SCAN: Learning to Classify Images without Labels

Wouter Van Gansbeke ， Simon Vandenhende，Stamatios Georgoulis，Marc Proesmans，Luc Van Gool

摘要

SCAN 模型提倡将特征学习和聚类解耦，又通过损失函数将语义上有意义的最近邻作为先验知识整合到可学习的方法中。但是文章提出的自标记阶段通过阈值截断的方式获得高质量伪标签数量有限，于是本文基于标签传播算法获得的标签更加平滑的特点对获得的高置信样本进行筛选，将训练集分为强伪标签数据集及弱伪标签数据集，接着使用分层混合的伪标签提升技术来提高模型泛化能力。相较于 SCAN，在 STL10 基准数据集上的准确度提升了 +2.7%。

关键词：深度聚类；一致性学习；标签传播；伪标签

1 引言

图像聚类是一项重要且具有挑战的任务，其目的是在没有标签的条件下将图像划分为不同的图像簇。目前的深度聚类方法在这个领域已经展现出优越的性能。早期深度聚类工作[1-5]通常是将传统聚类算法与将自编码器（auto-encoders，AE）或卷积神经网络（Convolutional Neural Network，CNN）相结合，通常借助一些初始化或者后处理（例如 k-means）来获得最终的聚类概率指示。然而也因此，它们适应于一些特殊的数据分布而非任意的数据分布，并且很难被应用到不同的数据中。

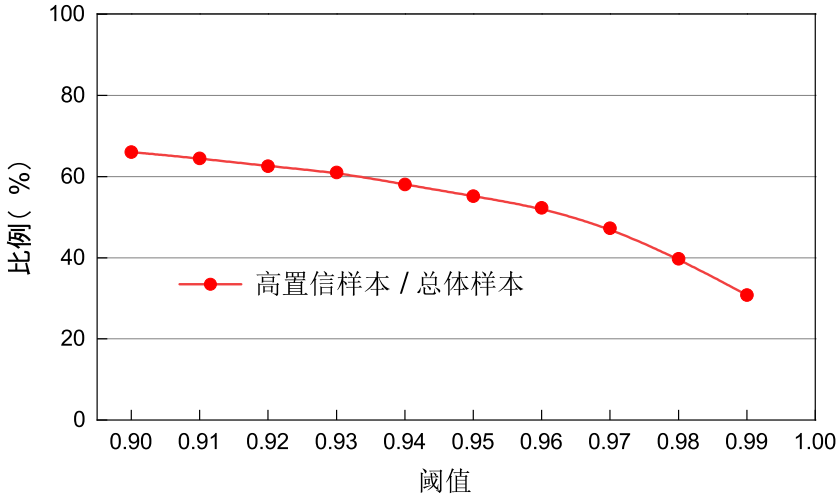


图 1: 定义聚类概率指示大于阈值的样本为高置信样本。根据训练阶段在 STL10 数据集上得到的聚类指示概率，采用不同的阈值对其截断可得到相应的高置信样本。结果表明获得的高置信样本即高质量伪标签数量有限。

近年来，对比学习[6]取得了很大成功，一些方法通过最大化图像及其增强图像之间的互信息来捕获更具区分性的图像表征，并使用多层感知机来直接获得聚类结果。然而，实例级对比损失将其他所有样本都作为负样本，这可能会将相似的样本分开，从而破坏聚类结构。又有一些方法[7-9]首先通过前置任务[10-12]学习高质量的表征，接着最大化样本及其近邻之间的聚类指示概率的相似度来获得最终的聚类结果，在实际应用中严重依赖近邻的质量。SCAN[13]一文中，提出的聚类方法利用伪标签技术来生成高质量的伪标签以提升聚类模型的鲁棒性。这类伪标签生成方法方法通常设置一个较大的概率阈值并对满足要求的样本生成 one-hot 的标签，最终利用生成的标签来更新模型。但是，这种简单的

伪标签生成方法难以获得高质量的伪标签。如图 1所示所示，将阈值设置为 0.99 得到的高置信样本仅占训练集的 30%，这意味着只能利用数据集中的少量样本来对模型进行提升。

为了解决以上问题，提出一种基于分层伪标签的图像聚类方法（SPC）。本文在 SCAN[13]的基础上，利用两个阶段来学习聚类：1）基于流形的一致性学习。构建一个静态的邻接图，通过近邻一致性来迫使相邻的样本具有相似的聚类指示概率；2）基于分层伪标签的模型精炼。根据第一阶段获得的聚类指示概率，设置一个较大的阈值将数据集划分为初始的一个包含高置信样本的强伪标签数据集及其伪标签，和一个包含其他样本的弱伪标签数据集。基于标签传播算法获得的标签更加平滑的特点，对初始的强、弱伪标签数据集进行筛选，最后使用两个数据集对模型进行训练以提高模型范化能力。将提出的方法在 STL10 基准数据集上进行测试，并于 SCAN 方法进行了比较。

2 相关工作

2.1 深度聚类

深度聚类的早期工作通常简单地将特征学习与浅层聚类相结合。例如，一些方法将堆叠式自动编码器（SAE）或卷积神经网络（CNN）与传统的聚类算法（例如 k-means[1-3]、子空间聚类[14-15] 和谱聚类[16-17]）相结合。然而，上述方法通常需要用传统聚类算法进行初始化，需后处理才能得到聚类指示概率，这限制了聚类的性能表现。接下来，端到端的学习方式将特征学习与聚类结合起来，同时学习网络的参数和对网络输出的特征进行聚类。第一组方法 (例如 DEC[1]，DAC[18]，DeepCluster[19] ，DeeperCluster[20]，或其他[21-23]) 利用 CNN 的架构作为群集图像的先验。从初始特征表示开始，通过从最有信心的样本中获得监督信号[1,18]或通过离线计算的聚类重新分配[19-20]，迭代地细化聚类。第二组方法通过最大化原始与增强图像之间标签的互信息[7,24-25]，或者最大化样本及其近邻样本之间聚类指示的似然估计[7,9]，以端到端的形式将图像用分类模型直接映射标签。然而，这种方法对初始化参数敏感或受劣质的初始嵌入特征影响，容易出现退化解[19-20]和锁定低层次的特征, 从而导致聚类性能不佳。此外,[25]中的实例级对比损失通常将其他所有样本作为负样本，这可能会将相似的样本推开并破坏语义簇结构。SCAN[13]等方法提倡两步无监督图像分类方法，同时利用图像表征和端到端学习的优点，最后通过训练-精炼（train-and-refinement）的方式来进行聚类。在训练阶段，它们通常借助具备学习高质量特征能力的前置任务（pretext task）来进行特征学习[6,26]。接着，它们通过最大化近邻样本之间聚类概率指示相似度的方式来训练分类模型[8,13,18]。在精炼阶段，它们通常选择聚类指示概率逼近 one-hot 向量的置信样本，并用 one-hot 形式的伪标签对这些样本进行标记，最后利用标记过的样本[8,13] 来对分类模型进行精炼。方法[27]通过独立训练多个聚类算法来生成多组伪标签，将其中共同的伪标签设置为高质量的伪标签，但这存在计算量大以及匈牙利算法难以对多组伪标签有效对齐的问题。本方法采取“训练-精炼”的方式，并提出分层伪标签学习，以此获得更多高质量的伪标签，进而提升对分类模型的精炼。

2.2 表征学习

图像表征学习作为图像聚类中的重要一环一直以来都是计算机视觉领域研究的重点之一，其早期工作使用手工制作的特征（例如，尺度不变特征变换（SIFT）[28]和定向梯度直方图（HOG）[29]）。

