Ad Hoc Networks 194 (2027) 104406

Contents lists available at ScienceDirect

Ad Hoc Networks

journal homepage: www.elsevier.com/locate/adhoc

Multi-Level Contrastive Knowledge Distillation for Resource-efficient Visual Inference on Edge Devices

Ting Liang, Lingyu Du, Guohao Lan *

Department of Software Technology, Delft University of Technology, Van Mourik Broekmanweg 6, Delft, 2628 XE, The Netherlands

ARTICLE INFO

Keywords:
Edge AI
Knowledge distillation
Unsupervised learning
Contrastive learning

ABSTRACT

Deploying deep neural networks on networked edge devices enables low-latency inference without cloud dependency, yet is constrained by tight computational, memory, and energy budgets. Knowledge distillation is a widely used approach to compress large models into compact ones suitable for edge hardware, but conventional methods require labeled data that is often costly or unavailable in practical sensor network applications. This paper proposes Multi-Level Contrastive Knowledge Distillation (MLCKD), an unsupervised framework that transfers multi-level relational knowledge from a contrastively pretrained deep teacher model to a lightweight student using only unlabeled data. Unlike existing self-supervised distillation methods that operate solely on the teacher's final embedding, MLCKD attaches lightweight auxiliary branches at multiple network stages to distill pairwise similarity structures at both intermediate and final representation levels. We introduce a two-stage offline training procedure which first equips the frozen teacher with contrastively trained auxiliary branches, and then optimizes the student to reproduce the teacher's relational structure at every level via Kullback–Leibler divergence. Evaluation with a ResNet-50-based teacher network and ResNet-18-based student network demonstrates that MLCKD substantially improves top-1 linear evaluation accuracy over the SimCLR baseline and nearly closes the performance gap with the full-capacity teacher. Transfer learning evaluation on eight downstream classification tasks confirms that the distilled representations generalize broadly, with MLCKD outperforming the self-supervised baseline on all tasks and surpassing supervised training by a considerable margin. Inference benchmarks on three edge platforms further show that the student achieves 1.6× to 2.8× speedup over the teacher while delivering stronger representations, confirming that MLCKD enables practical, label-free deployment of resource-efficient visual intelligence on edge hardware.

1. Introduction

Edge visual inference is rapidly becoming a foundational capability in wireless sensor network systems. Examples such as autonomous drones conducting infrastructure inspection [1], environmental sensor nodes monitoring biodiversity [2], and robotic agents navigating unstructured environments [3] all rely on deep neural networks to transform raw sensor streams into actionable decisions [4,5]. Deploying and executing these models locally on edge hardware, rather than offloading computation to remote cloud servers, delivers compelling operational advantages. It can reduce communication latency, lower bandwidth consumption across capacity-limited wireless links, stronger data privacy guarantees, and continued functionality under the intermittent connectivity that is inherent to ad hoc deployments [4,6,7].

However, the computational, memory, and energy constraints of embedded processors and low-power accelerators sharply limit the complexity of models that can run in real time [5,8]. Deep residual networks [9] and other high-capacity architectures routinely exceed the resource envelopes of edge platforms, creating a persistent gap between model capability and deployment feasibility. Model compression techniques, including quantization [10], pruning [11,12], and compact architecture design [13], can reduce this gap, but often degrade the quality of learned representations in the process.

Knowledge distillation (KD) [14,15] provides a complementary and effective pathway. Rather than modifying the network architecture or requiring specialized sparse-computation hardware, KD trains a compact *student* network to reproduce the representational knowledge of a larger, complex, but more capable *teacher*, yielding a model that is directly deployable on edge devices. A rich body of work has explored different forms of teacher knowledge for distillation, ranging from softened output logits [14], intermediate feature activations [16], and pairwise relational structures among samples [17], to contrastive

* Corresponding author.
E-mail address: g.lan@tudelft.nl (G. Lan).

https://doi.org/10.1016/j.adhoc.2026.104406
Received 16 April 2026; Received in revised form 3 July 2026; Accepted 24 August 2026
Available online 2 September 2026

objectives applied to the teacher's representation space [18]. Conventional KD methods, however, rely on labeled data to define the distillation objective [14,15], an assumption that is frequently violated in edge and sensor network settings. Annotations for medical imaging require scarce clinical expertise [19]; labels for remote environmental monitoring are prohibitively expensive to collect at scale [2]; and privacy-sensitive domains may preclude centralized data curation altogether [20]. These practical constraints motivate *unsupervised* knowledge distillation [15], where the teacher's knowledge is transferred using only the unlabeled data that edge devices can readily acquire during normal operation.

Self-supervised learning (SSL) provides a natural foundation for such label-free distillation [21–23]. SSL methods derive supervision directly from unlabeled data by designing pretext tasks that exploit intrinsic data properties such as spatial structure, temporal continuity, and instance-level consistency [24,25]. Among the various SSL paradigms, contrastive learning has emerged as particularly effective for visual representation learning [26,27]. Methods such as SimCLR [26] and MoCo [28] learn transferable representations by maximizing agreement between augmented views of the same instance while separating views of different instances. Subsequent non-contrastive approaches, including BYOL [29], Barlow Twins [30], and SwAV [31], have demonstrated that explicit negative pairs are not strictly necessary, and collectively these methods have established that self-supervised pretraining can match or surpass supervised baselines [26,28,29]. More recently, self-distillation methods such as DINO [32] and DINOv2 [33] have scaled self-supervised learning to over one billion parameters, producing general-purpose visual features that rival supervised models across diverse tasks.

A consistent finding across all of these studies, however, is that representation quality scales with model capacity [26,34]: large encoders learn substantially richer features than small ones trained under identical conditions. This creates a fundamental tension for edge deployment: *the models that benefit most from self-supervised learning are precisely those that are too large to run on resource-constrained hardware, while the compact models suitable for deployment learn weaker representations when trained independently*.

Recent work has begun to resolve this tension by integrating knowledge distillation with self-supervised learning [15]. In the offline setting, SSKD [35] leverages teacher features over augmented views to enrich the student's training signal, SimCLRv2 [34] uses a self-supervised teacher to generate pseudo-labels for unlabeled data, and SEED [36] distills pairwise similarity distributions within a MoCo-style contrastive framework. In the online setting, DisCo [37] jointly trains teacher and student with contrastive embedding consistency, and MOKD [38] combines self-distillation and cross-distillation via a cross-attention feature search strategy. While these methods are effective, they share a fundamental limitation: distillation operates exclusively on the teacher's *final* output, discarding the rich intermediate representations that contrastive pretraining shapes across the full depth of the network [16,18]. These intermediate features, which span low-level spatial patterns, mid-level structural cues, and progressively abstract semantic representations, carry complementary information that is not fully captured by the final embedding alone. By ignoring this multi-level structure, existing methods transfer only a partial view of the teacher's learned knowledge [39].

This paper proposes **MLCKD** (**M**ulti-**L**evel **C**ontrastive **K**nowledge **D**istillation), an unsupervised knowledge distillation framework that addresses this limitation by transferring relational knowledge from *both* intermediate and final representations of a contrastively pretrained teacher to a lightweight student. Lightweight auxiliary branches, implemented with parameter-efficient depthwise separable convolutions [13], are attached to selected stages of both networks and project intermediate feature maps into compact embeddings. The distillation objective matches pairwise similarity structures between teacher and student at every level via Kullback–Leibler (KL) divergence [14], providing dense, multi-scale supervision throughout the student network. Training proceeds in two offline stages: the auxiliary branches on the frozen teacher are first trained with the NT-Xent contrastive loss [26] to produce well-structured intermediate embeddings, after which the student backbone, its auxiliary branches, and its projection head are jointly optimized to reproduce the teacher's similarity distributions at all levels. Unlike methods that distill only final embeddings [36,40] or require computationally expensive online co-training [37,38], MLCKD captures the teacher's full representational hierarchy in an offline setting that integrates directly into standard self-supervised pretraining pipelines.

We evaluate MLCKD using ResNet-50 [9] as the teacher and ResNet-18 [9] as the student on two pretraining benchmarks (STL-10 [41] and ILSVRC2012 [42]) and eight downstream classification tasks. Under linear evaluation [26], the distilled student improves over the SimCLR ResNet-18 baseline by 6 and 3 percentage points on STL-10 and ILSVRC2012, nearly closing the gap with the full-capacity teacher on STL-10 (0.84 vs. 0.85). We also compare against SEED [36], a self-supervised distillation method that distills the teacher's final embedding only; MLCKD outperforms SEED by 3 and 2 percentage points on the benchmarks, confirming that the intermediate auxiliary branches supply structural knowledge that final-embedding distillation alone cannot convey. Transfer learning across eight downstream tasks shows that MLCKD outperforms both the standalone self-supervised baseline and supervised training from random initialization on all eight tasks, with gains of up to 5 percentage points over the self-supervised baseline and up to 20 percentage points over supervised training. Few-shot fine-tuning on CIFAR-100 and Food-101 further confirms that the advantage persists under limited labeled data, closing within 1 percentage point of the ResNet-50 teacher at the 20% label fraction. Inference benchmarks on three NVIDIA Jetson edge platforms show that the ResNet-18 student runs 1.6×–2.8× faster than the teacher.

The contributions of this paper are as follows:

1. We propose MLCKD, an unsupervised knowledge distillation framework that distills multi-level pairwise similarity structures from a pretrained teacher to a compact student through lightweight auxiliary branches and a two-stage offline training procedure, enabling the student to inherit the teacher's full representational hierarchy using only unlabeled data.
2. We conduct comprehensive experiments across two pretraining benchmarks and eight downstream classification tasks, demonstrating consistent and substantial improvements in both representation quality and transfer performance over standalone self-supervised baselines [26] as well as supervised training from random initialization.
3. We profile on three edge platforms spanning a wide range of compute capability, confirming that MLCKD delivers both improved accuracy and 1.6× to 2.8× inference speedup, directly addressing the resource efficiency requirements of intelligent sensor networks and edge AI applications.

Paper roadmap. The rest of the paper is organized as follows. Section 2 reviews related work on self-supervised learning, contrastive learning, model compression, and knowledge distillation. Section 3 presents the MLCKD framework in detail. Section 4 describes the experimental setup and results. Section 5 discusses limitations and outlines future directions.

2. Background and related work

This section reviews the technical foundations that underpin the proposed method. Section 2.1 introduces self-supervised learning, which enables representation learning without manual annotations. Section 2.2 focuses on contrastive learning, the SSL paradigm on which MLCKD is built. Section 2.3 discusses model compression techniques that motivate the need for compact architectures on edge devices, and Section 2.4 examines knowledge distillation, which bridges high-capacity representation learning and lightweight deployment. We conclude by identifying the gap that the proposed method addresses.

2.1. Self-supervised learning

Self-supervised learning (SSL) derives supervision signals directly from unlabeled data, eliminating the need for human annotations [21]. By designing pretext tasks that exploit intrinsic data properties, such as spatial structure, temporal continuity, and instance-level consistency, SSL generates pseudo-labels automatically and trains an encoder whose representations transfer to downstream tasks including classification, detection, and segmentation [22,23].

The general SSL pipeline comprises three stages: (i) defining a pretext task that produces pseudo-supervision, (ii) optimizing an encoder to learn general-purpose representations, and (iii) transferring the learned representations to target tasks via linear evaluation or fine-tuning [24]. The effectiveness of SSL depends critically on the pretext task: it must be sufficiently challenging to force the encoder to capture semantic content and spatial regularities that generalize across tasks [25]. Early pretext tasks for visual SSL include the following:

- **Image inpainting** trains the encoder to reconstruct masked or missing regions from surrounding context, requiring both local texture and global structural understanding [43].
- **Rotation prediction** classifies the discrete rotation angle applied to an input image (e.g., 0°, 90°, 180°, or 270°), encouraging the network to encode global semantic cues related to canonical orientations [44].
- **Jigsaw puzzle solving** permutes image patches and trains the model to recover the original, demanding reasoning about part–whole relationships and spatial dependencies [45].
- **Instance discrimination** treats every image as its own class and distinguishes instances under varied augmentations [46,47], producing transformation-invariant yet instance-separable representations that transfer to recognition tasks.

