Exploring Transferability of Self-Supervised Learning by Task Conflict Calibration

Huijie Guo[1*], Jingyao Wang[1,2*], Peizheng Guo[1,2], Xingchen Shen[1], Changwen Zheng[1,2†], Wenwen Qiang[1,2,‡]

[1]Institute of Software Chinese Academy of Sciences,
[2]University of Chinese Academy of Sciences,
{guohuijie,wangjingyao2023,guopeizheng2025,xingchen,changwen,qiangwenwen}@iscas.ac.cn

Abstract

In this paper, we explore the transferability of SSL by addressing two central questions: (i) what is the representation transferability of SSL, and (ii) how can we effectively model this transferability? Transferability is defined as the ability of a representation learned from one task to support the objective of another. Inspired by the meta-learning paradigm, we construct multiple SSL tasks within each training batch to support explicitly modeling transferability. Based on empirical evidence and causal analysis, we find that although introducing task-level information improves transferability, it is still hindered by task conflict. To address this issue, we propose a Task Conflict Calibration (TC^2) method to alleviate the impact of task conflict. Specifically, it first splits batches to create multiple SSL tasks, infusing task-level information. Next, it uses a factor extraction network to produce causal generative factors for all tasks and a weight extraction network to assign dedicated weights to each sample, employing data reconstruction, orthogonality, and sparsity to ensure effectiveness. Finally, TC^2 calibrates sample representations during SSL training and integrates into the pipeline via a two-stage bi-level optimization framework to boost the transferability of learned representations. Experimental results on multiple downstream tasks demonstrate that our method consistently improves the transferability of SSL models.

Code — https://github.com/PaulGHJ/TC2

Introduction

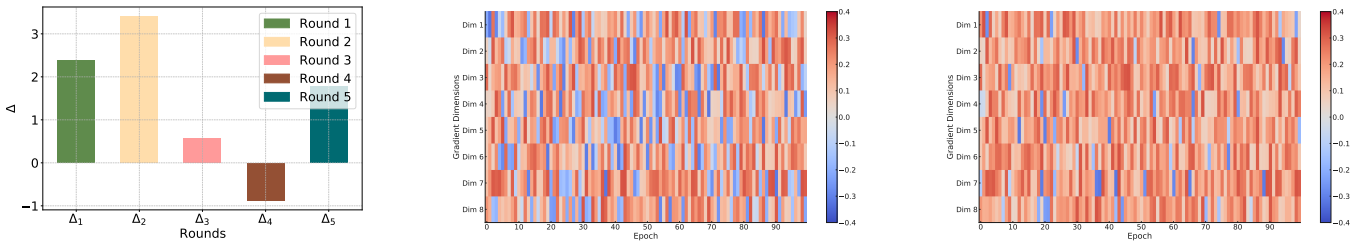
Traditional unsupervised learning lacks ground-truth labels and often performs worse than supervised learning. However, supervised methods heavily rely on costly manual annotations and still face generalization issues (Bousquet and Elisseeff 2002; Mignacco et al. 2020). As a compelling alternative, Self-Supervised Learning (SSL) has shown strong performance in image classification and object recognition (Schiappa, Rawat, and Shah 2023; Sun et al. 2025; Liu et al. 2022b), sometimes even surpassing supervised methods.

Existing self-supervised learning (SSL) methods primarily focus on designing reliable supervisory signals or introducing inductive biases to guide the model in learning effective feature representations. For example, discriminative SSL (D-SSL) methods construct positive pairs by generating different augmented views of the same sample and treat other samples as negatives (Chen et al. 2020a; He et al. 2020; Caron et al. 2020). Generative SSL (G-SSL) methods mask part of the input and train the model to reconstruct the masked regions, using the reconstruction error as the supervisory signal (He et al. 2022; Bao et al. 2021). However, these methods often overlook a critical question: how do such learning paradigms ensure the transferability of learned features? This question is not easy to answer and helps explain why these methods often perform poorly on out-of-distribution (OOD) data or transfer tasks. Additionally, this limits the theoretical insights and methodological advances needed to improve their transfer generalization capabilities.

In this paper, we focus on two key questions: (1) What is representation transferability? (2) How can it be effectively modeled? For the first question, we take an intuitive perspective: if a representation has good transferability, it should perform well across various downstream tasks. Based on this intuition, we define task-level transferability as the ability of a representation learned from one task to support the objective of another (Yosinski et al. 2014). To address the second question, we consider meta-learning as an established approach to modeling transferability. Prior works (Ni et al. 2021) suggest that SSL can be regarded as a special case of meta-learning. The main difference lies in task organization within each mini-batch and learning mechanism. Meta-learning typically involves multiple tasks, whereas SSL is generally considered to involve only a single task. Moreover, meta-learning is optimized in a bi-level manner. It achieves strong transferability through multi-task bi-level training, which implicitly models a distribution over tasks. This allows meta-learning to generalize well to unseen tasks. Motivated by this, we propose enhancing the transferability of SSL by introducing a meta-learning mechanism. Specifically, we construct multiple tasks within each mini-batch during training and train SSL models in a bi-level manner.

We further explore the effectiveness of the above meta-learning-like SSL method on transferability. In the experi-

*These authors contributed equally.
†Corresponding Author.
‡CO-Corresponding Author.



(a) (b) (c)

Figure 1: Effect of task-level information on CIFAR-100. (a) shows the effect of SimCLR trained on ImageNet and tested on CIFAR-100 before and after introducing task-level information in five runs. (b) and (c) show the training process corresponding to the worst and best rounds in five evaluations, visualizing the gradient similarity between tasks.

ment, we uniformly split each training batch into two subsets, which serve as two independent SSL tasks. Following the standard SSL training framework, each subset generates two augmented views via random data augmentation for training. We use SimCLR as the baseline, training on ImageNet-100 and testing on other datasets. First, we evaluate the Top-1 accuracy without task construction. Next, we assess performance with the multi-task learning mechanism through five independent training rounds and calculate the improvement for each round. From Figure 1a, the results demonstrate that incorporating task-level information significantly improves model performance, but the improvement is not consistent, with a nearly 3% performance gap between the best and worst outcomes, and some cases even showing negative gains. Further analysis reveals that gradient conflict between tasks during training significantly affects transferability, as shown in Figure 1b and 1c. Therefore, when introducing task-level information enhances transferability, inter-task gradient conflict remains an important issue.

To better understand the underlying causes of task conflict and its impact on transferability, we analyze it through both data generation and causal inference. Specifically, we construct a Structural Causal Model (SCM) to illustrate these relationships (as shown in Figure 2a), where Y_i, Y_j are task labels, $F^s_{i,j}$ is the shared generative factors, and F^u_i, F^u_j are task-specific factors. In meta-learning-like SSL, the model estimates generative factors from multiple tasks and predicts labels accordingly. Due to the shared training across tasks, the extracted factors inevitably mix semantics from all tasks(Figure 2b). As shown in Theorem 1, if task-specific factors are not correctly identified, these factors from other tasks act as confounders, hindering target task learning—this phenomenon is termed task conflict. It degrades per-task learning and limits the effectiveness of modeling task distributions, explaining the inconsistent transfer performance observed in meta-learning-like SSL. Moreover, these generative factors can be seen as semantic vectors composing the feature space. Since different tasks may involve distinct semantics, task conflict implies that target features may be contaminated by irrelevant semantics. Thus, enhancing SSL transferability requires calibrating each sample's feature representation to retain only task-relevant semantics within the meta-learning-like framework.

Based on the above analysis, we propose a new method, called Task Conflict Calibration (TC^2), to alleviate task conflict when explicitly modeling the transferability for SSL methods. First, TC^2 constructs multiple SSL tasks through batch splitting, which introduces task-level information into the model. Second, TC^2 learns a factor extraction network f_v to generate causal generative factors for all tasks and a weight extraction network f_w to generate a weight of a single sample of each task. We leverage techniques such as data reconstruction, orthogonality, and sparsity to ensure that the learned f_v can effectively generate causal generative factors suitable for multiple tasks, while the learned function f_w produces a weight matrix of semantic vectors that are specifically relevant to the target task. Third, based on f_v and f_w, we formulate the final version of TC^2, which serves to calibrate sample representations during the SSL training process. Finally, we embed TC^2 into the SSL training pipeline through a two-stage bi-level optimization framework to enhance the transferability of learned representations. Extensive transfer learning and ablation experiments demonstrate the effectiveness of TC^2. Our contributions include:

- We find that simply formulating the SSL training process as a meta-learning paradigm does not guarantee improved feature transferability. Through causal analysis, we demonstrate this limitation stems from task conflict.
- We propose a method to alleviate task conflict and improve the transferability of SSL models, leveraging a two-stage bi-level optimization mechanism.
- Experimental results on multiple downstream tasks demonstrate the effectiveness of our method while preserving the model's inherent representation capabilities.

Related Work

Self-Supervised Learning (SSL) has demonstrated remarkable performance in multiple fields, even rivaling supervised learning. Early SSL methods focused on designing pretexts (Albelwi 2022; Doersch, Gupta, and Efros 2015) to provide supervisory information for the unlabeled data. With the rise of contrastive learning, SSL gradually turns to learning more generalizable representations based on invariance to data augmentation. Typical contrastive learning constructs positive sample pairs through data augmentation so that positive pairs are close and negative pairs are separated in the embedding space. SimCLR (Chen et al.

2020a) introduces a simple yet effective contrastive learning framework, where the model is trained to maximize the consistency between positive pairs and minimize the consistency between negative pairs through the InfoNCE loss (Gutmann and Hyvärinen 2010). To reduce the computational cost caused by batch size, MoCo (He et al. 2020) uses a memory bank to increase the number of negative samples and a momentum encoder to learn representations. SwAV (Caron et al. 2020) enforces consistency between cluster assignments from different augmented views of the same image, rather than computing comparisons of sample pairs. BYOL (Grill et al. 2020) and SimSiam (Chen and He 2021) only constrain the consistency between positive samples to avoid trivial solutions through the online and target networks. Additionally, some studies aim to reduce redundancy among features(Liu et al. 2022a; Bardes, Ponce, and Lecun 2022). Barlow Twins (Zbontar et al. 2021) endeavors to make the normalized cross-correlation matrix of the augmented embeddings close to the identity matrix.