后期，一些工作采用深度神经网络（例如自动编码器[30]和对抗学习[31]）来提取图像的有效特征。

随着自监督学习的兴起，通过构建一系列前置任务并利用图像的不同先验来探索数据样本的内在分布，表征学习得到了很大的发展。这些自监督工作主要集中在前置任务上，基于图像变换前后标签不变形，通过输入图像应用变换（例如图像着色[10]、计算视觉单元[32]和图像旋转[11]）来学习良好的数据表达。具体来说，在最近成为图像处理学习方法的主要组成部分的对比学习中，通常优化对比损失（例如最大化不同视图之间的互信息[33] 或动量对比学习[6]）来拉近邻居和推开非邻居[34]。一些更新的对比学习方法结合数据增强策略（例如 SimCLR[26] 、BYOL[35]、PCL[36]和 HCSC[37]）获得了更好的图像表征。以 SimCLR[26] 为例，基于一致性正则化原则[38]，他们将样本 x_i 以及它的增强样本 $\mathcal{T}[x_i]$ 以如下方式进行对比学习：

$$\mathcal{L}_{\text{simclr}} = -\log \frac{\exp\left(\text{sim}\left(x_i, \mathcal{T}[x_i]\right)/\tau\right)}{\sum_{j=1}^{2B} 1_{j\neq i} \exp\left(\text{sim}\left(x_i, \mathcal{T}[x_j]\right)/\tau\right)} \tag{1}$$

其中 sim 是相似性函数，如余弦相似度，B 为批量大小，为可调节的温度系数；$1_{k\neq i}$ 是指示函数，当 $k \neq i$ 时对应的值为 1，否则为 0。对于 x_i，式 1中将 $\mathcal{T}[x_i]$ 作为正样本，其余 (2B-1) 个样本为负样本。InfoNCE[39]中已证明负样本越多则互信息下界越小，在足够大的批量下，式 1中损失最小化可达到表示学习的互信息最大化的目标。

2.3 伪标签技术

伪标签是已训练的分类模型对无标签数据进行概率预测并加以筛选的结果。它是一种常见的半监督分类模型训练方法，即通过联合标签样本和预测得到的伪标签来训练模型。在伪标签学习过程中，迫使模型对无标签样本进行预测，有利于最小化无标签样本预测的熵[40]，并将决策边界移动到低密度区域。目前常见的构造伪标签损失项的方法有两种：1）创建一个关于“样本-伪标签”的指定损失项[41-42]；2）将相同数量的有标签样本和伪标签样本合并为一个批量，共用标准的半监督分类损失[43-44]。Lee[45] 首次将伪标签应用于深度学习。他们将模型 $f(x)$ 的输出视为离散概率分布，把概率向量中最大元素指定的类设置为无标签样本的伪标签 $\widehat{y}_i = \text{argmax} f(x_i)$ 。最后用如下损失函数对有标签样本和伪标签样本进行联合学习：

$$\widehat{\mathcal{L}}_{ssl} = \underbrace{\frac{1}{n_l}\sum_{i=1}^{n_l} \ell_s(f(x), y)}_{labelledloss} + \lambda_u \underbrace{\frac{1}{n_u}\sum_{i=1}^{n_u} \ell_s(f(x), \tilde{y})}_{unlabelledloss} \tag{2}$$

其中，l_s 表示某种损失函数，λ 是权重参数。

最近一些年，越来越多基于伪标签的半监督学习算法被人们提出，其中包括 MixMatch[46]、ReMixMatch[47]、FixMatch[41]、LaplaceNet[48]以及 FlexMatch[49]等优秀工作。它们很好地促进了伪标签学习技术。

3 本文方法

3.1 本文方法概述

假定 $\mathcal{D} = \{x_1, x_2, \ldots, x_{|\mathcal{D}|}\}$ 为一个包含 n 张无标签的图像数据集。SCAN[13]的目标是将图像数据集 $\mathcal{D}$ 划分为 c 个清晰的簇，使得簇内图像相似，簇间图像不相似。令 Φ_θ 表示一个映射图片为语义特征的函数。如图 2所示，SCAN 使用两步无监督图像分类方法，基于最近邻一般属于同一类，利用了

表征学习和端到端学习方法的优点进行深度图像聚类：

1) 第一步，通过前置任务学习特征表示。表征学习方法在学习特征表示后需要 K-means 聚类，这会导致聚类退化。而在大多数情况下，样本的最近邻属于同一语义类，这使得它们适合于语义聚类。因此与表征学习方法不同，SCAN 基于特征相似性挖掘每个图像的最近邻。

2) 第二步，将语义上有意义的最近邻作为先验知识整合到可学习的方法中。我们通过使用损失函数将每个图像及其挖掘的近邻分类在一起，该损失函数在 softmax 之后最大化它们的点积，推动网络生成一致的 one-hot 预测。

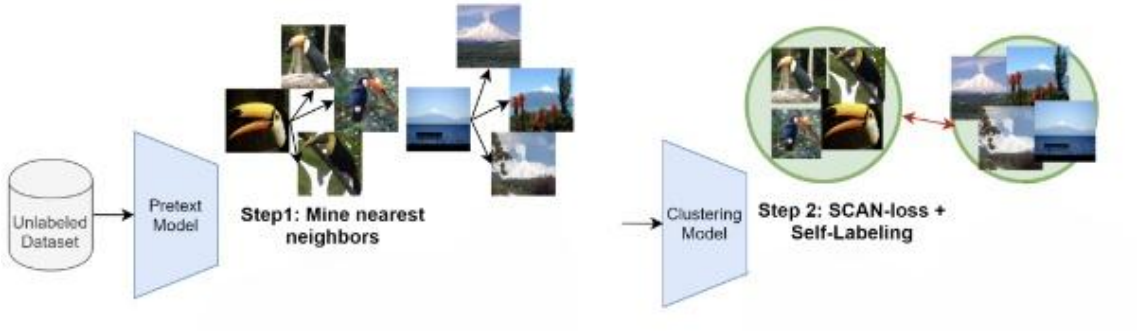


图 2: 方法示意图

以下为 SCAN[13]的具体步骤。首先，我们展示了如何从前置任务中挖掘最近的邻居作为语义聚类的先验。此外，我们还引入了额外的约束来选择一个合适的前置任务，能够产生语义上有意义的特征表示。其次，我们将获得的先验信息整合到一个新的损失函数中，对每个图像及其最近的邻居进行分类。此外，我们还展示了如何用自标记方法减轻最近邻居选择中固有的噪声问题。我们相信这些贡献中的每一个都与无监督图像分类相关。

3.2 语义聚类表征学习

在监督学习设置中，每个样本可以通过使用可用的真实标签与正确的聚类相关联。特别是，图像 $\mathcal{D}$ 和语义类 C 之间的映射通常可以通过最小化交叉熵损失来学习。然而，当我们无法获得这样的基本事实标签时，我们需要定义一个先验，以估计哪些样本可能属于一起，哪些样本不属于一起。然而，端到端的聚类学习对网络初始化敏感。并且，训练开始时网络尚未从图像中提取高级信息，聚类容易去抓住那些低级特征（例如颜色、纹理、对比度等），这对于语义聚类来说是次优的。为了克服这些局限性，SCAN[13]使用表征学习来获得更好的语义聚类先验知识。在表征学习中，前置任务 τ 以自监督的方式学习一个由权重为 θ 的神经网络嵌入函数 Φ_θ，该神经网络将图像映射为特征表示。在语义聚类中，特征表示应该对图像变换保持不变。为此我们增强了前置任务来最小化图像 X_i 及其增强 $T[X_i]$ 之间的距离：

$$\min_{\theta} \quad d(\Phi_\theta(X_i), \Phi_\theta(T[X_i])) \tag{3}$$

表征学习中的前置任务可以获得语义上有意义的特征，因此 SCAN[13]利用前置特征作为图像聚类的先验条件。

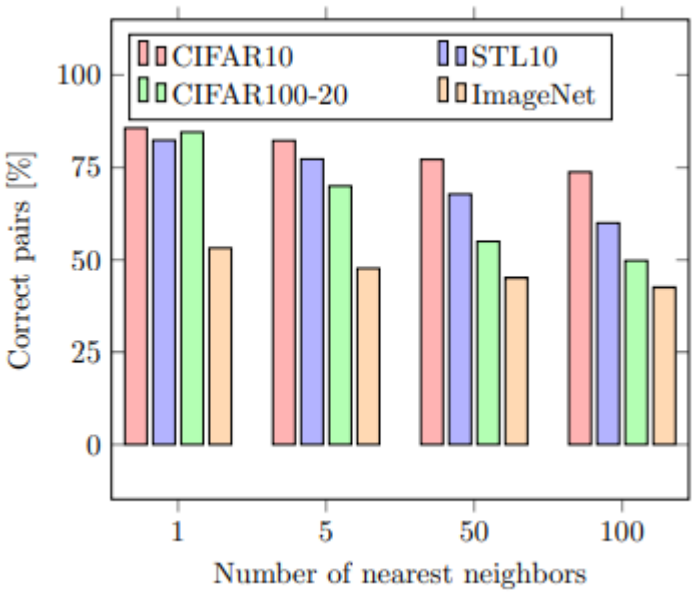


图 3: 相邻样本往往拥有相同的语义类

3.3 语义聚类损失函数

3.3.1 最近邻获取

由 3.2节已经获得语义上有意义的特征，但是，直接在获得的特征上应用 k-means 可能会导致聚类退化[19]。如图 3所示，通过表征学习，SCAN 在未标记的数据集 $\mathcal{D}$ 上训练模型 Φ_θ 来求解前置任务 τ，即实力判别函数。然后，对每个样本 $X_i \in \mathcal{D}$，在特征空间挖掘其 k 个最近邻。实验结果显示，挖掘的最近邻一般属于相同的语义实例。