More recently, masked image modeling (MIM) has emerged as a powerful pretext paradigm that extends image inpainting to patch-level prediction [48–50]. MAE [48] reconstructs raw pixel values from 75% masked patches using an asymmetric encoder–decoder. BEiT [49] predicts discrete visual tokens via a pre-trained tokenizer, bridging BERT-style pretraining to vision. SimMIM [50] further simplifies the pipeline with a lightweight linear head and random masking. Compared with contrastive methods, MIM approaches excel at learning fine-grained spatial features and scale favorably with Transformer architectures. Among the paradigms discussed above, instance discrimination is most directly related to the proposed method, as it provides the conceptual foundation for the contrastive learning framework described next.

2.2. Contrastive learning

Contrastive learning learns an embedding space in which semantically similar samples are pulled together and dissimilar ones are pushed apart [26,27]. Two augmented views of the same image form a *positive pair*, while views from different images constitute *negative pairs*. Letting $f(\cdot)$ denote an encoder with a projection head and $\mathrm{sim}(\cdot,\cdot)$ a similarity metric (e.g., cosine similarity), the contrastive loss maximizes $\mathrm{sim}\big(f(x),\, f(x^+)\big)$ for positives while minimizing $\mathrm{sim}\big(f(x),\, f(x^-)\big)$ for negatives, encouraging representations that are transformation-invariant yet instance-discriminative [26,28].

A central design dimension is the mechanism for managing negative samples. SimCLR [26] relies on large batch sizes and a learnable projection head, whereas MoCo [28] decouples the negative set size from the batch size via a momentum-updated queue. Subsequent work has demonstrated that explicit negatives are not strictly necessary. For example, BYOL [29] eliminates negatives through an asymmetric online–target architecture, Barlow Twins [30] drives the cross-correlation matrix of twin embeddings toward identity, and SwAV [31] enforces consistency between online cluster assignments from different views while introducing the multi-crop augmentation strategy. Collectively, these methods have established that self-supervised pretraining can match or surpass supervised baselines [26,28,29]. Building on these foundations, DINO [32] applies self-distillation to Vision Transformers, revealing emergent scene-layout properties in self-attention maps, and DINOv2 [33] scales this approach to 1.1B parameters, producing general-purpose features that rival supervised models across classification, segmentation, and retrieval.

Data augmentation plays a defining role in contrastive learning because it specifies the invariances encoded in the representation. SimCLR shows that combining strong augmentations (e.g., random cropping with color distortion) improves representation quality by preventing shortcut solutions [26]. However, overly aggressive transformations can compromise semantic consistency within positive pairs [51,52]. Tian et al. [53] formalize this trade-off through the *InfoMin* principle: optimal views should share minimal mutual information while retaining task-relevant content. These insights underscore that augmentation strategies must balance transformation robustness against semantic preservation for a given downstream task.

A recurring observation across these studies is that representation quality scales with model capacity [26,34]: larger encoders consistently learn richer and more transferable features than smaller ones trained under identical protocols. While this is desirable for maximizing accuracy, it presents a direct conflict with the resource constraints of edge deployment, motivating the model compression techniques discussed next.

2.3. Model compression

Deep neural networks achieve high accuracy at the cost of large parameter counts and substantial computational demands [9], presenting barriers to deployment on edge devices with limited compute, memory, and energy budgets. Model compression addresses these challenges by reducing model size and inference cost while seeking to preserve accuracy. This subsection focuses on quantization and pruning, the two strategies most relevant to edge deployment, although neural architecture search [54] and low-rank factorization [55] are also active research directions [8].

2.3.1. Quantization

Quantization reduces model size and accelerates inference by approximating weights and activations with lower-precision values [10]. Because deep networks are typically over-parameterized and robust to small perturbations, many models can be converted to FP16, INT8, or INT4 with marginal accuracy loss [10,56]. At the extreme, binary neural networks represent weights with single-bit values [57].

Quantization methods differ in *what* is quantized and *when*. Weight quantization lowers parameter precision to reduce storage and bandwidth, while activation quantization lowers intermediate tensor precision to enable efficient integer arithmetic [58]. Regarding timing, quantization-aware training (QAT) simulates quantization during training, allowing the model to compensate for discretization errors [10], while post-training quantization (PTQ) applies quantization after training using a small calibration set [10]. Recent PTQ methods such as GPTQ [59] and AWQ [60] have extended low-bit quantization to models with hundreds of billions of parameters with negligible accuracy degradation.

2.3.2. Network pruning

Network pruning removes weights or structural components that contribute minimally to model output, exploiting the substantial redundancy in modern deep networks [11,61]. Unstructured pruning zeroes out individual weights and can achieve high sparsity, but typically requires sparse-computation support for real speedups [12]. Structured pruning removes entire filters, channels, or layers, directly reducing FLOPs and memory footprint on standard hardware [62,63]. Recent work such as DepGraph [64] enables fully automatic structural pruning

across heterogeneous architectures. The importance criterion for selecting elements to prune remains a key design choice: magnitude-based scores are widely used for their simplicity, while regularization-based methods encourage removable structures during training [12].

While quantization and pruning effectively reduce model size, they operate on an already-trained network and can degrade representation quality, particularly when applied aggressively to lower-capacity architectures. Knowledge distillation takes a fundamentally different approach: rather than compressing an existing model, it trains a compact network from scratch under the guidance of a more capable teacher, directly optimizing the student's learned representations.

2.4. Knowledge distillation

Knowledge distillation (KD) transfers the learned representations of a high-capacity *teacher* to a smaller *student*, enabling the student to approximate the teacher's performance at lower cost [14,15]. KD methods are organized by training protocol: *offline* distillation uses a fixed pre-trained teacher; *online* methods train teacher and student jointly [65]; and *self-distillation* uses a single network as both teacher and student [15].

2.4.1. Distillation objectives

The seminal logit-based formulation by Hinton et al. [14] trains the student to match the teacher's softened output distribution via temperature-scaled softmax. These soft targets expose inter-class similarity structure absent from hard labels, improving student generalization. Feature-based methods such as FitNets [39] and attention transfer [16] align intermediate activations between teacher and student, demonstrating that knowledge encoded in hidden layers provides complementary supervision beyond what output logits alone can convey. Relation-based methods such as RKD [17] take a different perspective, enforcing consistency in pairwise distance and angular relations among samples rather than matching individual feature vectors. CRD [18] unifies feature-based and relation-based perspectives through a contrastive objective applied to the teacher's representation space, showing that structural knowledge transfer outperforms standard KL-divergence distillation on compression, ensemble distillation, and cross-modal transfer benchmarks. This contrastive formulation of distillation is particularly relevant to the present work, as MLCKD extends it to multiple intermediate representation levels in a self-supervised setting.

2.4.2. Knowledge distillation in self-supervised learning

A growing body of work integrates KD with contrastive learning, producing distillation methods that are effective when labeled data is scarce or entirely unavailable. In the *offline* setting, SSKD [35] leverages teacher features over augmented views to enrich the student's training signal, SimCLRv2 [34] uses a self-supervised teacher to generate pseudo-labels for unlabeled data, and SEED [36] distills pairwise similarity distributions within a MoCo-style contrastive framework. In the *online* setting, DisCo [37] jointly trains teacher and student with contrastive embedding consistency, and MOKD [38] combines self-distillation and cross-distillation with a cross-attention feature search strategy. Collectively, these studies demonstrate that integrating distillation with contrastive learning yields effective representation transfer to compact architectures, especially when strong pre-trained teachers are available.

2.4.3. Limitations of existing approaches

Despite their effectiveness, the self-supervised distillation methods reviewed above share a common limitation: they operate exclusively on the teacher's *final* embedding, discarding the intermediate representations that contrastive pretraining shapes across the full depth of the network [16,39]. Prior work in supervised KD has convincingly shown that intermediate features, spanning low-level spatial patterns to mid-level structural cues, carry complementary knowledge not captured by the final output alone [16,18,39]. Yet this insight has not been systematically exploited in the self-supervised distillation setting, where the absence of labels makes multi-level knowledge transfer both more challenging and more valuable, as the student cannot compensate for missing teacher guidance through task specific supervision.

The proposed MLCKD framework addresses this gap by distilling pairwise similarity structures at *both* intermediate and final representation levels, enabling the student to inherit the teacher's full representational hierarchy. Unlike SEED and SimCLRv2, which are limited to final-embedding distillation, and unlike DisCo and MOKD, which require computationally expensive online co-training, MLCKD operates in an offline setting with lightweight auxiliary branches, making it both more comprehensive in the knowledge it transfers and more practical for resource-constrained training environments.

3. Method

This section presents MLCKD, an unsupervised knowledge distillation framework that transfers multi-level relational knowledge from a contrastively pretrained teacher to a lightweight student using only unlabeled data. As established in Section 2.4, existing self-supervised distillation methods either operate solely on the teacher's final embedding [34,36] or require computationally expensive online co-training [37,38], and none systematically exploit the intermediate representations that contrastive pretraining shapes across the full network depth [16,39]. MLCKD addresses this gap by distilling pairwise similarity structures at both intermediate and final representation levels in an offline setting, enabling the student to inherit the teacher's complete representational hierarchy.

Below, we formulate the problem in Section 3.1, Section 3.2 describes the overall framework, Section 3.3 introduces the auxiliary branches for intermediate-layer distillation, and Section 3.4 details the two-stage training procedure.

3.1. Problem formulation

Knowledge distillation requires two design decisions: *what* knowledge to extract from the teacher, and *how* to transfer it to the student. In supervised settings, the teacher's softened logits provide natural supervision targets [14], optionally complemented by intermediate-layer hints [16,39]. In the unsupervised setting considered in this paper, no ground-truth labels are available and the downstream task is unknown at training time. This makes rendering task-specific signals such as classifier logits unsuitable. Therefore, the distillation process must instead rely on task-agnostic relational knowledge that captures transferable structure in the unlabeled data.

Let D denote a large unlabeled dataset and let f^T denote a high-capacity teacher encoder pretrained on D with a self-supervised contrastive objective (e.g., SimCLR [26]). The goal is to train a lightweight student encoder f^S using only D and the representational guidance provided by f^T, such that the student satisfies three requirements:

R1. The learned representation space of the student preserves the pairwise similarity structure captured by the teacher at multiple network depths.
R2. The resulting representations transfer effectively to diverse, *a priori* unknown downstream tasks.
R3. The student operates at substantially reduced inference cost in terms of memory, computation, and latency, enabling deployment on edge devices.