Transferability refers to the ability of a model trained on one task to be transferred to another (Pan and Yang 2009; Sun et al. 2024b; Zang et al. 2025). The interplay between out-of-distribution and transfer performance of convolutional neural networks is investigated by (Djolonga et al. 2020). According to (Ying et al. 2018), achieving optimal transferability often requires exhaustive search or the use of diverse task distributions. Multi-task learning (Standley et al. 2020) makes the model more adaptable by sharing knowledge among multiple related tasks, thereby improving its transferability to new ones. The connection between the structure of vision tasks and transferability is highlighted by (Zamir et al. 2018), which explores which tasks can be effectively transferred to any target task. To enable fast adaptation to new tasks with limited labels, meta-learning leverages many small-sample tasks to train a base learner (Hospedales et al. 2021; Sun et al. 2021). The connection between SSL and meta-learning to improve the transferability of self-supervised models is explored by some work(Ni et al. 2021).

Existing SSL methods construct models based on samples, which can be treated as an in-distribution problem for a specific task, without explicitly modeling transferability from the task level. In this paper, we explicitly model transferability for SSL by constructing multiple tasks and improving the transferability by resolving task conflict issues.

Problem Analysis and Motivation

This section begins by revisiting the self-supervised learning paradigm, highlighting the implicit task structure underlying it. We then explore the inherent connection between self-supervised learning and meta-learning through the lens of task modeling. Building on this, we empirically investigate how introducing task-level information can lead to task conflicts, which hinder the transferability of learned representations. Finally, we analyze the root causes of these conflicts from a causal perspective.

Rethinking SSL from a Task Perspective

In the training phase, data is organized into mini-batches $\mathcal{X}_{tr} = \{x_i\}_{i=1}^{N}$. In D-SSL methods such as SimCLR (Chen et al. 2020a) and Barlow Twins (Zbontar et al. 2021), each sample is stochastically augmented into two views (x_i^1, x_i^2). In G-SSL methods like MAE (He et al. 2022), x_i is divided into patches, masked, and reassembled into x_i^1, while the original sample serves as x_i^2. Thus, the augmented training set becomes $\mathcal{X}_{tr}^{aug} = \{(x_i^1, x_i^2)\}_{i=1}^{N}$, and SSL aims to learn a feature extractor f from these pairs.

D-SSL objectives typically consist of alignment (maximizing similarity within pairs) and regularization (e.g., enforcing uniformity (Wang and Isola 2020)). G-SSL achieves alignment by reconstructing x_i^2 from x_i^1 using an encoder-decoder structure. In both cases, one sample in the pair serves as an *anchor*, guiding the other to align with it. For D-SSL, this is explicit; for G-SSL, x_i^2 acts as the anchor for reconstructing x_i^1. This process can be interpreted as assigning one sample as a learning target and adjusting the other to match it in feature space. As a result, paired samples become tightly clustered, with the anchor acting like a cluster center. Consequently, $\mathcal{X}_{tr}^{aug}$ can be viewed as a mini-batch classification task with N implicit classes, where alignment functions similarly to class-specific feature learning.

Comparsion of SSL and Meta-Learning

During the meta-training phase of meta-learning, the model is trained on a collection of tasks sampled from a task distribution. Each training mini-batch consists of multiple tasks $\{\tau_1, \tau_2, \cdots, \tau_B\}$, where each task τ_i contains a support set S_i and a query set Q_i. The support set is used to adapt the model to the specific task, while the query set is used to evaluate and update the meta-learner's parameters based on the adaptation performance. The learning mechanism of meta-learning typically follows a bi-level optimization paradigm (Gordon et al. 2018; Jiang et al. 2022): the inner loop adapts task-specific parameters using the support set, and the outer loop updates the shared meta-parameters using the query set loss across all tasks in the mini-batch. This enables the model to acquire generalizable knowledge that can be quickly adapted to unseen tasks during meta-testing.

Comparing SSL and meta-learning, we can obtain SSL and meta-learning share certain similarities in task construction but differ significantly in mini-batch construction and learning mechanism (Ni et al. 2021). Based on the above analysis, we can obtain the collection of all instance pairs across the entire training set can be viewed as forming a multi-class classification task. Similar to traditional machine learning, the training process of SSL can be viewed as a mini-batch-based learning procedure. The basic unit of its training set is a sample pair, and multiple distinct pairs form a mini-batch. The training process essentially aims to approximate the conditional distribution $p(y|x)$. Notably, the learned distribution $p(y|x_{\text{train}})$ can generalize to $p(y|x_{\text{test}})$ only under the assumption that training and test data are i.i.d.. However, in OOD or transfer tasks, this i.i.d. assumption often fails, which poses a challenge to the performance of SSL in such scenarios. In contrast, meta-learning adopts a different structural unit: each basic unit in the training set corresponds to an entire task, and a mini-batch consists of multiple such tasks. During meta-training, the model learns to approximate the same $p(y|x)$ across all samples within

any given task. Since both training and test tasks are assumed to be sampled from the same (possibly unknown) task distribution, the i.i.d. assumption naturally holds at the task level. As a result, the function learned during meta-training can be effectively transferred to test tasks, making meta-learning more robust and reliable when facing OOD problem or transfer learning scenarios.

Empirical Evidence

One promising way to improve the transferability of SSL representations is to incorporate task-level information inspired by meta-learning. Specifically, we uniformly split each training batch $\mathcal{X}_{tr}^{aug}$ into two subsets B_1 and B_2, which serve as two distinct tasks. Each task is further divided into a support set B_i^s and query set B_i^q, where $i \in \{1, 2\}$, following the meta-learning framework. We adopt SimCLR as the baseline, training on ImageNet-100 and testing on multiple datasets. Here we just report results on CIFAR-100(Details are provided in the Appendix). First, we evaluate the Top-1 accuracy without task construction, denoted as $acc(\mathrm{SimCLR})$. Then, using the task-based multi-task learning mechanism, we conduct five independent runs, with the i-th result denoted as $acc(\mathrm{SimCLR} + \mathrm{T}, i)$. The corresponding performance gain is $\Delta_i = acc(\mathrm{SimCLR} + \mathrm{T}, i) - acc(\mathrm{SimCLR})$. Results in Figure 1a show that task-level design improves performance, but with noticeable instability—performance differences can reach nearly 3%, and negative gains are observed in some runs.

To investigate this instability, we analyze the first 100 training epochs of the best and worst performing runs. We compute the cosine similarity of gradients between tasks at each epoch. A negative value indicates conflicting gradients (angle $> 90°$), while a positive value indicates alignment. As shown in Figure 1b and Figure 1c, we observe that: (i) gradient similarity fluctuates during training, often exhibiting conflict; (ii) lower inter-task gradient conflict correlates with better transfer performance. These results suggest that while task-level information enhances SSL transferability, it introduces inter-task gradient conflict, which can negatively affect consistency and stability in performance.

Causal Analysis and Motivation

Causal Analysis. To investigate the underlying reasons for the above phenomenon, we propose using causal theory for analysis. We construct a Structural Causal Model (SCM) based on the causal generating mechanism (Suter et al. 2019; Hu et al. 2022), as shown in Figure 2(a). Specifically, the SCM comprises two tasks, τ_i and τ_j, where Y_i and Y_j denote the label variables for tasks τ_i and τ_j, and X_i and X_j represent the generated samples corresponding to these tasks. Meanwhile, F_i^u and F_j^u denote the sets of unique causal factors specific to tasks τ_i and τ_j, respectively, and $F_{i,j}^s$ contains the shared causal generative factors. Following (Sun et al. 2024a; Deshpande et al. 2022), the samples are generated with all the causal generative factors, e.g., for τ_i, the sample X_i is generated by both the task-specific factors F_i^u and the shared factors $F_{i,j}^s$. Since each causal generative factor in F_i^u, F_j^u, and $F_{i,j}^s$ corresponds to a specific semantic attribute (e.g., color, shape) in its respective task, they can be viewed as being determined by the label variable Y_i (or Y_j) of the tasks. Thus, we obtain Figure 2a, and for task τ_i, we refer to $F_{i,j}^s$ and F_i^u as the causal generating factors that are causally related to Y_i, while F_j^u is considered non-causal factors.

(a) (b)

Figure 2: (a) Overview of the SCM based on generation mechanisms; (b) The causal mechanism of task conflict.

Based on the above SCM, a good SSL model should utilize only the causal factors to learn the tasks, thereby achieving accurate decisions even in the absence of labels. However, to achieve modeling transferability, a multi-task joint learning mechanism is required, which causes the model to learn all the causal factors across tasks, including F_i^u, F_j^u, and F_{ij}^s. Consequently, for a specific task τ_i, the model may learn from non-causal factors originating from other tasks, which poses a significant challenge to attaining optimal predictions. To validate this assertion, we consider a scenario involving two tasks in the same SSL training batch. Here, we treat the pseudo-labels generated by data augmentation as Y_i and Y_j. Then, we have:

Theorem 1 *If the correlation between Y_i and Y_j is not equal to 0.5, then the optimal model for task i has a non-zero weight on F_j^u. If the correlation is equal to 0.5 with limited training samples, then the optimal classifier for task i also has non-zero weight on factor F_j^u.*

It shows that the learned SSL model leverages causal factors from other tasks to facilitate the learning of the target task (with proofs in Appendix). For example, in task τ_i, the model employs causal factors F_j^u from task τ_j to learn Y_i, creating a spurious correlation represented as $F_j^u \rightarrow Y_i$; similarly, $F_i^u \rightarrow Y_j$ exists for task τ_j. Therefore, the SCM for the learning phase, shown in Figure 2b, which contains two spurious paths. These spurious correlations lead to suboptimal learning for each task. For any target task, interference from other tasks during training disrupts learning. A direct indicator of this is when the gradients between tasks have angles greater than $90°$ (Figure 1), meaning that updates from other tasks suppress the gradient updates of the target task—a phenomenon we refer to as "task conflict".