3.3.2 损失函数

SCAN 基于最近邻一般属于同一语义类的特点，构造近邻样本对来指导一个由权重参数 η 参数化的神经网络，即聚类函数 Φ_η。具体是将样本 X_i 及其近邻 $\mathcal{N}_{X_i}$ 分至一类。函数 Φ_η 最后用 $softmax$ 函数对聚类结果执行软聚类，将样本 X_i 赋值给 c 类的概率表示为 $\Phi_\eta^c(X_i)$，通过最小化以下目标来学习 Φ_η 的权重：

$$\begin{aligned} \Lambda = -\frac{1}{|\mathcal{D}|}\sum_{X\in\mathcal{D}}\sum_{k\in\mathcal{N}_\mathcal{X}} log\langle \Phi_{\eta(X)}, \Phi_{\eta(k)}\rangle + \lambda\sum_{c\in C}\Phi_\eta^{'c} log\Phi_\eta^{'c}, \\ with \quad \Phi_\eta^{'c} = \frac{1}{|\mathcal{D}|}\sum_{X\in\mathcal{D}}\Phi_\eta^c(X) \end{aligned} \tag{4}$$

该目标函数第一项鼓励 Φ_η 保证样本 X_i 与其的近邻 $\mathcal{N}_{X_i}$ 预测结果一致，其中通过 Faiss 库获取样本 X_i 的最近 k 个近邻近邻 $\mathcal{N}_{X_i}$。同时为了避免 Φ_η 将所有样本分配给同一类，SCAN 加入一个熵项作为该目标函数的第二项，目的是为了鼓励预测结果均匀地分布在在所有类集合 C 上。

3.4 通过自标记进行微调

第 3.3.2节中的语义聚类损失强加了样本与其近邻之间的一致性。更具体地说，每个样本都有 K≥1 个邻居，但不可避免地，其中一些不属于同一语义集群。这些假正例导致了网络预测不准确。事实上，网络在聚类过程中形成的高度一致的预测可以被视为每个类别的“原型”。因此 SCAN 提出了一种自标记方法以利用这些“原型”，并纠正由于最近邻噪声引起的错误。在训练过程中，通过对输出概率

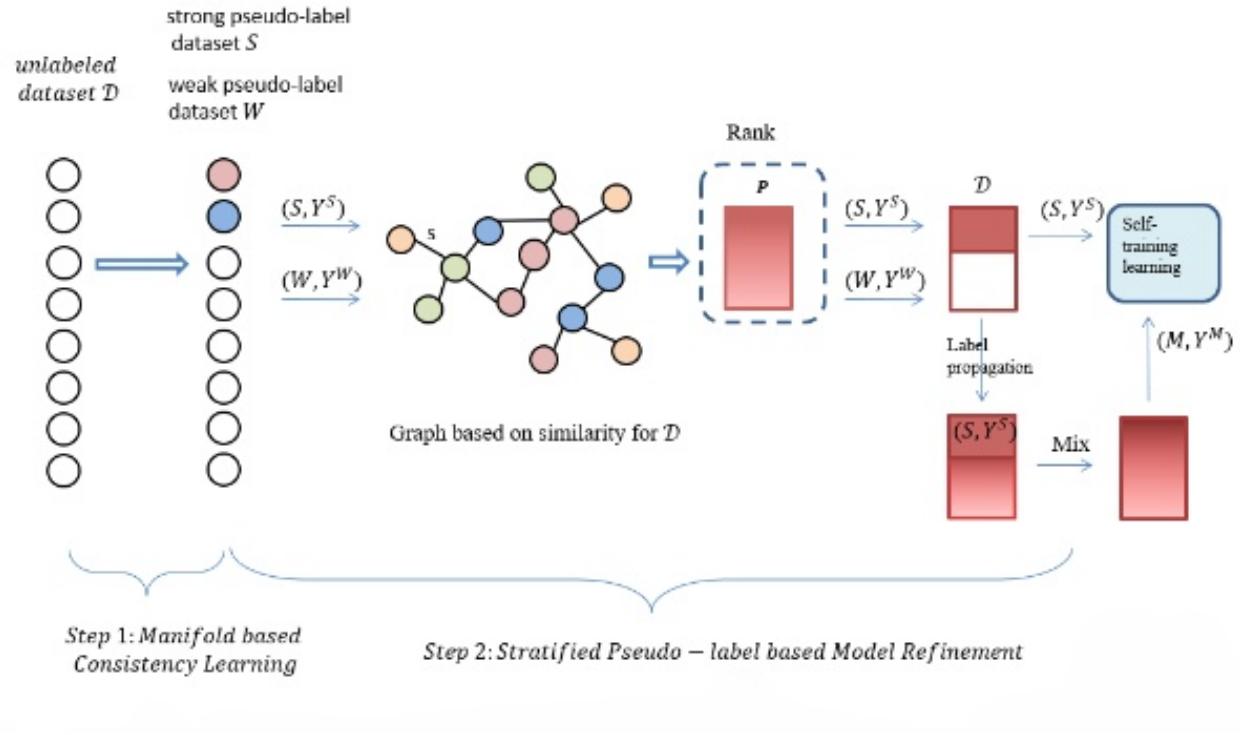


图 4: 基于分层伪标签的图像聚类框架图

进行阈值化，即 $p_{max} > threshold, p \in \Phi_\eta$ 来选择可信样本并获得其聚类后的伪标签。交叉熵损失用于更新获得的伪标签的网络权重。自标记步骤允许网络自我校正，使得预测逐渐变得更加确定。

4 复现细节

4.1 实验环境搭建

操作环境：Linux

深度学习框架：Pytorch

重要库包安装：Python 3.8.12，torch 1.12.1+cu116，torchvision 0.13.1+cu116，Faiss-gpu 1.6.4

4.2 创新点

标签传播算法是基于图的半监督学习方法，基本思路是从已标记的节点的标签信息来预测未标记的节点的标签信息，利用样本间的关系，建立完全图模型。因此，受到 SCAN 中自标记和标签传播的启发，如图 4所示，本文针对 SCAN 第二阶段阈值 τ 截断获得高置信样本初步将数据集划分为强伪标签数据集和弱伪标签数据集。接下来应用标签传播算法获得新的更加平滑的标签，设置一个采样阈值 r 筛选高置信样本来更新强、弱伪标签数据集，最后采用数据混合增强策略来提高模型泛化能力。