3.2. Framework overview

Fig. 1 illustrates an overview of the proposed framework. A teacher–student architecture is adopted in which the teacher encoder f^T is

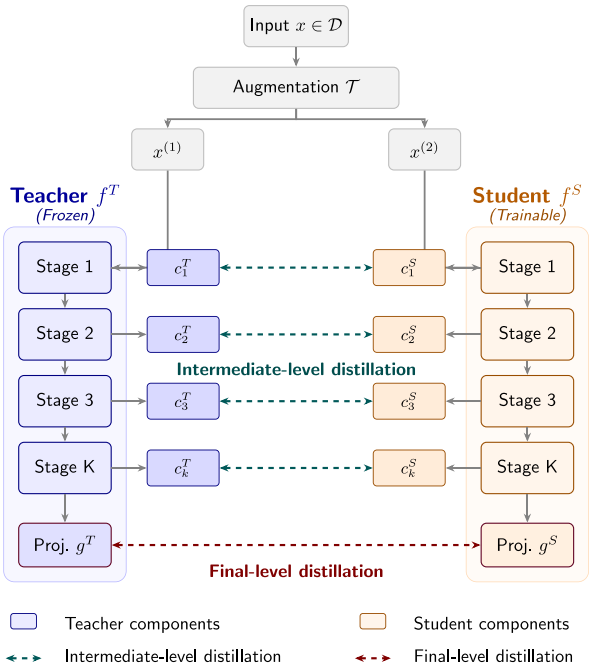

Fig. 1. Overview of the MLCKD framework. Task-agnostic relational knowledge is transferred from a frozen, contrastively pretrained teacher to a trainable, lightweight student at two complementary levels: the final embedding (via projection heads) and intermediate representations (via K auxiliary branches). The teacher remains fixed throughout distillation. Both levels are matched using KL divergence on pairwise similarity distributions.

pretrained on $\mathcal{D}$ using SimCLR [26] and subsequently kept fixed throughout distillation. The student encoder f^S receives the same augmented inputs and is optimized to align its representations with those of the teacher.

Distillation is performed at two complementary levels, addressing requirement **R1**:

Final-embedding level. Each encoder is followed by a projection head (g^T for the teacher, g^S for the student) that maps the encoder output to a latent space where pairwise similarity is measured. The teacher projection head is inherited from contrastive pretraining and remains frozen; the student projection head shares the same architecture but is randomly initialized. Matching the teacher's final similarity structure encourages the student to preserve the global geometry and semantic organization of the representation space.

Intermediate-embedding level. Lightweight auxiliary branches are attached to K selected stages of both encoders (Section 3.3). These branches project intermediate feature maps into compact embeddings, enabling cross-network comparison even when teacher and student differ in capacity or channel dimensionality. Intermediate representations capture complementary information – from low-level spatial patterns to mid-level structural abstractions – that is shaped by contrastive pretraining [16,39] but is not fully reflected in the final embedding alone. Distilling at both levels allows the student to inherit the teacher's full representational hierarchy rather than only its output behavior.

For each input sample $x \in \mathcal{D}$, a stochastic augmentation module samples two transformations $t, t' \sim \mathcal{T}$ and produces two views, $x^{(1)} = t(x)$ and $x^{(2)} = t'(x)$. These paired views form the basis of instance discrimination and provide shared input to both teacher and student during distillation. The framework components are:

• **Pretrained teacher encoder** f^T is a high-capacity backbone pretrained via SimCLR and frozen during distillation, providing supervision through its final embedding and intermediate-layer representations.

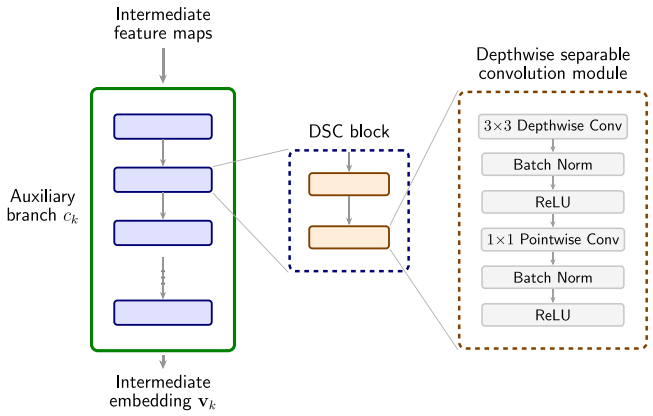

Fig. 2. Architecture of an auxiliary branch c_k. The branch consists of multiple DSC blocks (left), each containing two depthwise separable convolution modules [13] (center). Each module comprises a 3×3 depthwise convolution followed by a 1×1 pointwise convolution, with batch normalization and ReLU after each convolution (right).

• **Student encoder** f^S is a lightweight network whose parameters, along with those of its projection head g^S and auxiliary branches, are jointly optimized during distillation.

• **Projection heads** g^T, g^S are non-linear multilayer perceptron network (MLP) heads mapping final encoder outputs to a 128-dimensional contrastive latent space.

• **Auxiliary branches** $\{c_k^T\}_{k=1}^K$, $\{c_k^S\}_{k=1}^K$ are lightweight convolutional subnetworks projecting intermediate features into embeddings suitable for cross-network comparison (Section 3.3).

• **Distillation losses**: the student is trained to reproduce the teacher's pairwise similarity structure at every level via KL divergence (Section 3.4).

3.3. Auxiliary branches

A key technical challenge in multi-level distillation is comparing intermediate representations across architectures that may differ in channel dimensionality, spatial resolution, and semantic abstraction level. To address this, a set of lightweight subnetworks, referred to as *auxiliary branches*, are attached to K selected stages of both the teacher and student encoders. Each branch receives a stage's intermediate feature map and projects it into a compact embedding that can be directly compared between teacher and student. Functionally, these branches serve the same role as the projection head in standard contrastive learning [26], which maps features into a space where a relational objective operates, but they do so at intermediate network depths rather than at the final output.

As shown in Fig. 2, each auxiliary branch is implemented as a small convolutional network composed of multiple blocks, where each block contains two depthwise separable convolution modules (DSC) [13]. Depthwise separable convolutions factorize a standard convolution into a depthwise spatial filtering step followed by a pointwise (1×1) channel-mixing step, reducing parameter count and computational cost by a factor proportional to the kernel size [13]. Stacking two such modules per block increases nonlinearity and representational capacity while maintaining efficiency. This convolutional design was chosen over simpler alternatives (e.g., linear projections or standard MLPs) because the intermediate features are spatial feature maps whose local structure benefits from convolutional processing; flattening them into vectors for an MLP would discard spatial relationships that are precisely the information MLCKD aims to transfer.

An auxiliary branch is attached after each convolutional stage of the encoder, yielding K branches in total ($K = 4$ for both ResNet-50 and

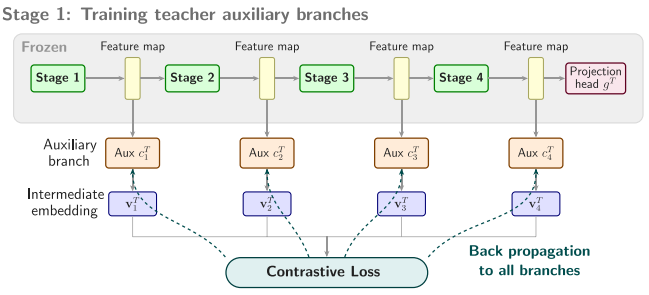

Fig. 3. Stage 1: Training teacher auxiliary branches. The teacher backbone (gray, frozen) produces intermediate feature maps at each stage. Auxiliary branches $\{c_k^T\}_{k=1}^4$ project these feature maps into compact embeddings $\{\mathbf{v}_k^T\}_{k=1}^4$, which are trained independently with the NT-Xent contrastive loss [26]. Gradients flow only through the auxiliary branches; the backbone remains fixed.

Algorithm 1 Stage 1: Training the teacher auxiliary branches

Input: Batch size N, temperature τ, frozen teacher backbone $\{f_k\}_{k=1}^K$, auxiliary branches $\{c_k\}_{k=1}^K$, augmentation distribution $\mathcal{T}$
Output: Trained auxiliary branches $\{c_k\}_{k=1}^K$

```
for each mini-batch {x_i}_{i=1}^N do
    for i = 1 to N do
        Sample t, t' ~ T; construct x̃_{2i-1} = t(x_i), x̃_{2i} = t'(x_i)
    end for
    for k = 1 to K do
        Compute {v_{p,k}}_{p=1}^{2N} via Eq. (2)
        Compute L_k via Eqs. (4)–(5)
    end for
    Update {c_k}_{k=1}^K by minimizing L_stage1 = Σ_{k=1}^K L_k
end for
    return Trained auxiliary branches {c_k}_{k=1}^K
```

ResNet-18 in our experiments). Because features at deeper stages are more abstract and semantically compact, the branch depth is varied across stages following an *inverse-depth* principle: shallower branches (fewer blocks) are used for deeper stages where the feature maps are already highly processed, and relatively deeper branches are used for earlier stages where the feature maps are larger, higher-dimensional, and less abstract. This design balances alignment quality against computational overhead. For the ResNet-50 teacher, a 1×1 convolution is additionally applied at the input of each branch to reduce the channel dimensionality of bottleneck features before projection. The exact branch configurations are specified in Section 4.

3.4. Training procedure

Our training proceeds in two sequential stages, as summarized in Fig. 3 (Stage 1) and Fig. 4 (Stage 2). Both stages operate on the same unlabeled dataset $\mathcal{D}$ with the same augmentation pipeline $\mathcal{T}$.

3.4.1. Stage 1: Training the teacher auxiliary branches

The teacher backbone f^T and its projection head g^T are pretrained using SimCLR [26]. Unlike the conventional SimCLR protocol, in which the projection head is discarded after pretraining, g^T is retained for use in Stage 2. The purpose of Stage 1 is to train auxiliary branches that produce well-structured intermediate embeddings suitable for distillation. The teacher backbone is frozen throughout this stage to preserve its learned representations; only the auxiliary branch parameters $\{c_k\}_{k=1}^K$ are optimized. Each branch is trained independently with the NT-Xent contrastive loss [26] applied to its own output embeddings, encouraging each intermediate level to develop a discriminative similarity structure.

Given a mini-batch of N samples $\{x_i\}_{i=1}^N$, two augmentations $t, t' \sim \mathcal{T}$ are sampled per sample to produce $2N$ augmented views:

$$\tilde{x}_{2i-1} = t(x_i), \qquad \tilde{x}_{2i} = t'(x_i), \qquad i = 1, \dots, N. \tag{1}$$

For each sample, its two views form a positive pair; all other $2(N-1)$ views in the mini-batch serve as negatives. Let $f_k(\cdot)$ denote the frozen teacher backbone output up to stage k, and let $c_k(\cdot)$ denote the kth auxiliary branch. The branch embedding for an augmented sample $\tilde{x}_p$ is:

$$\mathbf{v}_{p,k} = c_k\big(f_k(\tilde{x}_p)\big), \qquad p \in \{1, \dots, 2N\}, \quad k \in \{1, \dots, K\}. \tag{2}$$

Pairwise similarity is measured via cosine similarity, defined once here and used throughout the remainder of this section:

$$\mathrm{sim}(\mathbf{u}, \mathbf{w}) = \frac{\mathbf{u}^\top \mathbf{w}}{\|\mathbf{u}\|_2 \, \|\mathbf{w}\|_2}. \tag{3}$$

For the kth branch, the contrastive loss for a positive pair (p, q) follows the NT-Xent formulation [26]:

$$\ell_{k,p,q} = -\log \frac{\exp\big(\mathrm{sim}(\mathbf{v}_{p,k}, \mathbf{v}_{q,k})/\tau\big)}{\sum_{r=1}^{2N} \mathbb{1}_{r \neq p} \exp\big(\mathrm{sim}(\mathbf{v}_{p,k}, \mathbf{v}_{r,k})/\tau\big)}, \tag{4}$$

where $\tau > 0$ is a temperature parameter controlling the sharpness of the similarity distribution. The loss for branch k averages over all positive pairs in the mini-batch:

$$\mathcal{L}_k = \frac{1}{2N} \sum_{i=1}^{N} \big[\ell_{k,2i-1,2i} + \ell_{k,2i,2i-1}\big], \tag{5}$$

and the total Stage-1 objective sums over all K branches:

$$\mathcal{L}_{\text{stage1}} = \sum_{k=1}^{K} \mathcal{L}_k. \tag{6}$$

The training procedure is summarized in Algorithm 1. Upon completion, each auxiliary branch produces embeddings that encode a well-structured similarity space at its corresponding network depth, providing the multi-level supervision signal used in Stage 2.