Motivation. Based on the above causal analysis, enhancing SSL feature transferability requires not only constraining the learning mechanism as in meta-learning but also addressing task conflict during training. Generative factors can be viewed as semantic vectors (Pearl 2009; Wolff and Zettergren 2019; Schölkopf et al. 2021), and sample features as projections onto them. Task conflict arises when the multi-task learning process forces the model to learn all task-related semantic vectors. As a result, a target task's feature

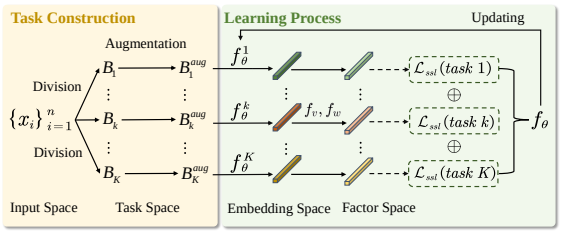


Figure 3: Overview of the proposed framework.

representation may contain non-zero projections onto irrelevant semantics from other tasks, degrading its effectiveness. To resolve this, we propose identifying the semantic vectors relevant to the target task and retaining only the corresponding feature projections. This approach effectively eliminates the interference of unrelated semantics, thereby mitigating task conflict and improving feature transferability.

Methodology

Task Construction

In this subsection, we describe how to construct SSL tasks, as shown in Figure 3. We begin by randomly sampling a mini-batch $X_{tr} = \{x_i\}_{i=1}^{N}$ from the training set.

Drawing inspiration from meta-learning (Vilalta and Drissi 2002; Finn, Abbeel, and Levine 2017), we divide the mini-batch X_{tr} into K groups, denoted as $B_1, ..., B_k, ...B_K$, with each group treated as an individual task. This grouping strategy allows the model to simulate diverse tasks within a single batch, promoting the learning of task-agnostic representations. Such representations are crucial for improving the model's transferability to unseen domains or tasks. For D-SSL, we randomly apply m data augmentation strategies to each group to generate the augmented dataset B_k^{aug}, containing m augmented views for each original sample. Each task consists of a support set and a query set. The support set is constructed by randomly selecting m' augmented views from each sample, while the remaining views are assigned to the query set. The support and query sets are then used to update the task-specific model parameters and guide the learning process for each task. For G-SSL, each sample is first divided into multiple patches. Then, m different random masks are applied to produce m masked views. Each task consists of a support set and a query set. The support set is constructed by randomly selecting m' masked views for each sample, along with the original (unmasked) sample. The remaining masked views are used to construct the query set. The model is trained to reconstruct the original input from the masked views by minimizing the reconstruction loss. By constructing multiple tasks, the SSL model can learn both the sample-level and task-level information, improving its transferability. However, the inevitability of task conflicts may affect the transferability of the model (Figure 1), and we will address this issue in the next subsection.

Task Conflict Calibration

In this subsection, we propose a method for calibrating task conflict. Specifically, we address the following three key questions: (i) how to learn the union of causal generative factors across all tasks; (ii) how to identify the task-specific causal factors for individual tasks; and (iii) how to alleviate the conflicts that may arise between tasks.

For question (i), we first compute the mean vector and covariance matrix of all sample representations in the training batch. The mean vector represents the center of the sample features, while the covariance matrix captures the relationships between the features. We then concatenate these two statistical quantities to form a new matrix $S \in R^{(d+1)\times d}$, for the subsequent factor extraction process. Next, we fed the concatenated matrix into a factor extraction network f_v. The output of f_v is constrained to be a factor matrix $V = f_v(S) \in \mathbb{R}^{d_v \times n_f}$, where each column represents the basis of a factor, d_v is the dimension of the generative factor (which is the same as the dimension of the sample representation, denoted as d), and n_f is the number of factors. At last, the loss function of learning f_v can be presented as:

$$\mathcal{L}_v(f_v) = \left\|V^{\mathrm{T}}V - I\right\|_F^2 + \left\|Z_{tr}^{aug} - Z_{tr}^{aug}VV^{\mathrm{T}}\right\|_F^2. \quad (1)$$

where Z_{tr}^{aug} represent the augmented dateset on the whole training batch, I is a identity matrix, $\|\cdot\|_F$ is the F-paradigm of the matrix. The first term is to constrain the orthogonality of the column vectors of V, and the second term is to constrain V to best reconstruct the training dataset.

Since the SSL task in this work is constructed by partitioning the entire training batch, and the full set encompasses all tasks, it is reasonable to base the learning on the entire training batch. This approach allows the model to capture the global characteristics of the data, facilitating generalization and transfer across tasks. Specifically, the mean vector and covariance matrix serve as sufficient statistics, summarizing the main features of the data and reflecting its overall distribution. By using these statistics to extract factors, the model learns shared representations that can adapt to different tasks, thereby improving transferability. As different factors represent distinct semantics, enforcing orthogonality among their corresponding vectors serves as a feasible approach to learning f_v. Minimizing the reconstruction error ensures that the factor matrix accurately reconstructs the data, allowing the shared factors across tasks to be effective and adaptable, further supporting transfer learning.

For question (ii), given a sample of a task, we input it into a leanrnable weight extraction network f_w. We constrain that the output of f_w is a weight vector with n_f dimension. Then, the loss function of learning f_w is presented:

$$\mathcal{L}_w(f_w) = \sum_k \sum_{z_j \in \text{task } k} \left\|z_j - W_j V^T\right\|_2^2 + \|W_j\|_1. \quad (2)$$

Here, V is a pre-given matrix, $W_j = f_w(z_j)V^T$ represents the weight vector of the j-th sample under task k, z_j is the representation of a sample. The first term ensures that each sample representation z_j is reconstructed by a weighted combination of a few shared factors in V, with W_j serving as its sample-specific weight vector. This encourages the model to capture task-specific structures through selective factor usage. The second term applies an l_1-norm regularization to W_j, promoting sparsity by activating only a limited number of factors. This improves interpretability and

reduces overfitting.Together, these terms guide f_w to assign sparse, task-aware weights over shared factors, enabling the identification of task-specific causal components.

The essence of task conflict lies in the causal generative factors of the target task being confounded by factors from other tasks. A feasible solution is to constrain the features of samples from the target task to contain only semantics that are relevant to the target task itself. From the perspective of feature representation, this can be interpreted as enforcing that the features of a sample are generated solely by the causal generative factors associated with the target task. Therefore, Bases on the solutions to question (i) and (ii), the solution to question (iii) for any sample z_j in the target task can be directly formulated as: $f_w(z_j)V^T$. Finally, we get the task conflict calibration loss by combining Eq.1 and Eq.2:

$$\mathcal{L}_{tc^2}(f_v, f_w) = \mathcal{L}_v + \lambda_s \mathcal{L}_w, \tag{3}$$

where λ_s is the parameter used for balancing two terms.

Learning Process

The training process with TC^2 in each batch is performed via two stages. In this first stage, we fix the SSL model and optimize the transfer function f_v and weight function f_w via bi-level optimization. In the second stage, we optimize the SSL model with f_v and f_w being held.

In the first stage, we optimize f_v and f_w to ensure the causality of the learned generative factors. Following (Zhu, An, and Huang 2021; Deng, Gould, and Zheng 2022), generative factors should remain invariant across different sample distributions within a task. An effective model captures these factors and adapts across tasks. Support and query sets from the same task represent shifted distributions, and the meta-learning objective is to learn invariants effective across both.

Thus, we adopt bi-level optimization to learn f_v and f_w, enforcing causal invariance. First, we first update f_v and f_w on the support sets of all tasks by the following formula:

$$\begin{aligned} &f'_v \leftarrow f_v - \alpha_1 \nabla_{f_v} \bar{\mathcal{L}}^s, \quad f'_w \leftarrow f_w - \alpha_2 \nabla_{f_w} \bar{\mathcal{L}}^s \\ &s.t. \bar{\mathcal{L}}^s = \tfrac{1}{K} \sum_{k=1}^{K} \mathcal{L}^s_{ssl}(f_w, f_v, B_k^{aug}) + \mathcal{L}_{tc^2}(f_v, f_w) \end{aligned} \tag{4}$$

Here, $\mathcal{L}^s_{ssl}(\cdot)$ is the self-supervised loss on support set S_k of task k, and α_1, α_2 are learning rates. Note that losses are computed in the factor space.

Next, f_v and f_w are updated on query sets:

$$\begin{aligned} &f_v \leftarrow f'_v - \alpha_3 \nabla_{f'_v} \bar{\mathcal{L}}^q, \quad f_w \leftarrow f'_w - \alpha_4 \nabla_{f'_w} \bar{\mathcal{L}}^q \\ &s.t. \bar{\mathcal{L}}^q = \tfrac{1}{K} \sum_{k=1}^{K} \mathcal{L}^q_{sse}(f'_w, f'_v, B_k^{aug}) + \mathcal{L}_{tc^2}(f'_v, f'_w) \end{aligned} \tag{5}$$

$\mathcal{L}^q_{sse}(\cdot)$ denotes the sum of squared errors on query set Q_k, with learning rates α_3, α_4.

In the second stage, with f_v and f_w fixed, we update f_θ via bi-level optimization to explicitly encode transferability. We first fine-tune f_θ for each task using its support set:

$$f^k_\theta \leftarrow f_\theta - \beta_1 \nabla_{f_\theta} \mathcal{L}^s_{ssl}(B_k^{aug}, f_\theta), \tag{6}$$

where β_1 is the learning rate. Then, we update f_θ by minimizing the total query loss across tasks:

$$f_\theta \leftarrow f_\theta - \beta_2 \nabla_{f_\theta} \sum_{k=1}^{K} \mathcal{L}^q_{sse}(B_k^{aug}, f^k_\theta), \tag{7}$$

with β_2 as the learning rate.

This two-stage bi-level optimization mitigates task conflicts that hinder learning, and enables multi-task modeling in SSL to enhance representation transferability.

Experiments

In this section, we conduct extensive experiments on benchmark datasets to evaluate the effectiveness of the proposed TC^2. More details and results are provided in the Appendix.

Transfer Learning

In this section, we study the transferability of our proposed method under different transfer tasks, covering 7 benchmark datasets. Note that we also validate the effectiveness of our proposed method for unsupervised learning, semi-supervised learning and generative tasks in the Appendix.

Video-based Task. We evaluate TC^2 on video tasks by transferring the pre-trained model to the UniTrack benchmark (Wang et al. 2021). Table 1 reports the results on five tasks using features from [layer3/layer4] of ResNet-50. The results show that our proposed method has improvements compared with existing self-supervised methods. SimCLR increase about 3% in the VOS (Perazzi et al. 2016), and BYOL increase over 2% in the MOT (Milan et al. 2016).