4.2.1 分层伪标签生成

给定数据集 $\mathcal{D}$，原 SCAN 一文中在一致性学习阶段我们能够获得数据集的预测概率 **Q**，损失函数如公式 5所示。见公式 6，引入一个流形的正则项来联合优化模型，以防止陷入局部最优（例如，大多数样本聚到一个簇中）。

$$\begin{aligned}\mathcal{L}_c &= -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=rn(\mathcal{N}^1(x_i))}\ell_{ce}(1,\mathbf{q}_i^T\mathbf{q}_j)\\ &= -\frac{1}{B}\sum_{i=1}^{B}\sum_{j=rn(\mathcal{N}^1(x_i))}\log\mathbf{q}_i^T\mathbf{q}_j\end{aligned} \tag{5}$$

$$\mathcal{L}_b = \sum_{l}^{c}\hat{\mathbf{q}}_l\log\hat{\mathbf{q}}_l \tag{6}$$

其中，$\hat{\mathbf{q}}_l = \frac{1}{B}\sum_i^B q_{il}$ 表示一个批量集合中 l 类的平均聚类指示。

由于此时的预测概率存在较大的不准确性，本文设定一个阈值 δ 来对 **Q** 进行截断，从而得到两个数据集，强伪标签数据集 $\mathcal{S}'$ 及弱伪标签数据集 $\mathcal{W}$，并满足 $\mathcal{S}'\cup\mathcal{W}=\mathcal{D}, \mathcal{S}'\cap\mathcal{W}=\emptyset$。给定一个样本 $x_i\in\mathcal{D}$，如果 $\max_l q_{il}>\delta$，则对 $\mathbf{q}_i$ 进行硬化以得到 one-hot 标签 $\mathbf{y}_i^{s'}$，并将 x_i 及 $\mathbf{y}_i^{s'}$ 分别加入 $\mathcal{S}'$ 及 $\mathcal{Y}^{s'}$ 中；否则将 x_i 及 $\mathbf{y}_i^w=\mathbf{q}_i$ 分别加入 $\mathcal{W}$ 及 $\mathcal{Y}^w$ 中。直觉来讲，强伪标签数据集 $\mathcal{S}'$ 关联的强伪标签集 $\mathcal{Y}^{s'}$ 比弱伪标签数据集 $\mathcal{W}$ 中样本关联的弱伪标签集 $\mathcal{Y}^w$ 更可靠。为了有效提升 $\mathcal{Y}^w$ 的质量并利用 $\mathcal{W}$ 来对模型进行提升，本文提出基于标签传播及数据混合的两种弱伪标签提升策略。

4.2.2 基于标签传播的弱伪标签提升

为了对生成的伪标签进行提升，本文借鉴谱聚类算法[50]提出基于标签传播的伪标签修正方法。具体地，根据语义特征空间 $f_{\mathcal{Z}}$ 中近邻节点之间特征的相似程度 s 计算一个相似性矩阵 **W**，并以此构建一个稀疏的加权图；同时，**W** 将在每次迭代完成后进行更新以更好地捕捉样本之间的相关性。**W** 的构造方法如下：

$$w_{ij}=\begin{cases}s(\mathbf{z}_i,\mathbf{z}_j), & j\in\mathcal{N}^2(x_i)\\ 0, & j\notin\mathcal{N}^2(x_i)\end{cases} \tag{7}$$

其中，$\mathcal{N}^2(x_i)$ 表示节点 x_i 的所有近邻节点集合，$s(,)$ 表示一个点积相似性函数。样本 x_i 通常可以使用 Faiss 库[51]计算得其近邻样本集合 $\mathcal{N}^2(x_i)$ 并保留 x_i 的最近的 k 个样本得到。

基于相似度矩阵 **W** 我们可以得到度矩阵 $\mathbf{D}=diag(\mathbf{W}_n)$，并进一步获得归一化矩阵 $\widetilde{\mathbf{W}}=\mathbf{D}^{-1/2}\mathbf{W}\mathbf{D}^{-1/2}$。为了提升弱伪标签集 $\mathcal{Y}^w$，根据谱学习算法[50]，我们可以使用经过筛选的 $\mathcal{Y}^s$ 并利用 $\widetilde{\mathbf{W}}$ 来对 $\mathcal{Y}^w$ 进行修正。具体的，本文提出如下目标函数来学习数据集 $\mathcal{D}$ 中样本的预测概率矩阵 **P**：

$$\begin{aligned}\mathcal{J}(\mathbf{P}) = &\frac{1}{2}\sum_{i,j=1}^{n}\widetilde{w}_{ij}\|\frac{\mathbf{p}_i}{\sqrt{d_{ii}}}-\frac{\mathbf{p}_j}{\sqrt{d_{jj}}}\|^2\\ &+\frac{\mu}{2}\sum_{i=1}^{n}\|\mathbf{p}_i-\mathbf{Y}_i\|^2\end{aligned} \tag{8}$$

对公式 8获得的的预测概率矩阵 P 进行排序，每个类别预测概率靠前的 r 个样本加入 $(\mathcal{S}'',\mathcal{Y}'')$ 并求交集，有如下公式：

$$(\mathcal{S},\mathcal{Y})=(\mathbf{S}',\mathcal{Y}')\cap(\mathcal{S}'',\mathcal{Y}'') \tag{9}$$

其中第一项根据相似度矩阵 **W** 强制近邻节点之间共享相似的标签，而第二项鼓励学到的预测概

率 **P** 逼近上一步得到的伪标签 **Y**。μ 是平衡两项的系数。为了加速计算，本文将以上目标函数的优化转化为使用共轭梯度方法来求解线性方程[50]($\mathbf{I}-\gamma\mathbf{W}$)$\mathbf{P}=\mathbf{Y}$，其中 $\gamma=\dfrac{1}{1+\mu}$。最后，对于任意的 $x_i\in\mathcal{S}$，本文根据 **P** 来更新其伪标签 y_i^w 为

$$y_i^w=\text{one-hot}(\underset{j}{\text{argmax}}p_{ij}) \tag{10}$$

其中 one-hot(j) 将得到第 j 个位置为 1 的硬标签。

4.2.3 基于分层混合的弱伪标签提升

受半监督学习启发，本文将 $\mathcal{S}$ 近似看成有标签数据集，将 $\mathcal{W}$ 视为无标签数据集，并使用 Mixup 算法[52]对强、弱伪标签数据进行混合，再应用半监督目标损失对分类模型进行精炼。使用 Mixup 数据增强方法可以根据 $\mathcal{S}$ 及 $\mathcal{W}$ 获得线性组合后的新样本来提高模型的鲁棒性。本文对 $\mathcal{S}$ 中的每个样本进行弱增强得到 $\hat{\mathcal{S}}$，同时对 $\mathcal{W}$ 中的每个样本进行强增强得到 $\hat{\mathcal{W}}$。随后，将 $\hat{\mathcal{S}}$ 和 $\hat{\mathcal{W}}$ 组合后生成批数据作为分层混合的数据源。

分层混合的具体操作如下，对任意的 $\hat{u}_i\in\hat{\mathcal{W}}$，从 $\hat{\mathcal{S}}$ 中取一个样本 $\hat{x}_i\in\hat{\mathcal{S}}$：

$$\begin{aligned}x_i'&=\lambda'\hat{x}_i+(1-\lambda')\hat{u}_i\\ y_i'&=\lambda'\hat{y}_i^s+(1-\lambda')\hat{y}_i^w\end{aligned} \tag{11}$$

其中，λ' 是两个样本混合程度的平衡系数，计算如下:

$$\lambda\sim Beta(\alpha,\alpha) \tag{12}$$

其中，$Beta$ 表示一个连续概率分布函数，α 为该分布的系数。

$$\lambda'=\max(\lambda,1-\lambda) \tag{13}$$

这里对 λ 平衡参数进行调整，使带有强伪标签的数据获得较大的混合比重，以更好地监督模型的训练。

经过上述分层混合操作之后，我们可以得到新的混合数据集 $\mathcal{M}$ 及混合标签集 $\mathcal{Y}^m$。利用这种混合数据来对模型进行更新将有利于提升模型的泛化能力。

4.2.4 基于分层伪标签的模型精炼

给定强伪标签数据集及其伪标签集 $(\mathcal{S},\mathcal{Y}^s)$，以及混合数据集及其混合标签集 $(\mathcal{M},\mathcal{Y}^m)$，本文的下一步目标是通过自训练学习来精炼聚类模型 f_θ，以进一步提升模型的性能。

在每个 mini-batch 操作中，本文将针对 $(\mathcal{S},\mathcal{Y}^s)$ 设计如下的损失函数来进行学习：

$$\mathcal{L}_{\mathcal{S}}=\frac{1}{B}\sum_{(x',')\in(\mathcal{S},\mathcal{Y}^s)}\ell_{ce}\left(y',f_{\mathcal{Q}}(x';\theta,\phi)\right) \tag{14}$$

针对 $(\mathcal{M},\mathcal{Y}^m)$ 设计如下的损失函数来进行学习：

$$\mathcal{L}_{\mathcal{M}} = \frac{1}{B} \sum_{(x',')\in(\mathcal{M},\mathcal{Y}^m)} \ell_{ce}\left(y', f_{\mathcal{Q}}(x';\theta,\phi)\right) \tag{15}$$