3.4.2. Stage 2: Knowledge distillation to the student

With the teacher encoder and its auxiliary branches fully trained, Stage 2 transfers the teacher's relational knowledge to the student. A critical design choice in this stage is *what* to distill. Rather than matching individual feature vectors, which would tightly couple the student to the teacher's specific activation patterns and may not generalize well across tasks, MLCKD distills the *pairwise similarity structure* produced by the teacher at each representation level. This relational formulation, inspired by RKD [17] and CRD [18], ensures that the student learns the geometric organization of the teacher's embedding space, which has been shown to be more transferable than point-wise feature alignment.

The student is equipped with the same auxiliary-branch configuration as the teacher (K branches at corresponding stages). During training, the same augmented mini-batch is forwarded through both networks. The teacher remains fixed, while the student backbone f^S, student auxiliary branches $\{c_k^S\}_{k=1}^K$, and student projection head g^S are jointly optimized. This joint optimization allows gradients from the intermediate-level losses to flow back through the student backbone, providing dense supervision at multiple depths rather than only at the final layer.

For a given auxiliary branch or projection head, let $\mathbf{z}_i$ and $\mathbf{z}_j$ denote the output embeddings for two augmented samples from either the teacher or student network. The pairwise similarity matrix is computed via the cosine similarity defined in Eq. (3):

$$\mathcal{A}_{i,j} = \mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j). \tag{7}$$

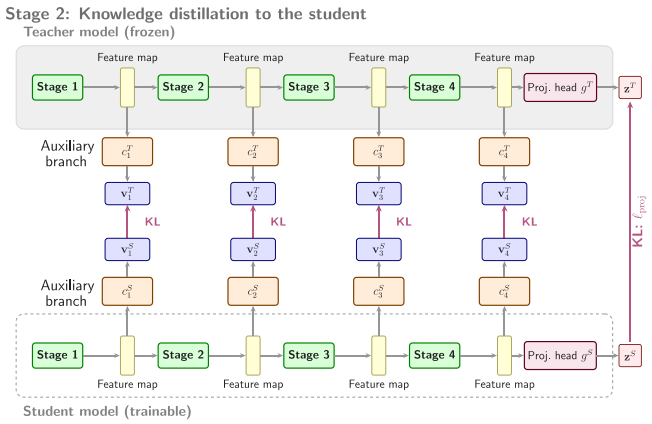

Fig. 4. Stage 2: Knowledge distillation to the student. The frozen teacher (gray, top) and its trained auxiliary branches produce intermediate embeddings $\{\mathbf{v}_k^T\}$ and a final representation $\mathbf{z}^T$. The trainable student (dashed, bottom) is equipped with the same auxiliary-branch configuration and produces corresponding embeddings $\{\mathbf{v}_k^S\}$ and $\mathbf{z}^S$. At each level, the student is trained to match the teacher's pairwise similarity structure via KL divergence. The total loss aggregates all intermediate-level losses $\{\ell_k\}$ and the final-level loss ℓ_{proj}.

A row-wise softmax with temperature τ converts the similarity matrix into a probability distribution over sample relations:

$$\mathcal{P}_{i,j} = \frac{\exp(\mathcal{A}_{i,j}/\tau)}{\sum_{r=1}^{2N} \exp(\mathcal{A}_{i,r}/\tau)}. \tag{8}$$

Each row of $\mathcal{P}$ describes the relational profile of one sample with respect to all others in the mini-batch, capturing which samples are mapped close together and which are separated in the teacher's (or student's) embedding space.

The distillation loss between the teacher's probability matrix $\mathcal{P}^{(T)}$ and the student's probability matrix $\mathcal{P}^{(S)}$ is computed using the Kullback–Leibler (KL) divergence, scaled by τ^2 to ensure that gradient magnitudes remain stable across different temperature settings [14]:

$$\ell = \tau^2 \sum_{i=1}^{2N} \sum_{j=1}^{2N} \mathcal{P}_{i,j}^{(T)} \log \frac{\mathcal{P}_{i,j}^{(T)}}{\mathcal{P}_{i,j}^{(S)}}. \tag{9}$$

The total Stage-2 loss aggregates the KL-divergence losses across all K auxiliary-branch pairs and the projection-head pair:

$$\mathcal{L}_{\text{stage2}} = \underbrace{\sum_{k=1}^{K} \ell_k}_{\text{intermediate levels}} + \underbrace{\ell_{\text{proj}}}_{\text{final level}}, \tag{10}$$

where ℓ_k and ℓ_{proj} are computed using Eqs. (7)–(9) with the embeddings from the corresponding teacher–student branch or projection-head pair. The intermediate-level and final-level losses are summed with equal weight. This design follows standard practice in feature-based knowledge distillation. FitNets [39] and attention transfer [16] both sum per-layer distillation terms uniformly, as do the layer-wise transformer distillation methods Patient-KD [66] and TinyBERT [67]. Moreover, since every term in Eq. (10) is a KL divergence between row-normalized similarity distributions over the same mini-batch, so all $K+1$ losses share the same scale and units by construction. There is no a priori reason to weight one level over another. However, the alternative, i.e., learning per-layer weights through a meta-network or similar machinery [68], is itself a non-trivial design that introduces additional training and hyperparameters, which conflicts with the unsupervised setting we target. We therefore use uniform weighting throughout. The complete Stage-2 procedure is summarized in Algorithm 2. After training, the auxiliary branches and projection heads are discarded; only the student backbone f^S is retained for downstream deployment.

Algorithm 2 Stage 2: Student distillation

Input: Teacher $(f^T, \{c_k^T\}_{k=1}^K, g^T)$, student $(f^S, \{c_k^S\}_{k=1}^K, g^S)$, batch size N, temperature τ, augmentation distribution $\mathcal{T}$
Output: Trained student encoder f^S

```
for each mini-batch {x_i}_{i=1}^N do
    Sample t, t' ~ T per sample; construct 2N augmented views
    Forward both teacher and student on the augmented mini-batch
    for k = 1 to K do
        Compute similarity matrices A_k^(T), A_k^(S) via Eq. (7)
        Compute probability matrices P_k^(T), P_k^(S) via Eq. (8)
        Compute ℓ_k via Eq. (9)
    end for
    Compute ℓ_proj analogously from projection-head outputs
    Update f^S, {c_k^S}, g^S by minimizing L_stage2 = Σ_k ℓ_k + ℓ_proj
end for
    return Trained student encoder f^S
```

Table 1
Number of residual blocks per stage for the teacher (ResNet-50, bottleneck blocks) and student (ResNet-18, basic blocks) encoders.

Network	Stage 1	Stage 2	Stage 3	Stage 4
ResNet-50 (teacher)	3	4	6	3
ResNet-18 (student)	2	2	2	2

4. Experiments

In this section, we evaluate the proposed MLCKD framework. We first describe the datasets, network architectures, and training configurations in Section 4.1. Section 4.2 defines the evaluation protocols and baseline methods. We reports linear evaluation results in Section 4.3, followed by Sections 4.4 and 4.5 presenting transfer learning and few-shot experiments results. Section 4.6 provides ablation studies, and Section 4.7 analyzes system efficiency when deployed on edge devices.

4.1. Experimental setup

4.1.1. Datasets

Pretraining datasets. Two datasets are used for self-supervised pretraining and distillation. **STL-10** [41] contains 100,000 unlabeled images alongside 5000 labeled training and 8000 labeled test images distributed across ten classes. Its dedicated unlabeled split makes it well suited for evaluating unsupervised methods. **ILSVRC2012** [42] comprises 1.28M training images and 50K validation images spanning 1000 categories, and serves as the primary large-scale benchmark.

Transfer learning datasets. To assess downstream transferability, the pretrained models are evaluated on eight classification tasks: Food-101 [69] (101 classes), Oxford 102 Flowers [70] (102 classes), DTD [71] (47 classes), Oxford-IIIT Pets [72] (37 classes), CIFAR-10 and CIFAR-100 [73] (10 and 100 classes), and Leaf [74] evaluated on both a binary health classification task (2 classes) and a multi-class species classification task (12 classes). For DTD, the first official train/test split is used. For Leaf, 80% of the data is randomly selected for training and 20% for testing.

4.1.2. Network architectures

Backbones. ResNet-50 [9] serves as the teacher encoder and ResNet-18 [9] as the student encoder in all experiments. Giving $K = 4$ branches per network in our current design, we divide the convolutional layers into four stages of residual blocks with an auxiliary branch

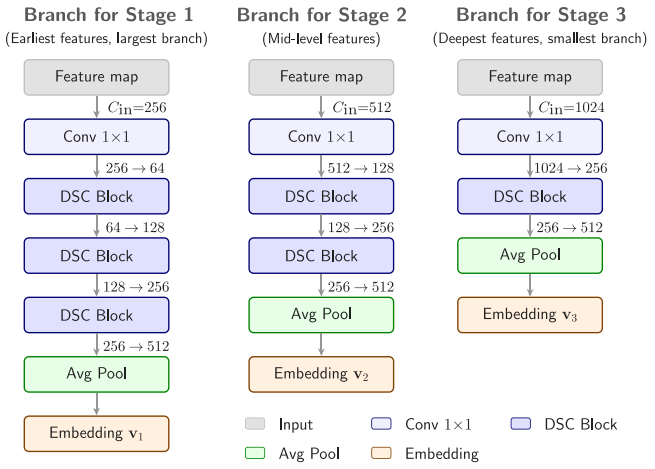

Fig. 5. Architecture of auxiliary branches for the first three intermediate stages of the teacher encoder. Each branch begins with a 1×1 convolution to reduce channel dimensionality, followed by depthwise separable convolution (DSC) blocks and global average pooling. Channel dimensions ($C_{\text{in}} \to C_{\text{out}}$) are annotated below each layer. The stage-4 branch shares the same single-DSC-block topology as stage 3 and is therefore not shown separately; it differs only in input channel dimensionality ($C_{\text{in}} = 2048$ vs. $C_{\text{in}} = 1024$).

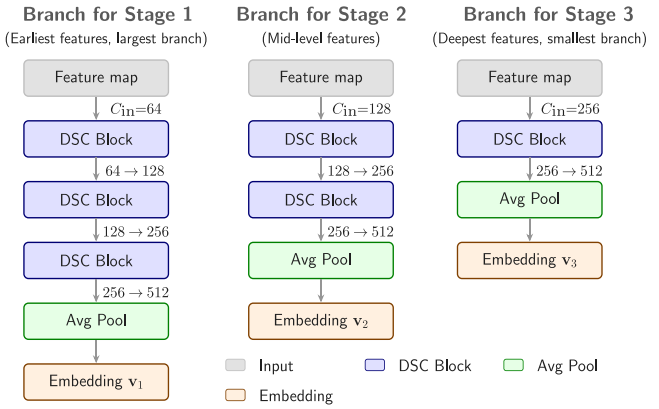

Fig. 6. Architecture of auxiliary branches for the first three intermediate stages of the student encoder. The stage-4 branch shares the same single-DSC-block topology as stage 3 and is therefore not shown separately; it differs only in input channel dimensionality ($C_{\text{in}} = 512$ vs. $C_{\text{in}} = 256$).

attached after each stage. Table 1 lists the block counts. ResNet-18 stacks two basic residual blocks per stage uniformly (2–2–2–2), while ResNet-50 uses bottleneck blocks in a 3–4–6–3 arrangement. The two backbones also differ in channel width: ResNet-50 produces 256, 512, 1024, and 2048 channels at stages 1–4 due to the bottleneck expansion, whereas ResNet-18 produces 64, 128, 256, and 512 channels. Following the standard SimCLR design [26], a two-layer MLP projection head maps the backbone output to a 128-dimensional latent space. The same projection-head architecture is used for both teacher and student to ensure a fair comparison.