Object Detection and Instance Segmentation. We evaluate our method on VOC 07 (Everingham et al. 2010) and COCO (Lin et al. 2014) for object detection and instance segmentation tasks with the most commonly used protocol (Zbontar et al. 2021). Several different self-supervised methods are used for comparison. We report the comparison results before and after introducing our proposed method in Table 2, showing that TC^2 brings stable performance improvements on different transfer tasks. For example, BYOL+TC^2 achieves SOTA on the VOC 07 detection task, and VICReg+TC^2 achieves SOTA on the COCO instance segmentation task.

OOD Tasks. We also assess our method on benchmark datasets that address the OOD challenge, i.e., PACS (Xu, Xiao, and López 2019), ColoredMNIST (Nam et al. 2020), and OfficeHome (Venkateswara et al. 2017). Specifically, we assess the performance of SSL baselines both before and after incorporating the proposed TC^2 across three datasets. For PACS, we train the SSL models on these domains (Photo, Sketch, and Cartoon) and test them on all four domains (Photo, Art, Cartoon, and Sketch), with the average performance also reported. For ColoredMNIST, we adopt the experimental setup from (Gat et al. 2020), evaluating the ability to generalize to new classes after training on base classes. Lastly, for OfficeHome, we randomly select one domain as the source for training and another as the target domain. Labels for the source domain are predefined, whereas those for the target domain remain unknown. We then compare the performance of SimCLR with and without the introduction of TC^2. As shown in Table 3, Table 4 and Table 7, TC^2 significantly enhances performance, thereby confirming its effectiveness on OOD tasks.

Method	SOT		VOS	MOT		MOTS		PoseTrack
	AUC_{XCorr}	AUC_{DCF}	J-mean	IDF1	HOTA	IDF1	HOTA	IDF1
SimCLR	47.3 / 51.9	61.3 / 50.7	60.5 / 56.5	66.9 / 75.6	57.7 / 63.2	65.8 / 67.6	67.7 / 69.5	72.3 / 73.5
MoCo	50.9 / 47.9	62.2 / 53.7	61.5 / 57.9	69.2 / 74.1	59.4 / 61.9	70.6 / 69.3	71.6 / 70.9	72.8 / 73.9
SwAV	49.2 / 52.4	61.5 / 59.4	59.4 / 57.0	65.6 / 74.4	56.9 / 62.3	68.8 / 67.0	69.9 /69.5	72.7 / 73.6
BYOL	48.3 / 55.5	58.9 / 56.8	58.8 / 54.3	65.3 / 74.9	56.8 / 62.9	70.1 / 66.8	70.8 / 69.3	72.4 / 73.8
Barlow Twins	44.5 / 55.5	60.5 / **60.1**	61.7 / 57.8	63.7 / 74.5	55.4 / 62.4	68.7 / 67.4	69.5 / 69.8	72.3 / 74.3
SimCLR+TC^2	51.3 / 54.2	**63.8** / 54.2	**63.4 / 59.3**	**70.8 / 78.3**	**62.9 / 64.5**	67.4 / **70.5**	68.9 / 71.1	73.2 / 74.3
BYOL+TC^2	**51.6 / 58.1**	59.9 / 58.9	61.7 / 56.9	67.3 / 76.6	57.9 / 63.8	**72.8** / 68.1	**73.8 / 72.3**	**75.6 / 76.2**

Table 1: Transfer learning on video tracking tasks. All methods use the ResNet-50 backbone with the best results in bold.

Method	VOC 07 det			COCO det			COCO seg		
	AP_{50}	AP	AP_{75}	AP_{50}	AP	AP_{75}	AP^m_{50}	AP^m	AP^m_{75}
Supervised	74.4	42.4	42.7	58.2	38.2	41.2	54.7	33.3	35.2
SimCLR	75.9	46.8	50.1	57.7	37.9	40.9	54.6	33.3	35.3
MoCo	77.1	46.8	52.5	58.9	39.3	42.5	55.8	34.4	36.5
BYOL	77.1	47.0	49.9	57.8	37.9	40.9	54.3	33.2	35.0
MAE	77.6	48.5	51.2	59.8	38.2	41.6	56.7	33.9	37.1
SimSiam	77.3	48.5	52.5	59.3	39.2	42.1	56.0	34.4	36.7
Barlow Twins	75.7	47.2	50.3	59.0	39.2	42.5	56.0	34.3	36.5
SwAV	75.5	46.5	49.6	58.6	38.4	41.3	55.2	33.8	35.9
VICRegL	75.9	47.4	52.3	59.2	39.8	42.1	56.5	35.1	36.8
SimCLR+TC^2	78.3	49.3	52.3	59.6	40.7	43.7	56.8	36.5	37.2
BYOL+TC^2	79.7	**50.7**	52.5	59.7	39.9	44.1	56.3	35.3	38.3
MAE+TC^2	**81.2**	50.6	53.4	62.3	40.5	44.7	**59.2**	35.1	**40.0**
VICRegL+TC^2	78.2	49.5	**54.1**	**62.7**	**42.6**	**45.3**	58.9	**37.9**	39.7

Table 2: Transfer learning on objection detection and instance segmentation. "AP" is the average precision, "AP_N" represents the average precision when the IoU (Intersection and Union Ratio) threshold is $N\%$.

Ablation Studies

In this subsection, we conduct ablation studies to analyze how our proposed method perform well.

Loss function. Our loss function contains two important components: $\mathcal{L}_v$, $\mathcal{L}_w$. We reduce the correlation between generative factors by minimizing $\mathcal{L}_v$ so that they can represent different semantic information. We also enforce correspondence between tasks and their generative factors by minimizing $\mathcal{L}_w$, while constraining the sparsity and diversity of the factors. Experimental results demonstrate the necessity of each component, as showned in Table 12.

Bi-level mechanism. Here, we conduct experiments to explore the impact of bi-level mechanism, as shown in Table 12. Experimental results show that the bi-level optimization mechanism indeed improves the performance of the model, demonstrating its effectiveness.

The number of tasks n_t. We divide each batch equally into multiple parts to construct SSL tasks. Here, we conduct experiments to analyze the impact of the number of tasks n_t on the experimental results on the CIFAR10 dataset. In the experiment, we set the number of tasks to 2, 4, 8, 16, and 32. Table 13 shows the effect of the number of tasks on the experiments. The experimental results show that $n_t = 4$ achieves the best results, also our settings.

Method	Photo	Sketch	Cartoon	Art	Average
SimCLR	86.4	85.1	87.2	74.3	83.3
SimCLR+TC^2	88.8	89.2	88.5	78.3	86.2
BYOL	83.9	84.6	82.7	64.5	78.9
BYOL+TC^2	85.6	87.2	85.0	69.9	81.9
Barlow Twins	83.1	82.7	83.4	64.8	78.5
Barlow Twins+TC^2	86.5	84.6	86.7	67.4	81.3
MAE	86.9	87.2	87.6	75.4	84.3
MAE+TC^2	89.1	90.0	89.2	78.9	86.8
SwAV	87.2	85.2	87.6	74.9	83.7
SwAV+TC^2	89.9	88.6	89.1	78.2	86.5
VICRegL	88.0	86.1	87.9	75.2	84.3
VICRegL+TC^2	91.5	89.3	90.2	79.6	87.7

Table 3: Transfer learning on PACS dataset.

Method	Average Accuracy(%)	Worst Accuracy (%)
SimCLR	85.2	82.4
SimCLR+TC^2	88.9	87.0
BYOL	85.9	83.1
BYOL+TC^2	89.4	87.8
MAE	86.8	84.0
MAE+TC^2	90.1	88.3

Table 4: Transfer learning on ColoredMNIST dataset.

Conclusion

This paper aims to explore how to improve the transferability of SSL. Through both theoretical analysis and empirical studies, we find that incorporating task-level information into SSL improves its transferability. However, task conflict introduces instability during the training process, limiting the transferability. To address this issue, we propose TC^2 to alleviate the negative impact of task conflict in self-supervised learning. TC^2 incorporates task-level information, calibrates task–factor correspondence through dedicated extraction functions, and optimizes the model using a two-stage bi-level optimization mechanism. Experimental results on various downstream tasks confirm the effectiveness of TC^2 in improving the transferability of SSL.

Acknowledgements

The authors would like to thank the anonymous reviewers for their valuable comments. This work was supported by the National Natural Science Foundation of China under Grants 62506355 and 42506186, and the numerical calculations in this study were partially performed on the ORISE Supercomputer (DFZX202416).

References

Albelwi, S. 2022. Survey on self-supervised learning: auxiliary pretext tasks and contrastive learning methods in imaging. *Entropy*, 24(4): 551.

Andriluka, M.; Iqbal, U.; Insafutdinov, E.; Pishchulin, L.; Milan, A.; Gall, J.; and Schiele, B. 2018. Posetrack: A benchmark for human pose estimation and tracking. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5167–5176.

Bao, H.; Dong, L.; Piao, S.; and Wei, F. 2021. BEiT: BERT Pre-Training of Image Transformers. In *International Conference on Learning Representations*.

Bardes, A.; Ponce, J.; and Lecun, Y. 2022. VICReg: Variance-Invariance-Covariance Regularization For Self-Supervised Learning. In *ICLR 2022-International Conference on Learning Representations*.

Bardes, A.; Ponce, J.; and LeCun, Y. 2022. Vicregl: Self-supervised learning of local visual features. *Advances in Neural Information Processing Systems*, 35: 8799–8810.

Bousquet, O.; and Elisseeff, A. 2002. Stability and generalization. *The Journal of Machine Learning Research*, 2: 499–526.

Caron, M.; Misra, I.; Mairal, J.; Goyal, P.; Bojanowski, P.; and Joulin, A. 2020. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33: 9912–9924.

Caron, M.; Touvron, H.; Misra, I.; Jégou, H.; Mairal, J.; Bojanowski, P.; and Joulin, A. 2021. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, 9650–9660.

Chen, T.; Kornblith, S.; Norouzi, M.; and Hinton, G. 2020a. A simple framework for contrastive learning of visual representations. In *International conference on machine learning*, 1597–1607. PMLR.

Chen, X.; and He, K. 2021. Exploring simple siamese representation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15750–15758.