最终在每个 mini-batch 操作中的精炼损失函数为：

$$\mathcal{L}_p = \mathcal{L}_{\mathcal{S}} + \lambda_{\mathcal{M}}\mathcal{L}_{\mathcal{M}} \tag{16}$$

其中，$\ell_{ce}(,)$ 表示交叉熵损失函数，$\lambda_{\mathcal{M}}$ 是平衡混合伪标签损失部分的权重系数。

4.3 与已有开源代码对比

与 SCAN[13]相比，不同之处在于第三阶段模型精炼部分加入了标签传播的方式来提升强弱伪标签数据的质量，又使用了数据混合的方式进一步提升模型的泛化能力。以下主要是基础的数据加载、标签传播、数据混合训练的代码。

4.3.1 数据类准备

```
# 第一阶段后特征提取用于标签传播全连接图的构建
def extract_features_simp(train_loader, model, args, forward_pass='backbone'):
    model.eval()
    embeddings_all = []
    target_all = []

    with torch.no_grad():
        for i, (batch_input) in enumerate(train_loader):
            X_n = batch_input['image'].cuda()
            Y_n = batch_input['target']
            if forward_pass == 'default':
                feats = model(X_n)
                feats = feats[0]

            elif forward_pass == 'backbone':
                feats = model(X_n, forward_pass=forward_pass)
            embeddings_all.append(feats.data.cpu())
            target_all.append(Y_n)
    embeddings_all = np.asarray(torch.cat(embeddings_all).numpy())
    target_all = np.asarray(torch.cat(target_all).numpy())

    if forward_pass == 'backbone':
        return embeddings_all, target_all
    return embeddings_all

# 重构数据加载器loader, 按照
class StreamBatchSampler(Sampler):
    def __init__(self, primary_indices, batch_size):
        self.primary_indices = primary_indices
        self.primary_batch_size = batch_size

    def __iter__(self):
        # 将indices转成迭代的元素
        primary_iter = iterate_eternally(self.primary_indices)
        # 构造batch
        return (primary_batch for (primary_batch)
                in grouper(primary_iter, self.primary_batch_size)
                )

    def __len__(self):
```

```
        return len(self.primary_indices) // self.primary_batch_size
def iterate_eternally(indices):
    def infinite_shuffles():
        while True:
            yield np.random.permutation(indices)
    return itertools.chain.from_iterable(infinite_shuffles())

def grouper(iterable, n):
    "Collect data into fixed-length chunks or blocks"
    # grouper('ABCDEFG', 3) --> "ABC DEF"
    args = [iter(iterable)] * n
    return zip(*args)

def create_data_loaders_simple(args, p, train_dataset, labels, l_indices,
    indices, data=None, checkpoint=False):
    if checkpoint:
        dataset = data
    else:
        dataset = DBSS(args, train_dataset, labels, l_indices)

    sampler = SubsetRandomSampler(dataset.labeled_idx)
    batch_sampler = BatchSampler(sampler, args.batch_size, drop_last=True)
    train_loader = DataLoader(dataset, batch_sampler=batch_sampler,
        num_workers=p['num_workers'], pin_memory=True) # 只有有标签的部分
    train_loader_noshuff = DataLoader(dataset, batch_size=args.batch_size,
        shuffle=False,num_workers=p['num_workers'],
        pin_memory=True,drop_last=False)

    batch_sampler_l = StreamBatchSampler(dataset.labeled_idx,
        batch_size=args.labeled_batch_size)
    batch_sampler_u = BatchSampler(SubsetRandomSampler(dataset.unlabeled_idx),
        batch_size=args.batch_size-args.labeled_batch_size, drop_last=True)

    train_loader_l = DataLoader(dataset, batch_sampler=batch_sampler_l,
        num_workers=p['num_workers'], pin_memory=True)
    train_loader_u = DataLoader(dataset, batch_sampler=batch_sampler_u,
        num_workers=p['num_workers'], pin_memory=True)


    return train_loader, train_loader_noshuff, train_loader_l,train_loader_u,
        dataset
```

4.3.2 标签传播

```
class DataFolder(Dataset):
    def __init__(self, dataset):
        super(DataFolder, self).__init__()
        transform = dataset.transform
        dataset.transform = None
        self.dataset = dataset

        if isinstance(transform, dict):
            self.e_transform = transform['standard']
            self.w_transform = transform['weak']
            self.s_transform = transform['augment']
        else:
            self.e_transform = transform
            self.w_transform = transform
```

```
15          self.s_transform = transform
16
17      def __len__(self):
18          return len(self.dataset)
19
20      def __getitem__(self, index):
21          x = self.dataset.__getitem__(index)
22          sample = x['image']
23
24          ### If unlabeled grab pseudo-label
25          labeled_case = self.is_labeled[index]
26          if labeled_case == 0:
27              target = self.p_labels[index] # 无标签得到的是概率指示
28          else:
29              target = self.p_labels[index] # 有标签得到的是自己的标签
30
31              ### If in feat mode just give base images
32          if self.feat_mode == True:
33              aug_images = []
34              e_sample = self.e_transform(sample)
35              aug_images.append(e_sample)
36              return aug_images
37
38          else:
39              if labeled_case == 1: # 强伪标签数据使用弱增强
40                  aug_images = []
41                  for i in range(self.aug_num):
42                      t_sample = self.w_transform(sample)
43                      aug_images.append(t_sample)
44              else:
45                  aug_images = []
46                  for i in range(self.aug_num): # 弱伪标签数据使用强增强
47                      t_sample = self.s_transform(sample)
48                      aug_images.append(t_sample)
49
50              return aug_images, target
51
52  class DBSS(DataFolder):
53      def __init__(self, args, dataset, labels, l_indices, load_im=False):
54          super(DBSS, self).__init__(dataset)
55          self.labeled_idx = l_indices.numpy()
56          self.unlabeled_idx = np.setdiff1d(np.arange(len(self.dataset)),
                self.labeled_idx)
57          self.targets = np.asarray(self.dataset.targets)
58          self.all_labels = []
59          self.feat_mode = True
60          self.probs_iter = np.zeros((len(self.dataset),
                len(self.dataset.classes)))
61          self.p_labels = []
62          self.label_dis = 1/(len(self.dataset.classes)) *
                np.ones(len(self.dataset.classes))
63          self.k = args.topk_lp
64          self.relabel_dataset(labels)
65
66      def relabel_dataset(self, labels):
67          self.all_labels = copy.copy(np.asarray(self.dataset.targets))
68          self.all_labels[self.labeled_idx] = labels.numpy() # 记录真实标签
69          self.p_labels = np.ones_like(self.all_labels) * - 1 #
                返回一个用-1填充的跟输入形状和类型一致的数组
70          self.is_labeled = np.ones(len(self.dataset))
71          self.is_labeled[self.unlabeled_idx] = 0 # 无标签位置标签设置为0，反之
```

```
72
73
74      def one_iter_true(self, X, logit, D, I, max_iter, epoch, num=0.005):
75          alpha = 0.99
76          labeled_idx = self.labeled_idx
77          unlabeled_idx = self.unlabeled_idx
78          k = self.k
79
80          # 标签传播结果验证省略。。。
81
82          N = D.shape[0]
83
84          # Create the graph--Wij, 不平衡矩阵稀疏化处理
85          row_idx = np.arange(N)
86          D = D[:, 1:] # 排除自己
87          I = I[:, 1:]
88          row_idx_rep = np.tile(row_idx, (k, 1)).T # 只沿着X轴复制的方法,
                k+maxLen-1倍的关系
89          indices = row_idx_rep.flatten('F')
90          indptr = I.flatten('F')
91          # 用csr_matrix构建稀疏矩阵, 进而减少计算量
92          W = scipy.sparse.csr_matrix((D.flatten('F'), (indices, indptr)),
                shape=(N, N))
93
94          # W = torch.tensor(D)
95          W = W + W.T.multiply(W.T > W) - W.multiply(W.T > W)
96
97          # Normalize the graph
98          W = W - scipy.sparse.diags(W.diagonal())
99          S = W.sum(axis=1)
100         S[S == 0] = 1
101         D = np.array(1. / np.sqrt(S))
102         D = scipy.sparse.diags(D.reshape(-1)) # 得到度矩阵
103         Wn = D * W * D # 得到归一化矩阵
104
105         # Initiliaze the y vector for each class
106         labels = self.all_labels
107         Z = np.zeros((N, len(self.dataset.classes)))
108         A = scipy.sparse.eye(Wn.shape[0]) - alpha * Wn # 对角线上的稀疏矩阵
109         # 返回一个稀疏 (m x n) 矩阵, 其中第 k 个对角线全为 1, 其他全为 0。
110         Y = np.zeros((N, len(self.dataset.classes)))
111         Y[labeled_idx, labels[labeled_idx]] = 1
112
113         print("标签传播开始")
114         for i in range(len(self.dataset.classes)):
115             f, _ = scipy.sparse.linalg.cg(A, Y[:, i], tol=1e-3, maxiter=max_iter)
116             Z[:, i] = f
117         Z[Z < 0] = 0 # 预测矩阵n*c
118         print("标签传播结束")
119
120
121         # Rink
122         probs_iter = F.normalize(torch.tensor(Z), 1).numpy() #
                按行操作将某一个维度除以那个维度对应的2范数
123
124         # 直接按照类别, 对每一个标签传播预测概率进行排序
125         P_s, idx = torch.sort(torch.from_numpy(probs_iter), dim=0,
                descending=True) # 按照第一维(每个预测类别内)进行排序
126         idx_groupi_n = []
127         for i in range(len(self.dataset.classes)):
128             num_labels = int(num * len(P_s[:, i]))
```

```
            idx_groupi_n.append(idx[0:num_labels, i]) # 内部取num
        P_labeled_idx = torch.cat(idx_groupi_n, dim=0) #
            连接成完整的new_labeled_idx

        if epoch == 0:
            _labeled_idx = np.setdiff1d(labeled_idx, P_labeled_idx) # labeled_idx
                去掉相交的部分
            new_labeled_idx = np.setdiff1d(labeled_idx, _labeled_idx) #
                拿到与labeled_idx相交的部分
            labeled_idx = new_labeled_idx
            # 更新强标签和弱标签部分
            self.labeled_idx = labeled_idx
            self.unlabeled_idx = np.setdiff1d(np.arange(len(self.dataset)),
                self.labeled_idx)
            unlabeled_idx = self.unlabeled_idx

        # probs_iter = F.normalize(torch.tensor(Z), 1).numpy()
        probs_iter[labeled_idx] = np.zeros(len(self.dataset.classes))
        probs_iter[labeled_idx, labels[labeled_idx]] = 1 # 将标签设置为1, 重置
        self.probs_iter = probs_iter
        # 初始化有标签的标签
        p_labels = np.argmax(probs_iter, 1) #argmax: 按行方向搜索最大值, 并返回索引;
            顺序无变化
        self.p_labels = p_labels # 写回类别属性
```