Auxiliary branch configurations. The branches follow the inverse-depth design described in Section 3.3: earlier stages receive deeper branches (more DSC blocks) and later stages shallower ones. This depth decreases monotonically until it reaches a minimum of one DSC block at stages 3. Stage 4 shares the same single-DSC-block topology as stage 3, differing only in input channel dimensionality (Cin = 1024 vs. Cin = 2048 for the ResNet-50 teacher; Cin = 256 vs. Cin = 512 for the ResNet-18 student). The complete branch configurations for both encoders are illustrated in Figs. 5 and 6. Because ResNet-50's bottleneck layers produce high-channel feature maps (up to 2048 channels), a 1×1 convolution is prepended to each ResNet-50 branch to reduce the channel dimensionality before the DSC blocks; ResNet-18 branches receive stage outputs directly. The same configurations are used for both STL-10 and ILSVRC2012 experiments.

4.1.3. Training configuration

Self-supervised pretraining. Both teacher and baseline student are pretrained using SimCLR. On ILSVRC2012, models are trained for 100 epochs with a batch size of 256, using the LARS optimizer [75] with an initial learning rate of 0.6, weight decay of 10^{-6}, a 10-epoch linear warmup, and cosine annealing. The contrastive temperature is set to $\tau = 0.1$. The same optimizer and schedule are used on STL-10.

Data augmentation. The augmentation pipeline follows the standard SimCLR protocol: random resized crop (scale range $[0.08, 1.0]$, aspect ratio range $[3/4, 4/3]$), random horizontal flip ($p = 0.5$), color jittering ($p = 0.8$), random grayscale ($p = 0.2$), and Gaussian blur ($p = 0.5$, kernel size 10% of the image dimension). The crop size is 96×96 for STL-10 and 224×224 for ILSVRC2012.

Teacher auxiliary branch training (Stage 1). The teacher backbone is frozen and only the auxiliary branches are optimized using the same optimizer, schedule, and contrastive temperature as pretraining.

Student distillation (Stage 2). The distillation temperature is set to $\tau = 0.5$; the effect of this choice is analyzed in the ablation study (Section 4.6). The auxiliary-branch and projection-head losses are summed with equal weight.

4.2. Evaluation protocols and baselines

We consider both linear evaluation and fine-tuning settings.

Linear evaluation. The pretrained backbone is frozen and a linear classifier is trained on top of the extracted features. On STL-10, the classifier is trained for 100 epochs with SGD (learning rate 0.0003, momentum 0.9, batch size 256); input images are randomly cropped to 96×96 with horizontal flipping during training and resized to 96×96 during testing. On ILSVRC2012, the classifier is trained for 100 epochs with SGD (learning rate 0.8, momentum 0.9, batch size 2048); images are randomly cropped to 224×224 during training, and resized to 256 pixels on the short side followed by a 224×224 center crop during testing.

Fine-tuning. The entire network (backbone + classifier) is jointly optimized on the target task for 200 epochs with SGD (Nesterov momentum 0.9, batch size 256). The learning rate is selected via grid search over $[0.001, 0.1]$ (log-spaced) and weight decay over $[10^{-6}, 10^{-3}]$ (log-spaced). Training augmentation consists of random resized crops to 224×224 and horizontal flips; test images are resized to 256 pixels on the short side and center-cropped to 224×224.

Baselines. The proposed **MLCKD** uses a pretrained ResNet-50 teacher to distill knowledge to a ResNet-18 student. We compare MLCKD against baselines in three groups: (i) **SimCLR (ResNet-50)** represents the upper bound of the teacher architecture, and (ii) **SimCLR (ResNet-18)** and **MoCo-V2 (ResNet-18)** represent standalone self-supervised learning on the student architecture without distillation. MoCo-V2 [28,76] is a momentum-based contrastive method trained directly on the student; (iii) lastly, we consider **SEED** [36], a self-supervised *distillation* method that, similar to MLCKD, transfers knowledge from the same ResNet-50 teacher to the ResNet-18 student, but distills only the teacher's final-embedding similarity structure. Comparing MLCKD against the standalone ResNet-18 baselines isolates the contribution of distillation; comparing against SEED isolates the contribution of *multi-level* transfer, since SEED shares MLCKD's final-level objective and differs only in the absence of intermediate-level distillation; and comparison against the ResNet-50 baseline shows how closely the student can approach the teacher's representation quality.

Table 2
Top-1 accuracy under linear evaluation. The best student result is highlighted in **bold**. We consider two standalone self-supervised baselines, i.e., SimCLR and MoCO-V2 with ResNet-18, and one final-embedding self-supervised distillation baseline, i.e., SEED. The reported improvements and gaps are absolute percentage points (pp), i.e., the arithmetic difference between two top-1 accuracies.

Method	Backbone	STL-10	ILSVRC2012
SimCLR (teacher)	ResNet-50	0.85	0.58
Standalone self-supervised (no distillation)			
SimCLR	ResNet-18	0.78	0.46
MoCo-V2	ResNet-18	0.76	0.44
Self-supervised distillation (ResNet-50 → ResNet-18)			
SEED	ResNet-18	0.81	0.47
MLCKD (ours)	ResNet-18	**0.84**	**0.49**
Improvement over SimCLR ResNet-18		+6 pp	+3 pp
Gap with SimCLR ResNet-50		−1 pp	−9 pp
Improvement over SEED		+3 pp	+2 pp

4.3. Linear evaluation results

Table 2 reports the top-1 classification accuracy under the linear evaluation protocol on STL-10 and ILSVRC2012.

Compared with the SimCLR ResNet-50 teacher, the MLCKD-trained ResNet-18 student closes the accuracy gap to within 1 percentage point on STL-10 (0.84 vs. 0.85), nearly matching the teacher despite having 2.25× fewer FLOPs and 2.19× fewer parameters (as reported later in Table 5). On ILSVRC2012, a larger gap of 9 percentage points remains (0.49 vs. 0.58). This disparity between the two datasets can be attributed to the difference in task complexity and data scale. STL-10 contains 100K unlabeled images across a relatively small number of visual categories, and the teacher's representational advantage over the student is modest to begin with; multi-level distillation is sufficient to transfer nearly all of this advantage. ILSVRC2012, by contrast, spans 1000 fine-grained categories over 1.28M images, placing substantially greater demands on model capacity. In this setting, the architectural bottleneck of the smaller ResNet-18 backbone limits how much of the teacher's knowledge can be absorbed, regardless of the distillation strategy. Nevertheless, the 3 percentage point improvement over the ResNet-18 baseline demonstrates that MLCKD extracts meaningful additional knowledge from the teacher even on this challenging benchmark. We note that further gains may be achievable by extending pretraining beyond the 100 epochs used in our experiments (compared with the 800–1000 epochs typical in the SimCLR literature), or by employing a more expressive student architecture, both of which would narrow the remaining capacity gap.

For the self-supervised distillation baseline SEED, it distills the similarity structure of the teacher's final embedding alone, whereas MLCKD distills relational structure at both intermediate and final levels. SEED is therefore a direct ablation of the multi-level design. As shown in Table 2, MLCKD improves over SEED by 3 and 2 percentage points on STL-10 and ILSVRC2012 linear evaluation, respectively, confirming that intermediate-level transfer supplies complementary structural knowledge that the student cannot recover from output-level distillation alone. The two standalone self-supervised baselines, SimCLR and MoCo-V2, cluster within 2 percentage points of each other on both datasets, indicating that the weak representations of the compact student arise from standalone contrastive pretraining on a low-capacity backbone rather than from any property specific to SimCLR. Between the two standalone baselines, MoCo-V2 trails SimCLR by 2 pp on both datasets. This is because MoCo-V2 relies on a momentum-updated memory queue whose stored features must stay consistent with the slowly evolving encoder, a condition that is only met after long pretraining (the original setup uses 800 epochs). Under our 100-epoch budget the queue is comparatively stale, weakening the contrastive signal, whereas SimCLR draws negatives from the current batch and is less affected by the shortened schedule.

4.4. Transfer learning results

Table 3 reports fine-tuning accuracy across eight downstream tasks (i.e., the eight transfer learning datasets mentioned in Section 4.1.1) for models pretrained on ILSVRC2012.

Comparison with the Baseline (SimCLR ResNet-18). MLCKD outperforms the SimCLR ResNet-18 baseline on all eight transfer learning tasks. These consistent gains across datasets that differ substantially in scale (from 2040 training images for Flowers to 50,000 for CIFAR-100), granularity (fine-grained species and texture recognition versus coarse object classification), and visual domain (natural scenes, textures, medical leaves) confirm that multi-level relational distillation produces representations with broad downstream transferability. The improvements are notably larger on tasks that demand fine-grained discrimination, i.e., DTD (textures), CIFAR-100 (100 categories), and Leaf-Species (12 species), each see +5 percentage point gains, suggesting that the intermediate-layer knowledge distilled by MLCKD captures mid-level spatial patterns and structural cues that are particularly valuable when downstream categories share similar global appearance and must be distinguished by subtle local differences.

Comparison with the Teacher (SimCLR ResNet-50). The gap between the MLCKD-trained ResNet-18 student and the ResNet-50 teacher varies across tasks. On CIFAR-10, Leaf-Binary, and Leaf-Species, the student comes within 1 percentage point of the teacher, effectively closing the capacity gap on these benchmarks. On CIFAR-100 and DTD the gap narrows to 2–3 percentage points, and on Oxford Pets and Food-101 it remains at 3–4 percentage points. The largest residual gap appears on Flowers (8 percentage points, 0.74 vs. 0.82). We can see that the student recovers a larger fraction of the teacher's accuracy on tasks with fewer categories or with categories that are visually distinct (e.g., CIFAR-10 with 10 broad classes, Leaf-Binary with 2 classes), whereas the gap persists on tasks that require finer inter-class discrimination across many visually similar categories (e.g., Flowers with 102 species). This is consistent with the capacity bottleneck observed in the linear evaluation results (Section 4.3): the ResNet-18 backbone has limited representational bandwidth, and fine-grained distinctions among a large number of categories are the first to suffer when capacity is reduced. Importantly, MLCKD consistently narrows this gap relative to the SimCLR ResNet-18 baseline. On Flowers, for instance, the baseline trails the teacher by 10 percentage points, whereas MLCKD reduces this to 8. This indicates that multi-level distillation partially compensates for the architectural limitation even on the most challenging tasks.

Comparison with Supervised Training. MLCKD surpasses supervised training from random initialization (Supervised†) on all eight tasks, with margins ranging from +8 percentage points (Flowers) to +20 percentage points (CIFAR-10). This result underscores two practical advantages for edge deployment. First, self-supervised pretraining with distillation provides a stronger initialization than random weights even when labeled data is subsequently available for fine-tuning, confirming that the teacher's relational knowledge transfers complementary information beyond what task-specific supervision alone can provide.

Comparison with MoCo-V2 (ResNet-18). MoCo-V2 falls below SimCLR (ResNet-18) on seven of eight tasks, with the largest deficits on Leaf-Species (−6 pp), CIFAR-10 (−5 pp), and Oxford Pets (−3 pp). As discussed in Section 4.3, this is because MoCo-V2's momentum queue requires long pretraining to maintain consistent encoder representations, which is beyond the 100-epoch budget we considered in our setting.

Comparison with SEED (ResNet-18). SEED improves over the SimCLR (ResNet-18) baseline on all eight tasks by 1–4 pp, showing that final-embedding distillation alone is already beneficial. MLCKD improves further over SEED on every task, with margins of 1 pp on

Table 3
Fine-tuning accuracy on transfer learning tasks. Best student results are in **bold**.