Chen, Z.; Ngiam, J.; Huang, Y.; Luong, T.; Kretzschmar, H.; Chai, Y.; and Anguelov, D. 2020b. Just pick a sign: Optimizing deep multitask models with gradient sign dropout. *Advances in Neural Information Processing Systems*, 33: 2039–2050.

Cıbuk, M.; Budak, U.; Guo, Y.; Ince, M. C.; and Sengur, A. 2019. Efficient deep features selections and classification for flower species recognition. *Measurement*, 137: 7–13.

Deng, W.; Gould, S.; and Zheng, L. 2022. On the strong correlation between model invariance and generalization. *Advances in Neural Information Processing Systems*, 35: 28052–28067.

Deshpande, S.; Wang, K.; Sreenivas, D.; Li, Z.; and Kuleshov, V. 2022. Deep multi-modal structural equations for causal effect estimation with unstructured proxies. *Advances in Neural Information Processing Systems*, 35: 10931–10944.

Djolonga, J.; Yung, J.; Tschannen, M.; Romijnders, R.; Beyer, L.; Kolesnikov, A.; Puigcerver, J.; Minderer, M.; D'Amour, A.; Moldovan, D.; et al. 2020. On Robustness and Transferability of Convolutional Neural Networks. *arXiv preprint arXiv:2007.08558*.

Doersch, C.; Gupta, A.; and Efros, A. A. 2015. Unsupervised visual representation learning by context prediction. In *Proceedings of the IEEE international conference on computer vision*, 1422–1430.

Ermolov, A.; Siarohin, A.; Sangineto, E.; and Sebe, N. 2021. Whitening for self-supervised representation learning. In *International Conference on Machine Learning*, 3015–3024. PMLR.

Everingham, M.; Van Gool, L.; Williams, C. K.; Winn, J.; and Zisserman, A. 2010. The pascal visual object classes (voc) challenge. *International journal of computer vision*, 88(2): 303–338.

Finn, C.; Abbeel, P.; and Levine, S. 2017. Model-agnostic meta-learning for fast adaptation of deep networks. In *International conference on machine learning*, 1126–1135. PMLR.

Gat, I.; Schwartz, I.; Schwing, A.; and Hazan, T. 2020. Removing bias in multi-modal classifiers: Regularization by maximizing functional entropies. *Advances in Neural Information Processing Systems*, 33: 3197–3208.

Gordon, J.; Bronskill, J.; Bauer, M.; Nowozin, S.; and Turner, R. E. 2018. Meta-learning probabilistic inference for prediction. *arXiv preprint arXiv:1805.09921*.

Grill, J.-B.; Strub, F.; Altché, F.; Tallec, C.; Richemond, P.; Buchatskaya, E.; Doersch, C.; Avila Pires, B.; Guo, Z.; Gheshlaghi Azar, M.; et al. 2020. Bootstrap your own latent-a new approach to self-supervised learning. *Advances in neural information processing systems*, 33: 21271–21284.

Gutmann, M.; and Hyvärinen, A. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, 297–304. JMLR Workshop and Conference Proceedings.

He, K.; Chen, X.; Xie, S.; Li, Y.; Dollár, P.; and Girshick, R. 2022. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 16000–16009.

He, K.; Fan, H.; Wu, Y.; Xie, S.; and Girshick, R. 2020. Momentum contrast for unsupervised visual representation learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 9729–9738.

He, K.; Gkioxari, G.; Dollár, P.; and Girshick, R. 2017. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, 2961–2969.

Hospedales, T.; Antoniou, A.; Micaelli, P.; and Storkey, A. 2021. Meta-learning in neural networks: A survey. *IEEE transactions on pattern analysis and machine intelligence*, 44(9): 5149–5169.

Hu, Z.; Zhao, Z.; Yi, X.; Yao, T.; Hong, L.; Sun, Y.; and Chi, E. 2022. Improving multi-task generalization via regularizing spurious correlation. *Advances in Neural Information Processing Systems*, 35: 11450–11466.

Jiang, Y.; Chen, Z.; Kuang, K.; Yuan, L.; Ye, X.; Wang, Z.; Wu, F.; and Wei, Y. 2022. The Role of Deconfounding in Meta-learning. In *International Conference on Machine Learning*, 10161–10176. PMLR.

Le, Y.; and Yang, X. 2015. Tiny imagenet visual recognition challenge. *CS 231N*, 7(7): 3.

Li, Y.; Pogodin, R.; Sutherland, D. J.; and Gretton, A. 2021. Self-supervised learning with kernel dependence maximization. *Advances in Neural Information Processing Systems*, 34: 15543–15556.

Lin, T.-Y.; Maire, M.; Belongie, S.; Hays, J.; Perona, P.; Ramanan, D.; Dollár, P.; and Zitnick, C. L. 2014. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, 740–755. Springer.

Liu, X.; Wang, Z.; Li, Y.-L.; and Wang, S. 2022a. Self-supervised learning via maximum entropy coding. *Advances in Neural Information Processing Systems*, 35: 34091–34105.

Liu, Y.; Jin, M.; Pan, S.; Zhou, C.; Zheng, Y.; Xia, F.; and Philip, S. Y. 2022b. Graph self-supervised learning: A survey. *IEEE Transactions on Knowledge and Data Engineering*, 35(6): 5879–5900.

Maji, S.; Rahtu, E.; Kannala, J.; Blaschko, M.; and Vedaldi, A. 2013. Fine-grained visual classification of aircraft. *arXiv preprint arXiv:1306.5151*.

Mignacco, F.; Krzakala, F.; Lu, Y.; Urbani, P.; and Zdeborova, L. 2020. The role of regularization in classification of high-dimensional noisy gaussian mixture. In *International conference on machine learning*, 6874–6883. PMLR.

Milan, A.; Leal-Taixé, L.; Reid, I.; Roth, S.; and Schindler, K. 2016. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*.

Mo, S.; and Tong, S. 2024. Connecting joint-embedding predictive architecture with contrastive self-supervised learning. *Advances in neural information processing systems*, 37: 2348–2377.

Nam, J.; Cha, H.; Ahn, S.; Lee, J.; and Shin, J. 2020. Learning from failure: De-biasing classifier from biased classifier. *Advances in Neural Information Processing Systems*, 33: 20673–20684.

Ni, R.; Shu, M.; Souri, H.; Goldblum, M.; and Goldstein, T. 2021. The close relationship between contrastive learning and meta-learning. In *International conference on learning representations*.

Pan, S. J.; and Yang, Q. 2009. A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10): 1345–1359.

Pearl, J. 2009. *Causality*. Cambridge university press.

Perazzi, F.; Pont-Tuset, J.; McWilliams, B.; Van Gool, L.; Gross, M.; and Sorkine-Hornung, A. 2016. A benchmark dataset and evaluation methodology for video object segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 724–732.

Ramesh, A.; Pavlov, M.; Goh, G.; Gray, S.; Voss, C.; Radford, A.; Chen, M.; and Sutskever, I. 2021. Zero-shot text-to-image generation. In *International conference on machine learning*, 8821–8831. Pmlr.

Ren, S.; He, K.; Girshick, R.; and Sun, J. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Huang, Z.; Karpathy, A.; Khosla, A.; Bernstein, M.; et al. 2015. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 115(3): 211–252.

Schiappa, M. C.; Rawat, Y. S.; and Shah, M. 2023. Self-supervised learning for videos: A survey. *ACM Computing Surveys*, 55(13s): 1–37.

Schölkopf, B.; Locatello, F.; Bauer, S.; Ke, N. R.; Kalchbrenner, N.; Goyal, A.; and Bengio, Y. 2021. Toward causal representation learning. *Proceedings of the IEEE*, 109(5): 612–634.

Standley, T.; Zamir, A.; Chen, D.; Guibas, L.; Malik, J.; and Savarese, S. 2020. Which tasks should be learned together in multi-task learning? In *International Conference on Machine Learning*, 9120–9132. PMLR.

Sun, C.; He, P.; Ji, Q.; Zang, Z.; Li, J.; Wang, R.; and Wang, W. 2024a. M2i2: Learning efficient multi-agent communication via masked state modeling and intention inference. *arXiv preprint arXiv:2501.00312*.

Sun, C.; He, P.; Wang, R.; and Zheng, C. 2025. Revisiting Communication Efficiency in Multi-Agent Reinforcement Learning from the Dimensional Analysis Perspective. In Das, S.; Nowé, A.; and Vorobeychik, Y., eds., *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, 1977–1986. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Sun, C.; Wu, B.; Wang, R.; Hu, X.; Yang, X.; and Cong, C. 2021. Intrinsic Motivated Multi-Agent Communication. In *Proceedings of the 20th International Conference on Autonomous Agents and MultiAgent Systems*, AAMAS '21, 1668–1670. Richland, SC: International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9781450383073.

Sun, C.; Zang, Z.; Li, J.; Li, J.; Xu, X.; Wang, R.; and Zheng, C. 2024b. T2mac: Targeted and trusted multi-agent communication through selective engagement and evidence-driven integration. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, 15154–15163.

Suter, R.; Miladinovic, D.; Schölkopf, B.; and Bauer, S. 2019. Robustly disentangled causal mechanisms: Validating deep representations for interventional robustness. In *International Conference on Machine Learning*, 6056–6065. PMLR.

Venkateswara, H.; Eusebio, J.; Chakraborty, S.; and Panchanathan, S. 2017. Deep hashing network for unsupervised domain adaptation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 5018–5027.

Vilalta, R.; and Drissi, Y. 2002. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18: 77–95.

Voigtlaender, P.; Krause, M.; Osep, A.; Luiten, J.; Sekar, B. B. G.; Geiger, A.; and Leibe, B. 2019. Mots: Multi-object tracking and segmentation. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 7942–7951.

Wang, T.; and Isola, P. 2020. Understanding contrastive representation learning through alignment and uniformity on the hypersphere. In *International Conference on Machine Learning*, 9929–9939. PMLR.

Wang, Z.; Tsvetkov, Y.; Firat, O.; and Cao, Y. 2020. Gradient vaccine: Investigating and improving multi-task optimization in massively multilingual models. *arXiv preprint arXiv:2010.05874*.

Wang, Z.; Zhao, H.; Li, Y.-L.; Wang, S.; Torr, P.; and Bertinetto, L. 2021. Do different tracking tasks require different appearance models? *Advances in Neural Information Processing Systems*, 34: 726–738.