4.3.3 强弱伪标签数据集混合训练

```
# mixup数据混合
def mixup_data(x_1, index, lam):
    mixed_x_1 = lam * x_1 + (1 - lam) * x_1[index, :]
    return mixed_x_1

# 混合损失
def mixup_criterion(pred, y_a, y_b, lam):
    criterion = nn.CrossEntropyLoss(reduction='none').cuda()
    return lam * criterion(pred, y_a) + (1 - lam) * criterion(pred, y_b)

# 混合训练
def train_semi(p, train_loader_l, train_loader_u, model, optimizer, epoch,
    global_step, args, ema_model=None):
    model.train()
    lr_length = len(train_loader_u)
    train_loader_l = iter(train_loader_l)

    if args.progress == True:
        tk0 = tqdm(train_loader_u, desc="Semi Supervised Learning Epoch " +
            str(epoch) + "/" + str(args.epochs),
                unit="batch", ncols=80)
        loss_meter = meter.AverageValueMeter()
    else:
        tk0 = train_loader_u

    # Training
    for i, (aug_images_u, target_u) in enumerate(tk0):
        aug_images_l, target_l = next(train_loader_l)
        target_l = target_l.to(args.device)
        target_u = target_u.to(args.device)
        target = torch.cat((target_l, target_u), 0)
```

```
        alpha = args.alpha
        index = torch.randperm(args.batch_size, device=args.device)
        lam = np.random.beta(alpha, alpha) # 生成参数为alpha的贝塔分布
        target_a, target_b = target, target[index]
        optimizer.zero_grad()
        adjust_learning_rate(optimizer, epoch, i, lr_length, args)
        count = 0
        for batch_l, batch_u in zip(aug_images_l, aug_images_u):
            batch_l = batch_l.to(args.device) # shape(32,3,96,96)
            batch_u = batch_u.to(args.device) # shape(16,3,96,96)
            batch = torch.cat((batch_l, batch_u), 0)
            m_batch = mixup_data(batch, index, lam)
            class_logit = model(m_batch)
            class_logit = class_logit[0]

            if count == 0:
                loss_sum = mixup_criterion(class_logit.double(), target_a,
                    target_b, lam).mean()
            else:
                loss_sum = loss_sum + mixup_criterion(class_logit.double(),
                    target_a, target_b, lam).mean()
            count += 1

        loss = loss_sum / (args.aug_num)
        loss.backward()
        optimizer.step()
        if args.progress == True:
            loss_meter.add(loss.item())
            tk0.set_postfix(loss=loss_meter.mean)
        global_step += 1
    print('loss:\t{:.4f}'.format(loss_meter.mean))
    return global_step, loss_meter.mean
```

5 实验结果分析

5.1 实验设置

5.1.1 实验数据集

基准数据集信息如表 1所示，在后续的实验中均使用原始的图像尺寸。

表 1: 基准数据集主要特征。

数据集名称	图像大小	# 训练集	# 测试集	# 类别数量
STL10	96×96	$5,000$	$8,000$	10

5.1.2 实现细节

为了公平起见，提出的方法中，图像的预训练特征学习实验设置和 SCAN 方法保持一致，在 STL10 数据集上采用 SimCLR[26]方法作为实例判别任务，使用 ResNet-18[53]作为主干网络，并在第一阶段和第二阶段学习同一个分类模型 Φ。在训练阶段中，STL10 数据集训练所需的 Epoch 次数均设置为 100，批量大小设置为 128，近邻数量为 20，正则项权重系数设置为 5，采用 Adam 优化器进行迭代优化，学习率和权重衰减分别设置为 0.0001 和 0.0001。在模型精炼阶段，对于 STL10 数据集，Epoch 次数均设

置为 50。强伪标签数据集和弱伪标签数据集的批量大小设置为 256。置信度阈值设置为 0.95，标签传播中近邻数量设置为 20，采样阈值 r 设置为 0.03。Beta 分布的系数设置为 1。采用 SGD 优化器进行迭代优化，学习率为 0.03，权重衰减为 0.0005，nesterov 动量为 0.9。两个阶段均使用 Faiss[51]库来搜索最近邻样本。

5.1.3 验证

基于聚类准确性（ACC），归一化互信息（NMI）[54]和调整后的 rand 指数（ARI）[55]进行评估。训练和测试数据集中的图像分别进行训练和模型测试，即在训练阶段屏蔽测试集图像。最后，实验结果为 3 次不同运行结果的平均值和标准差，且在训练过程中所有数据集均使用相同的主干网络、增强方法和前置任务。

5.2 对比试验

表 2展示了本文提出的方法在训练数据集与测试数据集分隔（训练数据集只用于训练，测试数据集只用于测试）的情况下，和 SCAN[13]的方法在 STL10 数据集上的聚类结果。具体地说，本文提出的方法 SPC 的结果相较方法 SCAN[13]的精炼阶段提升了 $+2.7\%$。

