects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2024.

[72] Linfeng Zhang, Xin Chen, Junbo Zhang, Runpei Dong, and Kaisheng Ma. Contrastive deep supervision. In *European Conference on Computer Vision*, pages 1–19. Springer, 2022.

[73] Nanxuan Zhao, Zhirong Wu, Rynson WH Lau, and Stephen Lin. What makes instance discrimination good for transfer learning? *arXiv preprint arXiv:2006.06606*, 2020.

[74] Mingkai Zheng, Shan You, Fei Wang, Chen Qian, Changshui Zhang, Xiaogang Wang, and Chang Xu. Ressl: Relational self-supervised learning with weak augmentation. *Advances in Neural Information Processing Systems*, 34:2543–2555, 2021.

[75] Aojun Zhou, Anbang Yao, Yiwen Guo, Lin Xu, and Yurong Chen. Incremental network quantization: Towards lossless cnns with low-precision weights. *arXiv preprint arXiv:1702.03044*, 2017.

[76] Bolei Zhou, Aditya Khosla, Agata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. *arXiv preprint arXiv:1412.6856*, 2014.