Wolff, P.; and Zettergren, M. 2019. A vector model of causal meaning. In *Proceedings of the twenty-fourth annual conference of the Cognitive Science Society*, 944–949. Routledge.

Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.-Y.; and Girshick, R. 2019. Detectron2. https://github.com/facebookresearch/detectron2.

Wu, Y.; Lim, J.; and Yang, M.-H. 2013. Online object tracking: A benchmark. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2411–2418.

Xu, J.; Xiao, L.; and López, A. M. 2019. Self-supervised domain adaptation for computer vision tasks. *IEEE Access*, 7: 156694–156706.

Ying, W.; Zhang, Y.; Huang, J.; and Yang, Q. 2018. Transfer learning via learning to transfer. In *International Conference on Machine Learning*, 5085–5094. PMLR.

Yosinski, J.; Clune, J.; Bengio, Y.; and Lipson, H. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems*, 27.

Yu, T.; Kumar, S.; Gupta, A.; Levine, S.; Hausman, K.; and Finn, C. 2020. Gradient surgery for multi-task learning. *Advances in Neural Information Processing Systems*, 33: 5824–5836.

Zamir, A. R.; Sax, A.; Shen, W.; Guibas, L. J.; Malik, J.; and Savarese, S. 2018. Taskonomy: Disentangling task transfer learning. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 3712–3722.

Zang, Z.; Sun, C.; Liu, L.; Sun, F.; and Zheng, C. 2025. Loss of Plasticity: A New Perspective on Solving Multi-Agent Exploration for Sparse Reward Tasks. In Das, S.; Nowé, A.; and Vorobeychik, Y., eds., *Proceedings of the 24th International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2025, Detroit, MI, USA, May 19-23, 2025*, 2299–2308. International Foundation for Autonomous Agents and Multiagent Systems / ACM.

Zbontar, J.; Jing, L.; Misra, I.; LeCun, Y.; and Deny, S. 2021. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, 12310–12320. PMLR.

Zhu, S.; An, B.; and Huang, F. 2021. Understanding the generalization benefit of model invariance from a data perspective. *Advances in Neural Information Processing Systems*, 34: 4328–4341.

Appendix

The appendices provide supplementary materials, including theoretical proofs, implementation details, and extended experimental results, as outlined below:

Appendix I presents the proof of the theorem 1.

Appendix II describes the setup and all results of the motivation experiment.

Appendix III provides the pseudo-code of the proposed method.

Appendix IV details the experimental setup and full results, including unsupervised validation, semi-supervised validation, transfer learning, and ablation studies.

I: Proof of Theorem 1

Taking contrastive learning as an example, SSL can be regarded as multiple binary classification problems, whose label information is generated by the adopted data augmentation. Therefore, we consider using two binary classification problems as research objects to analyze the task conflict problem caused by the introduction of multi-task information into SSL. Here, Y_i and Y_j represent the label variables of task i and task j respectively, and belong to $\{\pm 1\}$, while X_i and X_j correspond to sample variables.

From a generation perspective, training samples can be directly determined by generative factors. We set the generative factors of samples in tasks i and j to be F_i^u and F_j^u respectively, which can be characterized by Gaussian distribution. Assuming that there is no overlap between F_i^u and F_j^u, they can be represented by a multidimensional Gaussian distribution:

$$\begin{aligned} F_i^u &\sim \mathcal{N}(Y_i \cdot \mu_i, \sigma_i^2 I) \\ F_j^u &\sim \mathcal{N}(Y_j \cdot \mu_j, \sigma_j^2 I) \end{aligned} \tag{8}$$

where μ_i, σ_i^2 represents the mean and variance respectively. For simplicity, we assume that labels originate from different distributions and the sampling distribution of labels is the same, $P(Y=1) = P(Y=-1) = 0.5$ (not mandatory). In the multi-task joint learning setting, we define c_{ij} as the task correlation caused by task conflict, $P(Y_i = Y_j) = 1 - P(Y_i = Y_j) = c_{ij}$. Next, we can deduce that the optimal classifier for each task has non-zero weights for its non-generative factors according to the following theorem.

Theorem 1 *If the correlation between Y_i and Y_j is not equal to 0.5, then the optimal model for task i has a non-zero weight on F_j^u. If the correlation is equal to 0.5 with limited training samples, then the optimal classifier for task i also has non-zero weight on factor F_j^u.*

Proof 1 *We train a model using joint learning of two tasks and analyze whether the best classifier of the target task for a single task will be affected by the generative factors from other tasks. Taking task i as an example, the optimal Bayesian classifier is expressed as:*

$$\begin{aligned} P(Y_i|F_i^u, F_j^u) &= \frac{P(Y_i, F_i^u, F_j^u)}{P(F_i^u, F_j^u)} \\ &= \frac{P(Y_i, F_i^u, F_j^u)}{\sum_{Y_i \in \{-1,1\}} P(Y_i, F_i^u, F_j^u)} \end{aligned} \tag{9}$$

where the probability of $P(Y_i, F_i^u, F_j^u)$ could be written as:

$$\begin{aligned} P(Y_i, F_i^u, F_j^u) &= P(Y_i, F_i^u) \cdot P(F_j^u|Y_i, F_i^u) \\ &= P(Y_i, F_i^u) \cdot P(F_i^u|Y_i) \\ &= P(Y_i, F_i^u) \cdot \sum\nolimits_{Y_j \in \{-1,1\}} P(F_j^u, Y_j|Y_i) \\ &= P(Y_i) P(F_i^u|Y_i) \cdot \\ &\quad \sum\nolimits_{Y_j \in \{-1,1\}} P(F_j^u|Y_j) P(Y_j|Y_i) \end{aligned} \tag{10}$$

Since the generative factors obey Gaussian distribution, $P(Y_i, F_i^u, F_j^u) = sigmoid(\frac{\mu_i}{\sigma_i^2} F_i^u + \frac{\mu_j}{\sigma_j^2} F_j^u)$ where $\frac{\mu_i}{\sigma_i^2}$ and $\frac{\mu_j}{\sigma_j^2}$ are the regression vectors for the optimal Bayesian classifier, then we have:

$$\begin{aligned} &P(Y_i, F_i^u, F_j^u) \\ &= P(Y_i) P(F_i^u|Y_i) \cdot \sum\nolimits_{Y_j \in \{-1,1\}} P(F_j^u|Y_j) P(Y_j|Y_i) \\ &\propto e^{Y_i \cdot \frac{\mu_i}{\sigma_i^2} F_i^u} (c_{ij} e^{Y_i \cdot \frac{\mu_j}{\sigma_j^2} F_j^u} + (1 - c_{ij}) e^{-Y_i \cdot \frac{\mu_j}{\sigma_j^2} F_j^u}) \\ &= c_{ij} e^{Y_i \cdot (\frac{\mu_i}{\sigma_i^2} F_i^u + \frac{\mu_j}{\sigma_j^2} F_j^u)} + (1 - c_{ij}) e^{Y_i \cdot (\frac{\mu_i}{\sigma_i^2} F_i^u - \frac{\mu_j}{\sigma_j^2} F_j^u)} \\ &= c_{ij} e^{Y_i \cdot \gamma_1} + (1 - c_{ij}) e^{Y_i \cdot \gamma_2} \end{aligned} \tag{11}$$

where $\gamma_1 = \frac{\mu_i}{\sigma_i^2} F_i^u + \frac{\mu_j}{\sigma_j^2} F_j^u, \gamma_2 = \frac{\mu_i}{\sigma_i^2} F_i^u - \frac{\mu_j}{\sigma_j^2} F_j^u$, Thus, we have:

$$P(Y_i|F_i^u, F_j^u) = \frac{1}{1 + \frac{p_{tc} e^{Y_i \cdot \gamma_1} + (1 - p_{tc}) e^{Y_i \cdot \gamma_2}}{p_{tc} e^{-Y_i \cdot \gamma_1} + (1 - p_{tc}) e^{-Y_i \cdot \gamma_2}}} \tag{12}$$

When $c_{ij} = 0.5$:

$$P(Y_i|F_i^u, F_j^u) = \frac{1}{1 + e^{Y_i \cdot (\gamma_1 + \beta^-)}} = \frac{1}{1 + e^{2Y_i \cdot (\frac{\mu_i}{\sigma_i^2} F_i^u)}} \tag{13}$$

It can be shown that when the correlation is 0.5, the optimal classifier only uses the generative factors F_i^u for task i.

When $c_{ij} \neq 0.5$, according to Eq.12, the optimal classifier for each task will be affected by the generative factors of another task.

II: Motivational Experiment

In this section, we introduce the full details and experimental results of the motivating experiments in the main text.

Existing SSL models do not explicitly incorporate transferability into their loss functions or architectures, prompting us to explore whether modeling transferability during training can further enhance performance. Motivated by meta-learning (Finn, Abbeel, and Levine 2017; Hospedales et al. 2021) which acquires transferable knowledge from diverse tasks, we construct multiple tasks during the SSL training phase to examine the impact on transferability. In our experiments, each randomly sampled training batch B is evenly divided into two subsets, B_1 and B_2, which serve as inputs for two independent tasks. For each task, samples are augmented using two random transformation strategies