表 2: STL10 基准数据集的最优结果比较。

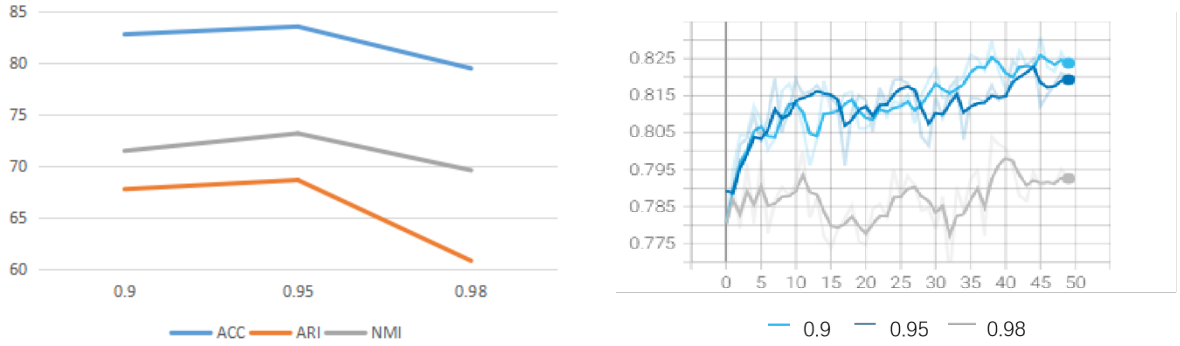
Dataset	**STL10**		
Setup	ACC	NMI	ARI
SCAN† (Best)[13]	80.9	69.8	64.6
SPC† (Best)	83.6	68.7	73.2

5.3 消融实验

在本节中，验证各个损失项对聚类性能的影响，在 STL10 数据集上进行了消融实验。具体来说，为了能够体现每个损失项带来的性能增益，以损失项构造的退化模型作为基线，然后引入进行正则约束，最后在此基础上加入验证伪标签监督学习的作用。如表 3所示，$\mathcal{L}_b$ 的引入能够控制样本均匀分布在所有簇中，在 STL10 数据集上模型性能比基线提高了 46.8%(ACC)，24.3%(NMI)，38.8%(ARI)，这表明有效防止了模型陷入平凡解。这表明有效避免了退化解的出现。最后，引入 $\mathcal{L}_p$ 带来了 5.7%(ACC)、1.6%(NMI) 和 11.6%(ARI) 的提升，这个结果说明了分层伪标签监督学习的有效性，增加的高质量伪标签数量提升了模型性能。

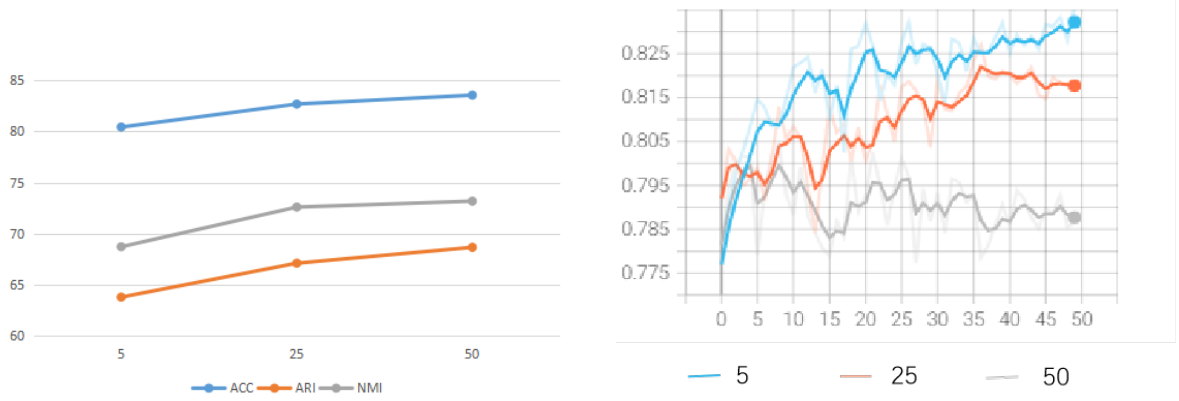
表 3: SPC 方法在 STL10 数据集上的消融分析。

Dataset	**STL10**		
Setup	ACC	NMI	ARI
$\mathcal{L}_c$	31.1	42.8	22.8
$\mathcal{L}_c + \mathcal{L}_b$	77.9	67.1	61.6
$\mathcal{L}_c + \mathcal{L}_b + \mathcal{L}_p$	83.6	68.7	73.2



((a)) 评估值变化曲线　　((b)) 训练过程准确度变化曲线

图 5: STL10 数据集精炼阶段中阈值 δ 参数敏感性分析



((a)) 评估值变化曲线　　((b)) 训练过程准确度变化曲线

图 6: STL10 数据集精炼阶段中阈值 k_2 参数敏感性分析

5.4　参数敏感性分析

5.4.1　置信度阈值 δ

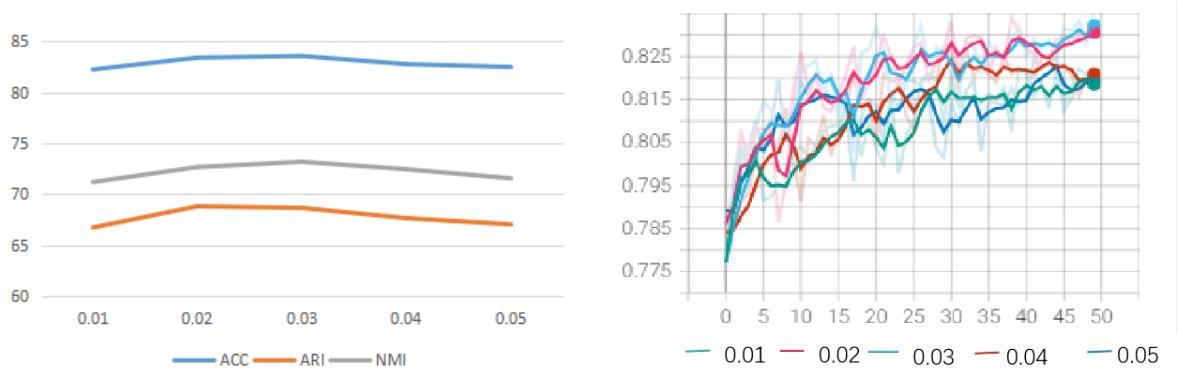
阈值 δ 控制了初步筛选出的伪标签的置信度，所以设定合适的阈值有助于获得足够置信的伪标签。以 STL10 数据集为例，通过设置不同的参数值验证对伪标签的影响，其中 $\delta \in 0.9, 0.95, 0.98$。从图 5(a)中不同阈值对应的聚类结果可以得出，当时整体呈现波动式增长，当 $\delta = 0.95$ 时获得最佳性能。但是当 $\delta = 0.98$ 时性能下降，其原因是置信度阈值过高导致了伪标签数量大幅减少，从而致使后续伪标签监督学习的过程缺乏足够的样本，进而影响了聚类结果。

5.4.2　近邻数量 k_2

近邻数量 k_2 相似度矩阵的构建有直接影响，从而影响已标记的强伪标签节点预测未标记节点的标签。因此，设置 $k_2 \in 5, 20, 50$ 来观察模型对不同 k_2 值的敏感性。结果如图 6(a)所示，且当 $k_2 = 20$ 时取得最优聚类性能。

5.4.3　采样阈值 r

采样阈值 r 作用在第一次标签传播后，目的是筛选阈值截断获得的强伪标签数据集。因此，通过设置 $r \in 0.01, 0.02, 0.03, 0.04, 0.05$ 来观察模型对 r 的敏感性。如图 7(a)所示，随着采样阈值 r 的增大，强伪标签数据集的采样范围也在不断增大。当 $r \geqslant 0.03$，强伪标签中的噪声样本增加；当 $r \leqslant 0.03$ 时，高置信样本过少，从而使得聚类性能降低。



((a)) 评估值变化曲线　　((b)) 训练过程准确度变化曲线

图 7: STL10 数据集精炼阶段中阈值 r 参数敏感性分析

6　总结与展望

本文介绍了复现的文章 SCAN[13]的基本思想，即提倡将特征学习和聚类解耦，基于最近邻一般属于一类的情况来帮助语义聚类，最后通过阈值截断自标记来提升模型性能。但针对阈值截断获取到置信样本不足的问题，本文的创新点是先阈值截断获取强弱伪标签数据集，并利用标签传播的特性来提升弱标签质量，最后将强弱伪标签数据集进行混合训练来提升模型泛化能力。通过实验验证，相较于 SCAN 在 STL10 数据集上提升了 +2.7%。

目前实现过程中仍存在不足，比如只对一个数据集进行了实验。又实验结果显示，简单的标签传播概率预测矩阵排序相对于原始的阈值截断的方法提升不明显，仍有改进的地方。在未来的工作中，将进一步研究结构化信息，标签传播过程中如何调节不同质量样本的传播能量，以及利用近接图中边的权重关系进行数据增强进而提升模型泛化能力。

参考文献

[1] XIE J, GIRSHICK R, FARHADI A. Unsupervised Deep Embedding for Clustering Analysis[C]//Proceedings of ICML 2016. 2016: 478-487.