Dataset	Sup.[a] (R18)	SimCLR (R50)	SimCLR (R18)	MoCo-V2 (R18)	SEED (R18)	MLCKD (R18)
Food-101 [69]	0.65	0.81	0.73	0.72	0.75	**0.77**
Flowers [70]	0.66	0.82	0.72	0.70	0.73	**0.74**
DTD [71]	0.50	0.64	0.56	0.54	0.60	**0.61**
Oxford Pets [72]	0.61	0.78	0.71	0.68	0.72	**0.75**
CIFAR-10 [73]	0.75	0.96	0.90	0.85	0.92	**0.95**
CIFAR-100 [73]	0.64	0.79	0.72	0.70	0.73	**0.77**
Leaf-Binary [74]	0.89	0.99	0.94	0.94	0.95	**0.98**
Leaf-Species [74]	0.75	0.94	0.88	0.82	0.89	**0.93**

[a] Supervised training from random initialization. R50/R18: ResNet-50/ResNet-18.

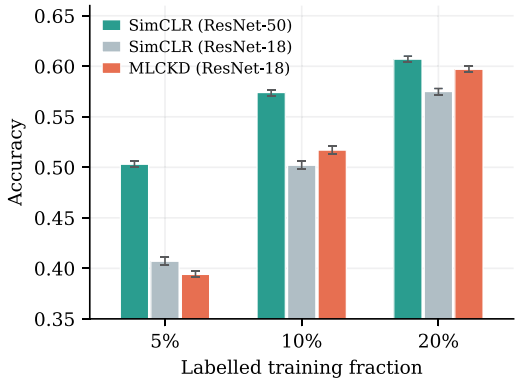

Fig. 7. Few-shot fine-tuning accuracy on CIFAR-100 with 5%, 10%, and 20% of labeled data.

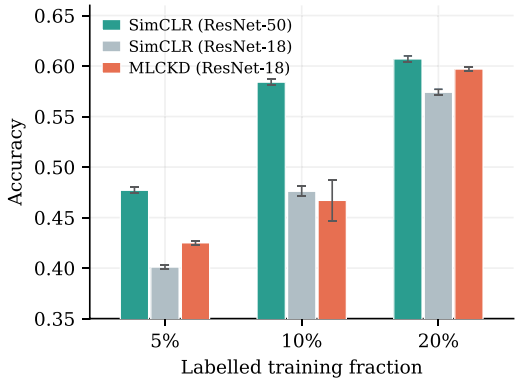

Fig. 8. Few-shot fine-tuning accuracy on Food-101 with 5%, 10%, and 20% of labeled data.

Flowers and DTD rising to 4 pp on CIFAR-100 and Leaf-Species. Because SEED and MLCKD share the same frozen teacher, the same student backbone, and the same final-level distillation objective, this consistent gap is attributable solely to the intermediate auxiliary branches introduced by MLCKD. The gains are largest on fine-grained tasks, i.e., CIFAR-100, Leaf-Species, and Oxford Pets, where mid-level spatial and structural cues matter most, and smallest on coarser tasks such as Leaf-Binary and CIFAR-10 where global semantics already provide sufficient discriminative signal. This pattern directly supports the claim that intermediate-level transfer captures complementary knowledge that the final embedding alone cannot convey.

4.5. Few-shot fine-tuning results

To evaluate representation quality under limited labeled data, we fine-tune on CIFAR-100 and Food-101 using 5%, 10%, and 20% of the labeled training data. Results are illustrated in Figs. 7 and 8. On CIFAR-100, MLCKD outperforms the SimCLR ResNet-18 baseline at 10% and 20% label fractions. On Food-101, MLCKD matches or exceeds the baseline at 5% and 20%, and is underperformed only in the 10% setting (0.47 vs. 0.46). However, given the overlapping error bars, suggests this reflects run-to-run variance rather than a systematic failure of the proposed distillation strategy. With 20% labeled data MLCKD surpasses the baseline on and closes within 1 pp of the ResNet-50 teacher in both datasets, which confirms that the distilled representations generalize effectively once a modest amount of task-specific supervision is available.

Table 4
Effect of temperature hyperparameters (τ_1, τ_2) on linear evaluation accuracy (with STL-10 dataset). Lower τ_1 sharpens auxiliary-branch contrast; higher τ_2 softens the distillation target, allowing the student to capture finer relational structure.

τ_1 (Stage 1)	τ_2 (Stage 2)	Top-1 Acc.
0.1	0.1	0.74
0.1	0.5	**0.82**
0.5	0.1	0.81
0.5	0.5	0.79

4.6. Effect of temperature hyperparameters

The proposed method uses two temperature parameters: τ_1 for training the teacher auxiliary branches (Stage 1) and τ_2 for student distillation (Stage 2). Their effects are evaluated on STL-10 under the linear evaluation protocol, with results shown in Table 4. The two temperature parameters serve distinct roles, and the ablation reveals a clear asymmetry in their optimal settings. The Stage-1 temperature τ_1 controls the sharpness of the contrastive signal used to train the auxiliary branches. A lower τ_1 concentrates gradients on hard negatives, producing more discriminative intermediate embeddings. The Stage-2 temperature τ_2 governs how the teacher's similarity matrix is softened before the student matches it. A higher τ_2 flattens the distribution, exposing subtler inter-instance relations and providing richer supervision. The strongest effect in the ablation is the 8 percentage point gain from raising τ_2 from 0.1 to 0.5 at fixed $\tau_1 = 0.1$ (0.74 $\rightarrow$ 0.82), confirming that a soft distillation target is critical when branch embeddings are already sharp.

The worst configuration ($\tau_1 = 0.1$, $\tau_2 = 0.1$, accuracy 0.74) collapses both stages to near-one-hot similarity distributions, discarding the relational structure that distillation is designed to transfer. Conversely, the symmetric soft setting ($\tau_1 = 0.5$, $\tau_2 = 0.5$, accuracy 0.79) undersharpens the branch embeddings, reducing their discriminability. The optimal configuration is therefore *asymmetric*: sharp branch training ($\tau_1 = 0.1$) paired with soft distillation ($\tau_2 = 0.5$). This is consistent with the dark-knowledge principle of Hinton et al. [14], i.e., higher temperature during distillation reveals richer teacher knowledge. Our results extend it to relational distillation at intermediate representation levels.

Table 5
Model complexity comparison between teacher and student. The ResNet-18 student requires 2.25× fewer FLOPs and 2.19× fewer parameters than the ResNet-50 teacher for a single $3 \times 224 \times 224$ input, enabling practical deployment on resource-constrained edge devices.

Model	FLOPs (G)	Parameters (M)
ResNet-50 (teacher)	4.09	25.56
ResNet-18 (student)	1.82	11.69
Reduction	2.25×	2.19×

Table 6
Specifications of the edge devices used for inference benchmarking. The three platforms span a wide range of compute capability, from the resource-constrained Nano to the high-performance AGX Orin.

Device	CPU Cores	CPU Freq.	RAM	GPU Cores	GPU Freq.
Jetson Nano	4	1.43 GHz	4 GB	128	0.91 GHz
Jetson Xavier NX	6	1.90 GHz	16 GB	384	1.10 GHz
Jetson AGX Orin	12	2.20 GHz	32 GB	2048	1.30 GHz

4.7. System efficiency on edge devices

A central motivation for this work is enabling the deployment of high-quality self-supervised representations on the resource-constrained hardware found in ad hoc and sensor network nodes. While the preceding sections demonstrate that MLCKD improves representation quality, practical deployment additionally requires that the distilled model fits within the compute, memory, and energy budgets of edge platforms. This subsection quantifies these deployment characteristics by comparing the ResNet-18 student against the ResNet-50 teacher in terms of model complexity and measured inference latency on three NVIDIA Jetson edge devices that span a representative range of edge computing capability.

4.7.1. Model complexity

Table 5 compares the computational cost of the teacher and student architectures. The ResNet-18 student requires 2.25× fewer FLOPs and 2.19× fewer parameters than the ResNet-50 teacher. In absolute terms, this corresponds to a reduction from 4.09G to 1.82G FLOPs and from 25.56M to 11.69M parameters. These are savings that directly translate to lower memory footprint, reduced storage requirements, and lower per-inference energy consumption on edge hardware. Crucially, these savings are achieved while *improving* representation quality over the SimCLR ResNet-18 baseline (Section 4.3), confirming that MLCKD does not merely compress the model but actively enhances the student's learned features relative to what the same architecture can learn independently.

4.7.2. Inference latency on edge devices

To validate that the theoretical complexity reduction translates to real-world speedups, we benchmark inference latency on three NVIDIA Jetson platforms that represent distinct points on the edge computing spectrum: the Jetson Nano (128 GPU cores, 4 GB RAM), the Jetson Xavier NX (384 GPU cores, 16 GB RAM), and the Jetson AGX Orin (2048 GPU cores, 32 GB RAM). Device specifications are listed in Table 6, and latency results are reported in Table 7.

The ResNet-18 student achieves 1.60× to 2.79× inference speedup over the ResNet-50 teacher across the three platforms. Two observations are worth highlighting. First, the speedup is most pronounced on the Jetson Nano (2.79×), the most resource-constrained device in the evaluation. On this platform, the ResNet-50 teacher requires 32.1 ms per sample, exceeding the 33 ms budget required for 30 fps real-time processing only marginally. By contrast, the ResNet-18 student completes inference in 11.5 ms, comfortably meeting the real-time constraint. This demonstrates that MLCKD enables real-time visual inference on ultra-low-cost edge hardware where the teacher model would be borderline infeasible. Second, even on the high-end Jetson AGX Orin, where both models run in under 4 ms, the student still provides a 1.60× speedup (3.2 ms → 2.0 ms). In latency-sensitive applications such as multi-sensor fusion in autonomous systems, these per-frame savings compound across thousands of inferences per second, reducing both cumulative latency and energy consumption.

Table 7
Average inference latency (ms) per sample on three NVIDIA Jetson edge platforms, measured over 50 forward passes with a $3 \times 224 \times 224$ input. The ResNet-18 student achieves 1.6×–2.8× speedup over the ResNet-50 teacher, with the largest gain on the most resource-constrained device.

Device	ResNet-50 (SimCLR)	ResNet-18 (MLCKD)	Speedup
Jetson Nano	32.1	11.5	2.79×
Jetson Xavier NX	4.8	2.7	1.77×
Jetson AGX Orin	3.2	2.0	1.60×

4.7.3. Energy efficiency and memory footprint

Beyond inference latency, two further operational metrics are relevant for resource-constrained deployment: energy consumption per inference and peak memory usage.

Energy per inference. On Jetson devices the GPU draws power at approximately its configured Thermal Design Power (TDP) ceiling during active inference, regardless of which model is running [77,78]. The instantaneous power is therefore a property of the device and its power mode rather than the model, i.e., the Jetson Nano operates at 5–10 W [79] and the AGX Orin at up to 60 W [78], with the power draw remaining approximately constant across models on the same platform under the same power-mode setting [80]. Since power is approximately fixed during inference and only inference time differs, energy per inference (Joules = Watts × seconds) scales proportionally with latency. Therefore, the 1.6×–2.8× latency reduction reported in Table 7 implies a corresponding reduction in per-inference energy, with the largest gain on the Jetson Nano (2.79×), where the resource constraints make energy efficiency most critical.

Peak memory footprint. Inference memory consists of two components: weight storage and intermediate activations. In FP32, ResNet-50 requires approximately 150–200 MB (weights ≈102 MB plus peak activations) and ResNet-18 approximately 70–110 MB (weights ≈47 MB plus peak activations), reflecting the 2.19× parameter reduction reported in Table 5. Both models fall well within the 4 GB unified memory of the most constrained platform tested, the Jetson Nano, confirming that memory is not the binding deployment constraint on any of the three platforms. Nevertheless, the student's ∼2× smaller memory footprint, which follows directly from the 2.19× parameter reduction reported in Table 5, provides meaningful memory saving for concurrent processes such as sensor data acquisition and communication stacks that run alongside the inference pipeline in real sensor network deployments.