Method	CIFAR-10		CIFAR-100		STL-10	
	Top-1	5-NN	Top-1	5-NN	Top-1	5-NN
SimCLR	91.80±0.16	88.42±0.15	66.83±0.23	56.57±0.18	90.51±0.14	85.68±0.11
BYOL	91.73±0.23	89.26±0.22	66.60±0.16	56.82±0.18	91.99±0.13	88.64±0.20
MoCo	91.69±0.13	88.66±0.15	67.02±0.16	56.29±0.25	90.64±0.29	88.01±0.20
Barlow Twins	90.88±0.19	88.78±0.21	66.67±0.10	56.39±0.24	90.71±0.13	85.31±0.23
SimSiam	91.71±0.27	88.65±0.17	67.22±0.26	56.36±0.19	91.01±0.20	88.16±0.22
W-MSE	91.99±0.12	89.87±0.25	67.64±0.16	56.45±0.26	91.75±0.23	88.59±0.15
SwAV	90.17±0.19	89.52±0.24	66.56±0.17	57.01±0.24	90.72±0.29	86.24±0.26
DINO	91.83±0.25	90.15±0.33	67.15±0.21	56.48±0.19	91.03±0.12	86.15±0.25
SSL-HSIC	91.95±0.14	89.91±0.17	67.22±0.26	57.01±0.27	92.09±0.20	88.91±0.29
C-JEPA	91.65±0.08	90.05±0.21	67.46±0.13	56.88±0.22	91.89±0.13	88.07±0.15
VICRegL	90.99±0.13	88.75±0.26	68.03±0.32	57.34±0.29	92.12±0.26	90.01±0.21
SimCLR+TC^2	93.84±0.22	91.75±0.23	**69.89±0.26**	59.65±0.27	93.89±0.27	88.99±0.26
BYOL+TC^2	**94.89±0.24**	**91.96±0.22**	68.93±0.25	59.79±0.26	**94.93±0.25**	**91.19±0.27**
C-JEPA+TC^2	93.55±0.12	91.34±0.17	68.23±0.11	58.52±0.12	93.47±0.15	89.17±0.23
Barlow Twins+TC^2	94.41±0.27	91.81±0.26	68.86±0.21	**59.99±0.28**	94.35±0.21	89.51±0.27
Method	Tiny ImageNet		ImageNet-100		ImageNet	
	Top-1	5-NN	Top-1	5-NN	Top-1	5-NN
SimCLR	48.82±0.15	32.86±0.25	70.15±0.16	89.75±0.14	68.32±0.31	89.76±0.23
BYOL	51.01±0.12	36.24±0.28	74.89±0.23	92.83±0.21	74.31±0.29	91.62±0.25
MoCo	50.92±0.22	35.55±0.18	72.81±0.12	89.75±0.11	71.37±0.27	88.94±0.11
Barlow Twins	49.74±0.26	33.61±0.19	72.88±0.23	90.99±0.19	73.22±0.31	91.01±0.27
SimSiam	51.14±0.21	35.67±0.18	73.01±0.22	92.61±0.27	70.02±0.14	88.76±0.23
W-MSE	49.22±0.16	35.44±0.10	76.01±0.27	93.12±0.21	70.85±0.31	91.57±0.20
SwAV	52.02±0.25	37.40±0.11	75.77±0.18	92.86±0.17	69.12±0.24	89.38±0.26
DINO	51.13±0.30	37.86±0.19	75.43±0.18	93.32±0.19	70.58±0.24	91.32±0.27
SSL-HSIC	51.37±0.15	36.03±0.12	74.77±0.22	92.56±0.25	72.13±0.28	90.33±0.29
VICRegL	51.52±0.13	36.24±0.16	75.96±0.21	92.97±0.26	70.24±0.27	91.60±0.24
SimCLR+TC^2	52.21±0.25	37.42±0.26	74.89±0.28	92.97±0.29	72.99±0.30	92.96±0.27
BYOL+TC^2	**54.25±0.28**	**39.45±0.27**	**77.74±0.29**	**95.82±0.25**	**77.05±0.26**	**93.99±0.29**
Barlow Twins+TC^2	52.94±0.19	37.55±0.23	75.95±0.22	93.34±0.23	75.94±0.24	92.99±0.27

Table 5: Unsupervised Evaluation. The classification accuracies with a ResNet-18 for the first four datasets and ResNet-50 for the last two datasets. The best results are highlighted in bold.

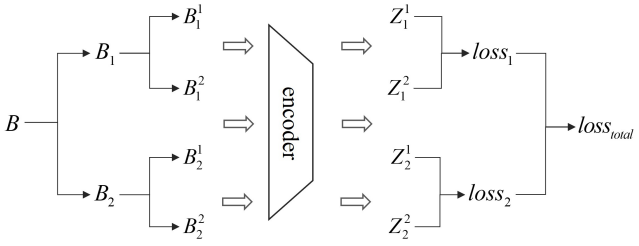


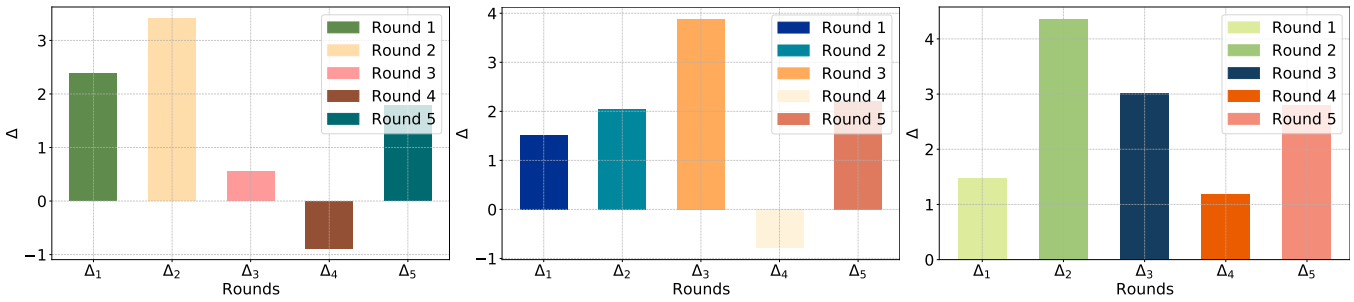
Figure 4: Model training pipeline for the motivational experiment. The encoder is jointly optimized based on two tasks.

to generate paired views. These views are processed by a shared encoder (ResNet-18) to extract feature representations. Each task follows a conventional contrastive learning framework (e.g., SimCLR), and the encoder is jointly optimized using the combined loss from both tasks, as illustrated in Figure 4. We adopt SimCLR as the baseline model, training on ImageNet-100 and testing on CIFAR-100. First, we measure the baseline model's Top-1 accuracy without task construction, denoted as $acc(\mathrm{SimCLR})$. Then, under identical experimental settings, we evaluate the performance of the multi-task learning mechanism across five independent training rounds, with the i-th round's accuracy denoted as $acc(\mathrm{SimCLR} + \mathrm{T}, i)$. Finally, we calculate the improvement in classification performance for each round as $\Delta_i = acc(\mathrm{SimCLR} + \mathrm{T}, i) - acc(\mathrm{SimCLR})$. The experimental results in Figure 1a show that Incorporating task-level information into SSL improves transferability, yet it remains vulnerable to gradient conflicts between tasks.

To validate the universality of this phenomenon, we further test on additional datasets. Specifically, in addition to CIFAR-100, we selected the Flower102 (Cıbuk et al. 2019) and Aircraft (Maji et al. 2013) datasets to evaluate model transferability. The experimental results in Figure 5 reached a similar conclusion: incorporating task-level information into SSL significantly improves performance, although outcomes vary between the best and worst cases.

III: Pseudo-Code

In this section, the pseudocode of the algorithm is presented in Algorithm 1.



(a) Performance on CIFAR-100 (b) Performance on Flower102 (c) Performance on Aircraft

Figure 5: Effect of task-level information. It shows the effect of SimCLR trained on ImageNet and tested on three different datasets before and after introducing task-level information in five runs.

Algorithm 1: SSL with TC^2

Input: Training set X; initialize self-supervised model with a encoder f_θ; Randomly initialize networks f_v and f_w.
Parameter: Learning rates β_1 and β_2 for the learning of f_θ; Learning rates α_1, α_2, α_3, α_4 for the learning of W and V; Parameter λ_s for the balancing term in loss function.
Output: self-supervised model f_θ

for sample batch X_{tr} from X **do**
  Obtain $B = \{B_1, ..., B_k, ..., B_K\}$ via splitting
  Obtain $B^{aug} = \{B_1^{aug}, ..., B_k^{aug}, ..., B_K^{aug}\}$ via augmentation
  Update f_v and f_w with Eq.4 and Eq.5
  Update SSL model f_θ via Eq.6 and Eq.7
end for
return f_θ

IV: Experiment Details and Results

Data Augmentation and Paramter setting

We adopt standard image augmentation strategies, including random cropping, resizing, horizontal flipping, color jittering, grayscaling, and Gaussian blurring. Specifically, each image is randomly cropped to a size ranging from 20% to 100% of the original area, with an aspect ratio randomly selected between $3/4$ and $4/3$. For generating two augmented views, the following transformations are applied with specified probabilities: horizontal flipping with probability 0.5, color jittering with configuration $(0.4, 0.4, 0.4, 0.1)$ with probability 0.8, and conversion to grayscale with probability 0.1. For ImageNet-100 and ImageNet, we follow slightly stronger augmentation settings: cropping size from 8% to 100% of the original area, color jittering with configuration $(0.8, 0.8, 0.8, 0.2)$ applied with probability 0.8, grayscaling with probability 0.2, and Gaussian blurring with probability 0.5. All experiments are conducted on NVIDIA V100 GPUs. To ensure statistical robustness, all reported results are averaged over 10 runs with different random seeds.

We adopt ResNet-18 and ResNet-50 as the backbone networks to extract visual features. Each backbone is followed by a linear projection head that maps the learned representations into a latent embedding space for contrastive learning.

Method	pre-train data	ViT-B	ViT-L	ViT-H	ViT-H$_{448}$
DINO	IN1K	82.8	-	-	-
MoCo	IN1K	83.2	84.1	-	-
BEiT	IN1K+DALLE	83.2	85.2	-	-
MAE	IN1K	83.6	85.9	86.9	87.8
MAE+TC^2	IN1K	86.2	87.3	88.2	89.0

Table 6: Comparisons with previous results on ImageNet-1K. The ViT models are B/16, L/16, H/14. The pre-training data is the ImageNet-1K training set (except the tokenizer in BEiT was pre-trained on 250M DALLE data (Ramesh et al. 2021)). All results are on an image size of 224, except for ViT-H with an extra result of 448.

To implement our proposed Task Conflict Calibration (TC^2) module, we use a lightweight three-layer multi-layer perceptron (MLP), consisting of two hidden layers with ReLU activation and an output layer producing task-specific modulation weights. By default, we construct four self-supervised tasks within each mini-batch, enabling the model to capture diverse task-level variations and facilitating the analysis of task conflict. This configuration balances computational efficiency with effective task modeling.