Training cost. MLCKD introduces two additional training stages beyond standalone SimCLR pretraining. In Stage 1, the teacher backbone is kept fully frozen; only the $K = 4$ auxiliary branches, each a small stack of depthwise separable convolution blocks, are trained. The per-iteration cost is therefore a single forward pass through the ResNet-50 teacher (4.09G FLOPs, no backward) plus the forward and backward passes through the four compact branches. In Stage 2, the teacher again remains frozen, so the per-iteration cost is one forward pass through ResNet-50 (4.09G FLOPs, no backward) plus the full forward and backward pass through the ResNet-18 student backbone, its four auxiliary branches, and its projection head ($\approx 2 \times 1.82\text{G} = 3.64\text{G}$ FLOPs). Stage 2 therefore adds approximately the forward cost of the ResNet-50 teacher on top of baseline SimCLR ResNet-18 training, resulting in roughly 2× the per-iteration compute of standalone student pretraining. However, it is important to note that this overhead is a one-time offline expense: training is performed once on a server, and the auxiliary

branches and projection heads are discarded afterwards. The deployed model is an unmodified ResNet-18 whose inference cost is identical to the standalone baseline.

Summary. Taken together, these results establish that MLCKD achieves a genuine accuracy-efficiency trade-off rather than merely a compression of the teacher. The distilled ResNet-18 student delivers stronger representations than standalone self-supervised learning, approaches the teacher's accuracy on most benchmarks, runs 1.6×–2.8× faster, consumes proportionally less energy per inference, and occupies roughly half the memory — all without changing the deployed architecture beyond a standard ResNet-18. For sensor network and ad hoc deployments, where inference must run continuously on fixed hardware budgets, this combination of properties matters more than any single metric in isolation.

5. Discussion and conclusion

This paper presented MLCKD, an unsupervised knowledge distillation framework that transfers multi-level relational knowledge from a contrastively pretrained teacher to a lightweight student using only unlabeled data. The preceding experiments establish that multi-level distillation consistently outperforms both standalone contrastive learning and final-embedding-only distillation (SEED), with the largest gains on fine-grained tasks where intermediate spatial and structural cues are most discriminative, and that these accuracy gains transfer directly to the edge hardware setting as a 1.6×–2.8× inference speedup without modifying the deployed architecture.

5.1. Limitations and future directions

The edge evaluation in this work focuses on high-performance edge platforms, i.e., the Jetson family, which are representative of capable edge devices but sit at the upper end of the embedded computing spectrum. In practice, wireless sensor network deployments are dominated by far more constrained hardware, i.e., microcontrollers such as ESP32 or ARM Cortex-M4 class devices with 512 KB to 8 MB of RAM and no floating-point GPU acceleration [81]. On these platforms a 47 MB ResNet-18 model is infeasible without aggressive compression. A single distilled student is therefore unlikely to be Pareto-optimal across a heterogeneous node population; different node classes would require models of different capacity, which the current single-student design cannot accommodate directly. Combining MLCKD with post-training quantization [82,83], heterogeneous and architecture-specific distillation [84,85], or input-side optimization [86,87], are promising directions for making MLCKD applicable across the full hardware diversity of real sensor network deployments.

The current framework assumes homogeneous architectures in which teacher and student share the same backbone family to enable straightforward stage-wise alignment of auxiliary branches. However, the most capable self-supervised teachers available today are Vision Transformers (DINO [32], DINOv2 [33]), and a CNN student cannot straightforwardly benefit from them under the current design. Extending MLCKD to *heterogeneous* teacher–student pairs would require architecture-agnostic alignment strategies [84,85] and is an active open problem in the knowledge distillation literature [15]. The main difficulty is that ViTs produce sequences of patch tokens with global receptive fields, while CNNs produce spatially organized feature maps with local structure, so the branch-matching approach that MLCKD relies on does not transfer directly. Recent work on cross-architecture KD addresses this through several strategies. One approach introduces dedicated projectors that map CNN feature maps into spaces compatible with the transformer teacher's token representations [88]. Another line of work targets the stage-to-layer correspondence problem directly. Rather than assuming a fixed, hand-defined mapping between teacher stages and student layers, cross-layer attention mechanisms dynamically determine which teacher representations are most informative for each student stage at training time [84,88]. However, these methods have so far been studied in supervised settings; whether they can be adapted to the relational, label-free, unsupervised setting in MLCKD is an open question.

Third, the equal weighting of auxiliary-branch and projection-head losses in the current formulation follows the default design choice used in feature-based knowledge distillation literature [16,39,66,67]. Equal weighting is defensible in the current design because every term is a KL divergence over the same mini-batch, so the losses are already commensurate. But it assumes each stage contributes equally transferable knowledge, which is unlikely to hold across tasks. Developing adaptive or learned weighting schemes could improve the distillation efficiency further. For stance, one solution is loss-proportional weighting [89], where the weight assigned to each stage at each training step is proportional to its current KL divergence, so supervision concentrates on whichever branch is furthest from the teacher. Another option is to learn the stage weights through a meta-network [68], which can discover cross-stage importance patterns that are not predictable from loss magnitudes alone. However, these schemes are used for supervised learning, and whether any of these gains carry over from supervised to the relational, label-free objective in MLCKD is an open question for future work.

CRediT authorship contribution statement

Ting Liang: Writing – original draft, Software, Methodology. **Lingyu Du:** Software, Methodology, Investigation. **Guohao Lan:** Writing – review & editing, Writing – original draft, Supervision, Project administration, Methodology, Investigation, Funding acquisition, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the "Agrifood 11: Glastuinbouw – Digital Twin" research project under the Dutch National Growth Fund programme NXTGEN Hightech.

Data availability

The authors do not have permission to share data.

References

[1] M. Zakaria, E. Karaaslan, F.N. Catbas, Advanced bridge visual inspection using Real-Time machine learning in edge devices, Adv. Bridg. Eng. 4 (1) (2023) 1–16.
[2] D. Tuia, B. Kellenberger, S. Beery, B.R. Costelloe, S. Zuffi, B. Risse, A. Mathis, M.W. Mathis, F. van Langevelde, T. Burghardt, R. Kays, et al., Perspectives in machine learning for wildlife conservation, Nat. Commun. 13 (1) (2022) 792.
[3] A. Karimi, D.-A. Duecker, A. Saeidi, A comprehensive review on autonomous navigation, ACM Comput. Surv. 57 (8) (2025) 1–66.
[4] M. Bi, Y. Wang, Z. Cai, X. Tong, A Privacy-Preserving mechanism based on local differential privacy in edge computing, China Commun. 17 (9) (2020) 50–65, http://dx.doi.org/10.23919/JCC.2020.09.005.
[5] J. Chen, X. Ran, Deep learning with edge computing: A review, Proc. IEEE 107 (8) (2019) 1655–1674.
[6] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge computing: Vision and challenges, IEEE Internet Things J. 3 (5) (2016) 637–646.
[7] Y. Mao, C. You, J. Zhang, K. Huang, K.B. Letaief, A survey on mobile edge computing: The communication perspective, IEEE Commun. Surv. Tutor. 19 (4) (2017) 2322–2358.
[8] G. Menghani, Efficient deep learning: A survey on making deep learning models smaller, faster, and better, ACM Comput. Surv. 55 (12) (2023) 1–37.

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

[10] A. Gholami, S. Kim, Z. Dong, Z. Yao, M.W. Mahoney, K. Keutzer, A survey of quantization methods for efficient neural network inference, in: Low-Power Computer Vision, Chapman and Hall/CRC, 2022, pp. 291–326.

[11] H. Cheng, M. Zhang, J.Q. Shi, A survey on deep neural network pruning—Taxonomy, comparison, analysis, and recommendations, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 10558–10578.

[12] T. Liang, J. Glossner, L. Wang, S. Shi, X. Zhang, Pruning and quantization for deep neural network acceleration: A survey, Neurocomputing 461 (2021) 370–403.

[13] A.G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, H. Adam, MobileNets: Efficient convolutional neural networks for mobile vision applications, 2017, arXiv preprint arXiv:1704.04861.

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

[15] J. Gou, B. Yu, S.J. Maybank, D. Tao, Knowledge distillation: A survey, Int. J. Comput. Vis. 129 (6) (2021) 1789–1819.

[16] S. Zagoruyko, N. Komodakis, Paying more attention to attention: Improving the performance of convolutional neural networks via attention transfer, in: 5th International Conference on Learning Representations, ICLR, Toulon, France, 2017.

[17] W. Park, D. Kim, Y. Lu, M. Cho, Relational knowledge distillation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2019, pp. 3967–3976.

[18] Y. Tian, D. Krishnan, P. Isola, Contrastive representation distillation, in: International Conference on Learning Representations, ICLR, 2020.

[19] N. Tajbakhsh, L. Jeyaseelan, Q. Li, J.N. Chiang, Z. Wu, X. Ding, Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation, Med. Image Anal. 63 (2020) 101693.

[20] N. Rieke, J. Hancox, W. Li, F. Milletari, H.R. Roth, S. Albarqouni, S. Bakas, M.N. Galtier, B.A. Landman, K. Maier-Hein, et al., The future of digital health with federated learning, Npj Digit. Med. 3 (1) (2020) 1–7.

[21] L. Ericsson, H. Gouk, C.C. Loy, T.M. Hospedales, Self-Supervised representation learning: Introduction, advances, and challenges, IEEE Signal Process. Mag. 39 (3) (2022) 42–62.

[22] J. Gui, T. Chen, J. Zhang, Q. Cao, Z. Sun, H. Luo, D. Tao, A survey on Self-Supervised learning: Algorithms, applications, and future trends, IEEE Trans. Pattern Anal. Mach. Intell. 46 (12) (2024) 9052–9071.

[23] A. Jaiswal, A.R. Babu, M.Z. Zadeh, D. Banerjee, F. Makedon, A survey on contrastive Self-Supervised learning, Technologies 9 (1) (2020) 2.

[24] L. Jing, Y. Tian, Self-Supervised visual feature learning with deep neural networks: A survey, IEEE Trans. Pattern Anal. Mach. Intell. 43 (11) (2020) 4037–4058.

[25] I. Misra, L.v.d. Maaten, Self-Supervised learning of Pretext-Invariant representations, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 6707–6717.

[26] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, ICML, PMLR, 2020, pp. 1597–1607.

[27] P.H. Le-Khac, G. Healy, A.F. Smeaton, Contrastive representation learning: A framework and review, IEEE Access 8 (2020) 193907–193934.

[28] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 9729–9738.

[29] J.-B. Grill, F. Strub, F. Altché, C. Tallec, P. Richemond, E. Buchatskaya, C. Doersch, B. Avila Pires, Z. Guo, M. Gheshlaghi Azar, B. Piot, K. Kavukcuoglu, R. Munos, M. Valko, Bootstrap your own latent – A new approach to self-supervised learning, in: Advances in Neural Information Processing Systems, NeurIPS, Vol. 33, 2020, pp. 21271–21284.

[30] J. Zbontar, L. Jing, I. Misra, Y. LeCun, S. Deny, Barlow twins: Self-Supervised learning via redundancy reduction, in: Proceedings of the 38th International Conference on Machine Learning, ICML, 2021, pp. 12310–12320.

[31] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, A. Joulin, Unsupervised learning of visual features by contrasting cluster assignments, in: Advances in Neural Information Processing Systems, NeurIPS, Vol. 33, 2020.

[32] M. Caron, H. Touvron, I. Misra, H. Jégou, J. Mairal, P. Bojanowski, A. Joulin, Emerging properties in Self-Supervised vision transformers, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, ICCV, 2021, pp. 9650–9660.

[33] M. Oquab, T. Darcet, T. Moutakanni, H. Vo, M. Szafraniec, V. Khalidov, P. Fernandez, D. Haziza, F. Massa, A. El-Nouby, M. Assran, N. Ballas, W. Galuba, R. Howes, P.-Y. Huang, S.-W. Li, I. Misra, M. Rabbat, V. Sharma, G. Synnaeve, H. Xu, H. Jégou, J. Mairal, P. Labatut, A. Joulin, P. Bojanowski, DINOv2: Learning robust visual features without supervision, Trans. Mach. Learn. Res. (TMLR) (2024).

[34] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, G.E. Hinton, Big Self-Supervised models are strong Semi-Supervised learners, Adv. Neural Inf. Process. Syst. 33 (2020) 22243–22255.

[35] G. Xu, Z. Liu, X. Li, C.C. Loy, Knowledge distillation meets Self-Supervision, in: European Conference on Computer Vision, ECCV, Springer, 2020, pp. 588–604.

[36] Z. Fang, J. Wang, L. Wang, L. Zhang, Y. Yang, Z. Liu, SEED: Self-Supervised distillation for visual representation, 2021, arXiv preprint arXiv:2101.04731.

[37] Y. Gao, J.-X. Zhuang, S. Lin, H. Cheng, X. Sun, K. Li, C. Shen, DisCo: Remedying Self-Supervised learning on lightweight models with distilled contrastive learning, in: European Conference on Computer Vision, ECCV, Springer, 2022, pp. 237–253.

[38] K. Song, J. Xie, S. Zhang, Z. Luo, Multi-Mode online knowledge distillation for Self-Supervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 11848–11857.

[39] A. Romero, N. Ballas, S.E. Kahou, A. Chassang, C. Gatta, Y. Bengio, FitNets: Hints for thin deep nets, in: International Conference on Learning Representations, ICLR, 2015.

[40] S. Abbasi Koohpayegani, A. Tejankar, H. Pirsiavash, CompRess: Self-Supervised learning by compressing representations, in: Advances in Neural Information Processing Systems, NeurIPS, 33, 2020.

[41] A. Coates, A. Ng, H. Lee, An analysis of Single-Layer networks in unsupervised feature learning, in: Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics (AISTATS), JMLR Workshop and Conference Proceedings, 2011, pp. 215–223.

[42] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., ImageNet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (2015) 211–252.

[43] D. Pathak, P. Krahenbuhl, J. Donahue, T. Darrell, A.A. Efros, Context encoders: Feature learning by inpainting, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2016, pp. 2536–2544.

[44] S. Gidaris, P. Singh, N. Komodakis, Unsupervised representation learning by predicting image rotations, 2018, arXiv preprint arXiv:1803.07728.

[45] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving Jigsaw puzzles, in: European Conference on Computer Vision, ECCV, Springer, 2016, pp. 69–84.

[46] A. Dosovitskiy, J.T. Springenberg, M. Riedmiller, T. Brox, Discriminative unsupervised feature learning with convolutional neural networks, Adv. Neural Inf. Process. Syst. 27 (2014).

[47] Z. Wu, Y. Xiong, S.X. Yu, D. Lin, Unsupervised feature learning via Non-Parametric instance discrimination, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 3733–3742.

[48] K. He, X. Chen, S. Xie, Y. Li, P. Dollár, R. Girshick, Masked autoencoders are scalable vision learners, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 16000–16009.

[49] H. Bao, L. Dong, S. Piao, F. Wei, BEiT: BERT Pre-Training of image transformers, in: International Conference on Learning Representations, ICLR, 2022.

[50] Z. Xie, Z. Zhang, Y. Cao, Y. Lin, J. Bao, Z. Yao, Q. Dai, H. Hu, SimMIM: A simple framework for masked image modeling, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2022, pp. 9653–9663.

[51] M. Zheng, S. You, F. Wang, C. Qian, C. Zhang, X. Wang, C. Xu, ReSSL: Relational Self-Supervised learning with weak augmentation, Adv. Neural Inf. Process. Syst. 34 (2021) 2543–2555.

[52] X. Wang, G.-J. Qi, Contrastive learning with stronger augmentations, IEEE Trans. Pattern Anal. Mach. Intell. 45 (5) (2022) 5549–5560.

[53] Y. Tian, C. Sun, B. Poole, D. Krishnan, C. Schmid, P. Isola, What makes for good views for contrastive learning? in: Advances in Neural Information Processing Systems, NeurIPS, Vol. 33, 2020.

[54] M. Tan, Q.V. Le, EfficientNet: Rethinking model scaling for convolutional neural networks, in: Proceedings of the 36th International Conference on Machine Learning, ICML, 2019, pp. 6105–6114.

[55] E.J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen, LoRA: Low-Rank adaptation of large language models, in: International Conference on Learning Representations, ICLR, 2022.

[56] T. Choudhary, V. Mishra, A. Goswami, J. Sarangapani, A comprehensive survey on model compression and acceleration, Artif. Intell. Rev. 53 (2020) 5113–5155.

[57] C. Yuan, S.S. Agaian, A comprehensive review of binary neural network, Artif. Intell. Rev. 56 (11) (2023) 12949–13013.

[58] J. Cheng, P.-s. Wang, G. Li, Q.-h. Hu, H.-q. Lu, Recent advances in efficient computation of deep convolutional neural networks, Front. Inf. Technol. Electron. Eng. 19 (2018) 64–77.

[59] E. Frantar, S. Ashkboos, T. Hoefler, D. Alistarh, GPTQ: Accurate Post-Training quantization for generative Pre-Trained transformers, in: International Conference on Learning Representations, ICLR, 2023.

[60] J. Lin, J. Tang, H. Tang, S. Yang, W.-M. Chen, W.-C. Wang, G. Xiao, X. Dang, C. Gan, S. Han, AWQ: Activation-Aware weight quantization for On-Device LLM compression and acceleration, in: Proceedings of Machine Learning and Systems, MLSys, 2024.

[61] M. Denil, B. Shakibi, L. Dinh, M. Ranzato, N. De Freitas, Predicting parameters in deep learning, Adv. Neural Inf. Process. Syst. 26 (2013).

[62] Y. He, X. Zhang, J. Sun, Channel pruning for accelerating very deep neural networks, in: Proceedings of the IEEE International Conference on Computer Vision, ICCV, 2017, pp. 1389–1397.

[63] Y. He, Y. Ding, P. Liu, L. Zhu, H. Zhang, Y. Yang, Learning filter pruning criteria for deep convolutional neural networks acceleration, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2020, pp. 2009–2018.

[64] G. Fang, X. Ma, M. Song, M.B. Mi, X. Wang, DepGraph: Towards any structural pruning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2023, pp. 16091–16101.

[65] Y. Zhang, T. Xiang, T.M. Hospedales, H. Lu, Deep mutual learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2018, pp. 4320–4328.

[66] S. Sun, Y. Cheng, Z. Gan, J. Liu, Patient knowledge distillation for BERT model compression, in: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), Association for Computational Linguistics, Hong Kong, China, 2019, pp. 4323–4332, http://dx.doi.org/10.18653/v1/D19-1441.

[67] X. Jiao, Y. Yin, L. Shang, X. Jiang, X. Chen, L. Li, F. Wang, Q. Liu, TinyBERT: Distilling BERT for natural language understanding, in: Findings of the Association for Computational Linguistics: EMNLP 2020, Association for Computational Linguistics, Online, 2020, pp. 4163–4174, http://dx.doi.org/10.18653/v1/2020.findings-emnlp.372.

[68] Y. Jang, H. Lee, S.J. Hwang, J. Shin, Learning what and where to transfer, in: Proceedings of the 36th International Conference on Machine Learning (ICML), in: Proceedings of Machine Learning Research, Vol. 97, PMLR, 2019.

[69] L. Bossard, M. Guillaumin, L. Van Gool, Food-101 – mining discriminative components with random forests, in: Computer Vision – ECCV 2014: 13th European Conference, Proceedings, Part VI, Springer, 2014, pp. 446–461.

[70] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, 2008, pp. 722–729.

[71] M. Cimpoi, S. Maji, I. Kokkinos, S. Mohamed, A. Vedaldi, Describing textures in the wild, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2014, pp. 3606–3613.

[72] O.M. Parkhi, A. Vedaldi, A. Zisserman, C.V. Jawahar, Cats and dogs, in: 2012 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, 2012, pp. 3498–3505.

[73] A. Krizhevsky, G. Hinton, Learning Multiple Layers of Features from Tiny Images, Tech. Rep. 0, University of Toronto, Toronto, Ontario, 2009, [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf.

[74] S.S. Chouhan, U.P. Singh, A. Kaul, S. 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, IEEE, 2019, pp. 700–707.

[75] Y. You, I. Gitman, B. Ginsburg, Large batch training of convolutional networks, 2017, arXiv preprint arXiv:1708.03888.

[76] X. Chen, H. Fan, R. Girshick, K. He, Improved baselines with momentum contrastive learning, 2020, arXiv preprint arXiv:2003.04297.

[77] NVIDIA Corporation, Power Optimization with NVIDIA Jetson, NVIDIA Technical Blog, 2023, URL https://developer.nvidia.com/blog/power-optimization-with-nvidia-jetson/.

[78] NVIDIA Corporation, Jetson Linux Developer Guide: Clock Frequency and Power Management, 2024.

[79] NVIDIA Corporation, Jetson Nano 2GB Developer Kit User Guide, NVIDIA Developer Documentation, 2021, URL https://developer.nvidia.com/embedded/learn/jetson-nano-2gb-devkit-user-guide.

[80] M. Borowski, S. Paszkiel, Power requirements evaluation of embedded devices for Real-Time video line detection, Energies 16 (18) (2023) 6677, http://dx.doi.org/10.3390/en16186677.

[81] S. Han, H. Mao, W.J. Dally, Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding, 2015, arXiv preprint arXiv:1510.00149.

[82] J. Xie, C. Ding, X. Li, S. Ren, Y. Li, Z. Lu, Nestquant: Post-training integer-nesting quantization for on-device dnn, IEEE Trans. Mob. Comput. (2025).

[83] I. Hubara, Y. Nahshan, Y. Hanani, R. Banner, D. Soudry, Accurate post training quantization with small calibration sets, in: International Conference on Machine Learning, PMLR, 2021, pp. 4466–4475.

[84] Z. Hao, J. Guo, K. Han, Y. Tang, H. Hu, Y. Wang, C. Xu, One-for-all: Bridge the gap between heterogeneous architectures in knowledge distillation, Adv. Neural Inf. Process. Syst. 36 (2023) 79570–79582.

[85] Y. Peng, H. Ye, C. Huang, X. Hu, J. Chen, R. Zeng, Revisiting Cross-Architecture distillation: Adaptive Dual-Teacher transfer for lightweight video models, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 40, 2026, pp. 8367–8375, no. 10.

[86] H. Touvron, M. Cord, M. Douze, F. Massa, A. Sablayrolles, H. Jégou, Training data-efficient image transformers & distillation through attention, in: International Conference on Machine Learning, PMLR, 2021, pp. 10347–10357.

[87] L. Du, X. Zhang, G. Lan, Resource-efficient gaze estimation via Frequency-domain Multi-task contrastive learning, ACM Trans. Sens. Netw. 21 (4) (2025) 1–24.

[88] Y. Liu, J. Cao, B. Li, W. Hu, J. Ding, L. Li, Cross-architecture knowledge distillation, in: Proceedings of the Asian Conference on Computer Vision, 2022, pp. 3396–3411.

[89] S. Luo, D. Chen, C. Wang, Knowledge distillation with deep supervision, in: 2023 International Joint Conference on Neural Networks, IJCNN, IEEE, 2023, pp. 1–8.