Unsupervised Learning

We employ the most commonly used SSL protocol (Ermolov et al. 2021) to train a supervised classifier based on the frozen feature extractors. In experiment, we use a ResNet-18 as a feature extractor for the first four datasets, and a ResNet-50 for other datasets. Several existing SSL methods are used as baselines, including SimCLR (Chen et al. 2020a), BarlowTwins (Zbontar et al. 2021), BYOL (Grill et al. 2020), SimSiam (Chen and He 2021), W-MES (Ermolov et al. 2021), SwAV (Caron et al. 2020), MoCo (He et al. 2020),DINO (Caron et al. 2021), SSL-HSIC (Li et al. 2021) and VICRegL (Bardes, Ponce, and LeCun 2022),and C-JEPA (Mo and Tong 2024). The classification accuracies of a linear classifier (Top-1) and a 5-nearest neighbors (5-NN) classifier for evaluation on small datasets, and Top-5 classification accuracies for evaluation on other datasets. Table 5 shows the results of unsupervised

Method	A → C	A → P	A → R	C → A	C → P	C → R	P → A	P → C	P → R	R → A	R → C	R → P
SimCLR	58.2	63.5	69.8	78.9	69.7	66.8	63.4	52.3	58.4	56.1	72.9	71.0
SimCLR+TC^2	60.1	65.7	71.2	79.0	73.6	68.2	66.1	56.4	60.5	58.0	59.4	74.9
BYOL	58.5	64.1	69.2	78.1	68.5	67.2	64.0	53.8	58.7	56.9	72.1	71.4
BYOL+TC^2	60.2	66.4	72.4	80.0	72.4	70.2	67.3	56.4	60.2	58.8	75.9	73.2
Barlow Twins	59.3	63.8	70.2	79.5	70.2	67.6	64.3	54.0	58.1	57.0	71.8	69.4
Barlow Twins+TC^2	61.2	65.7	72.9	81.3	73.8	70.2	66.1	57.3	60.8	60.1	74.6	71.5

Table 7: Transfer learning on OfficeHome dataset.

Method	Epochs	1%		10%	
		Top-1	Top-5	Top-1	Top-5
MoCo	200	43.8±0.2	72.3±0.1	61.9±0.1	84.6±0.2
BYOL	200	54.8±0.2	78.8±0.1	68.0±0.2	88.5±0.2
SimCLR	1000	48.3±0.2	75.5±0.1	65.6±0.1	87.8±0.2
MoCo	1000	52.3±0.1	77.9±0.2	68.4±0.1	88.0±0.2
BYOL	1000	53.2±0.2	78.4±0.2	68.8±0.2	89.0±0.1
SimSiam	1000	54.9±0.2	79.5±0.2	68.1±0.1	89.0±0.3
SwAV	1000	53.9±0.1	78.5±0.2	70.2±0.1	89.9±0.2
SSL-HSIC	1000	55.4±0.3	80.1±0.2	70.4±0.1	90.1±0.1
BarlowTwins	1000	55.0±0.1	79.2±0.1	69.7±0.2	89.3±0.2
VICRegL	1000	54.9±0.1	79.6±0.2	67.2±0.1	89.4±0.2
SimCLR+TC^2	1000	50.9±0.2	77.4±0.1	67.3±0.2	89.9±0.3
BYOL+TC^2	1000	55.8±0.3	80.2±0.2	70.5±0.2	90.8±0.1
Barlow Twins+TC^2	1000	**57.9**±0.2	**81.6**±0.3	**71.9**±0.2	**92.2**±0.2

Table 8: Semi-Supervised Evaluation on ImageNet. We finetune the pre-trained model using 1% and 10% training samples of ImageNet, and the Top-1 and Top-5 under linear evaluation are reported. The best results are highlighted in bold.

experiments on multiple datasets, where "+TC^2" means our proposed method with task conflict calibration. On small-scale datasets, BYOL and SimCLR improve by more than 3% on CIFAR10 and Tiny ImageNet (Le and Yang 2015) datasets. On large-scale datasets, SimCLR also improved over 4% on the ImageNet-100 dataset (Russakovsky et al. 2015). In summary, our method adds task information to baselines while mitigating the impact of task conflict, improving results in unsupervised learning. In addition, we conducted comparative experiments with generative self-supervised learning methods, and the results are presented in Table 6. The results also demonstrate the effectiveness of our method.

To provide a fair comparison with existing multi-task learning approaches, we conducted experiments on the CIFAR-10 dataset, using the same task partitioning strategy described earlier. The baseline methods included PCGrad (Yu et al. 2020), GradVac (Wang et al. 2020) and GradDrop (Chen et al. 2020b), and we evaluated performance using the Top-1 classification accuracy. For self-supervised learning, we employed SimCLR. Experimental results show that our method outperforms by 2.13%, 1.71%, and 1.33%, respectively.

Semi-supervised Learning

Here, we adopt the most commonly used protocol for semi-supervised learning (Zbontar et al. 2021). We create two balanced subsets by sampling 1% and 10% of the training dataset. We fine-tune all models on these two subsets for 50 epochs with different learning rates for the classifier and the backbone network. Specifically, we use 0.05 and 1.0 for the classifier, and 0.0001 and 0.01 for the backbone network, on the 1% and 10% subsets. Table 8 reports the classification results on ImageNet compared with existing methods using two pre-trained models. From the results, SimCLR increases about 2.3%, and Barlow Twins increases about 3% at the 1% subset setting.

Transfer Learning

Video-based Task. We transfer pretrained models to a series of video tasks based on an evaluation platform called UniTrack (Wang et al. 2021). UniTrack does not require additional training and, therefore, provides a more direct comparison between different representations. The video dataset we use contains SOT (Wu, Lim, and Yang 2013), VOS (Perazzi et al. 2016), MOT (Milan et al. 2016), MOTS (Voigtlaender et al. 2019), PoseTrack (Andriluka et al. 2018). All methods use the same ResNet-50 backbone and are evaluated based on UniTrack.

Object Detection and Instance Segmentation. We adopt the most commonly used protocol for transfer learning (Zbontar et al. 2021). We evaluate our framework on Pascal VOC and COCO datasets for object detection

Method	n_f=$d/4$	n_f=$d/2$	n_f=d	n_f=$2d$
SimCLR+TC^2	90.78	92.53	93.35	93.01
BYOL+TC^2	90.54	92.67	93.24	92.87

Table 9: The impact of the number of factors.

λ_s	0.001	0.01	0.1	1
SimCLR+TC^2	91.65	92.53	93.45	93.15
MAE+TC^2	93.49	96.09	96.83	96.81

Table 10: Parametric analysis of λ_s.

and instance segmentation. Specifically, we use Faster R-CNN (Ren et al. 2015) with a C4-backbone (Wu et al. 2019) for the VOC detection task, and use Mask R-CNN (He et al. 2017) with the same C4-backbone and a $1\times$ schedule for the COCO detection and segmentation tasks. In the fine-tuning stage, we search for an optimal learning rate on the target datasets and keep other parameters the same as in the Detection2 library. In the training stage, we train our Faster R-CNN model on the VOC 07+12 trainval set (16K images) and reduce the initial learning rate by a factor of 10 at 18K and 22K iterations. We also train Faster R-CNN on the VOC 07 trainval set (5K images), but with fewer iterations. For the Mask R-CNN model, we train it on the COCO 2017 train split and report the results on the val split.

Ablation Study Results

The number of generative factors n_f. The dimension of the factor space spanned by the generative factors is the number of n_f. We analyze the impact of n_f on CIFAR10 dataset. Specifically, we set n_f to $d/4, d/2, d, 2d$. Experimental results in Table 9 show that optimal results are obtained when the number of generative factors is the same as the dimension of the embedding space.

The parameter λ_s. In this section, we conduct an experimental study of the model trade-off parameters. Our proposed method contain an important parameter λ_s to balance $\mathcal{L}_v$ and $\mathcal{L}_w$. Specifically, we vary in the range [0.001, 0.01, 0.1, 1], and use the SimCLR+TR method to record the classification accuracy of our method using ResNet-18 on the CIFAR-10 dataset. The results in Table 10 show that our method has small changes in accuracy under different parameter settings.

Batch size. We also investigate the impact of batch size on unsupervised learning performance. Specifically, we conduct experiments with batch sizes of 128, 256, 512, and 1024, and report the results in Table 11. The results demonstrate that our method consistently improves model performance across different batch size settings, with relatively minor sensitivity to batch size variations.

Trade-Off Performance

Considering that our proposed method is a plug-and-play method, we conduct experiments for the trade-off performance. We compare the trade-off performance of multiple

batch size	128	256	512	1024
SimCLR+TC^2	91.78	92.12	93.12	94.84
BYOL+TC^2	91.54	92.08	93.35	94.89

Table 11: The impact of the batch size.

Method	Top-1	5-NN
SimCLR	91.80	88.42
SimCLR+TC^2	**93.84**	**90.75**
SimCLR+TC^2 w/o $\mathcal{L}_v$	91.85	88.58
SimCLR+TC^2 w/o $\mathcal{L}_w$	90.79	87.65
SimCLR+TC^2 w/o $\mathcal{L}_v$ and $\mathcal{L}_w$	88.75	84.37
SimCLR+TC^2 w/o bi-level	89.79	83.14

Table 12: The impact of the loss function.

baselines before and after using our TC^2 with the same backbone. Figure 6 illustrates the trade-off performance. The results indicate that incorporating the proposed TC^2 significantly improves performance while maintaining an acceptable computational cost and parameter size compared to the original SSL methods without introducing TC^2.

Method	n_t=2	n_t=4	n_t=8	n_t=16	n_t=32
SimCLR+TC^2	93.35	93.96	93.88	93.84	94.01
BYOL+TC^2	93.24	94.23	94.18	93.89	93.27

Table 13: The impact of the number of tasks.

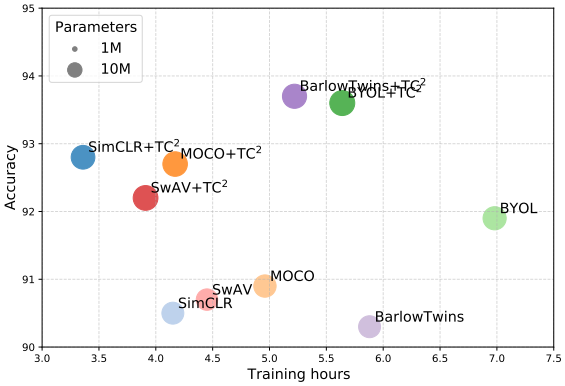


Figure 6: Trade-off performance.