[2] YANG B, FU X, SIDIROPOULOS N D, et al. Towards K-means-friendly Spaces: Simultaneous Deep Learning and Clustering[C]//Proceedings of ICML 2017: vol. 70. 2017: 3861-3870.

[3] TIAN K, ZHOU S, GUAN J. DeepCluster: A General Clustering Framework Based on Deep Learning [C]//Proceedings of ECML PKDD 2017. 2017: 809-825.

[4] AGGARWAL C C, WOLF J L, YU P S, et al. Fast Algorithms for Projected Clustering[C]//Proceedings of ACM SIGMOD 1999. 1999: 61-72.

[5] YANG J, PARIKH D, BATRA D. Joint Unsupervised Learning of Deep Representations and Image Clusters[C]//Proceedings of CVPR 2016. 2016: 5147-5156.

[6] HE K, FAN H, WU Y, et al. Momentum Contrast for Unsupervised Visual Representation Learning[C]//Proceedings of CVPR 2020. 2020: 9726-9735.

[7] ZHONG H, WU J, CHEN C, et al. Graph Contrastive Clustering[C]//Proceedings of ICCV 2021. 2021: 9224-9233.

[8] WU J, LONG K, WANG F, et al. Deep Comprehensive Correlation Mining for Image Clustering[C]//Proceedings of ICCV 2019. 2019: 8149-8158.

[9] DANG Z, DENG C, YANG X, et al. Nearest Neighbor Matching for Deep Clustering[C]//Proceedings of CVPR 2021. 2021: 13693-13702.

[10] ZHANG R, ISOLA P. Colorful Image Colorization[C]//Proceedings of ECCV 2016. 2016: 649-666.

[11] GIDARIS S, SINGH P, KOMODAKIS N. Unsupervised Representation Learning by Predicting Image Rotations[C]//ICLR 2018. 2018: 1-16.

[12] PATHAK D, KRAHENBUHL P, DONAHUE J, et al. Context encoders: Feature learning by inpainting [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 2536-2544.

[13] VAN GANSBEKE W, VANDENHENDE S, GEORGOULIS S, et al. SCAN: Learning to Classify Images Without Labels[C]//Proceedings of ECCV 2020. 2020: 268-285.

[14] JI P, ZHANG T, LI H, et al. Deep Subspace Clustering Networks Pan[C]//Proceedings of NeurlIPS 2017. 2017: 23-32.

[15] YAMAGUCHI M, IRIE G, KAWANISHI T, et al. Subspace structure-aware spectral clustering for robust subspace clustering[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 9875-9884.

[16] LAW M T, URTASUN R, ZEMEL R S. Deep spectral clustering learning[C]//International conference on machine learning. 2017: 1985-1994.

[17] SHAHAM U, STANTON K. SpectralNet: Spectral Clustering using Deep Neural Networks[C]//ICLR 2018. 2018: 1-20.

[18] CHANG J, WANG L, MENG G, et al. Deep Adaptive Image Clustering[C]//Proceedings of ICCV 2017. 2017: 5880-5888.

[19] CARON M, BOJANOWSKI P, JOULIN A, et al. Deep clustering for unsupervised learning of visual features[C]//Proceedings of ECCV 2018. 2018: 132-149.

[20] CARON M, BOJANOWSKI P, MAIRAL J, et al. Unsupervised pre-training of image features on non-curated data[C]//Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019: 2959-2968.

[21] ASANO Y M, RUPPRECHT C, VEDALDI A. Self-labelling via simultaneous clustering and representation learning[J]. arXiv preprint arXiv:1911.05371, 2019.

[22] HAEUSSER P, PLAPP J, GOLKOV V, et al. Associative deep clustering: Training a classification network with no labels[C]//Pattern Recognition: 40th German Conference, GCPR 2018, Stuttgart, Germany, October 9-12, 2018, Proceedings 40. 2019: 18-32.

[23] YANG J, PARIKH D, BATRA D. Joint unsupervised learning of deep representations and image clusters [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 5147-5156.

[24] JI X, VEDALDI A, HENRIQUES J. Invariant Information Clustering for Unsupervised Image Classification and Segmentation[C]//Proceedings of ICCV 2019. 2019: 9864-9873.

[25] LI Y, HU P, LIU Z, et al. Contrastive clustering[C]//Proceedings of AAAI 2021: vol. 35: 10. 2021: 8547-8555.

[26] CHEN T, KORNBLITH S, NOROUZI M, et al. A simple framework for contrastive learning of visual representations[C]//Proceedings of ICML 2020. 2020: 1575-1585.

[27] MAHON L, LUKASIEWICZ T. //German Conference on Artificial Intelligence (Künstliche Intelligenz). 2021: 158-178.

[28] LOWE D G. Object recognition from local scale-invariant features[C]//Proceedings of ICCV 1999: vol. 2. 1999: 1150-1157.

[29] DALAL N, TRIGGS B. Histograms of oriented gradients for human detection[C]//Proceedings of CVPR 2005: vol. 1. 2005: 886-893.

[30] VINCENT P, LAROCHELLE H, BENGIO Y, et al. Extracting and Composing Robust Features with Denoising Autoencoders[C]//Proceedings of ICML 2008. 2008: 1096-1103.

[31] DONAHUE J, SIMONYAN K. Large scale adversarial representation learning[C]//Proceedings of NeurIPS 2019. 2019: 1096-1103.

[32] NOROOZI M, PIRSIAVASH H, FAVARO P. Representation Learning by Learning to Count[C]//Proceedings of ICCV 2017. 2017: 5898-5906.

[33] BACHMAN P, HJELM R D, BUCHWALTER W. Learning representations by maximizing mutual information across views[C]//Proceedings of NeurIPS 2019: vol. 32. 2019.

[34] HADSELL R, CHOPRA S, LECUN Y. Dimensionality Reduction by Learning an Invariant Mapping [C]//Proceedings of CVPR 2006: vol. 2. 2006: 1735-1742.

[35] GRILL J B, STRUB F, ALTCHÉ F, et al. Bootstrap Your Own Latent A New Approach to Self-Supervised Learning[C]//Proceedings of NeurIPS 2020. 2020: 1-14.

[36] LI J, ZHOU P, XIONG C, et al. Prototypical contrastive learning of unsupervised representations[J]. arXiv preprint arXiv:2005.04966, 2020.

[37] GUO Y, XU M, LI J, et al. HCSC: Hierarchical Contrastive Selective Coding[C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2022: 9706-9715.

[38] SAJJADI M, JAVANMARDI M, TASDIZEN T. Regularization with stochastic transformations and perturbations for deep semi-supervised learning[J]. Advances in neural information processing systems, 2016, 29.

[39] OORD A V D, LI Y, VINYALS O. Representation learning with contrastive predictive coding[J]. arXiv preprint arXiv:1807.03748, 2018.

[40] CHAPELLE O, SCHOLKOPF B, ZIEN A. Semi-supervised learning (chapelle, o. et al., eds.; 2006)[book reviews][J]. IEEE Transactions on Neural Networks, 2009, 20(3): 542-542.

[41] SOHN K, BERTHELOT D, CARLINI N, et al. Fixmatch: Simplifying semi-supervised learning with consistency and confidence[J]. Advances in neural information processing systems, 2020, 33: 596-608.

[42] HU Z, YANG Z, HU X, et al. Simple: similar pseudo label exploitation for semi-supervised classification [C]//Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021: 15099-15108.

[43] ARAZO E, ORTEGO D, ALBERT P, et al. Pseudo-labeling and confirmation bias in deep semi-supervised learning[C]//2020 International Joint Conference on Neural Networks (IJCNN). 2020: 1-8.

[44] TARVAINEN A, VALPOLA H. Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results[J]. Advances in neural information processing systems, 2017, 30.

[45] LEE D H, et al. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks[C]//Workshop on challenges in representation learning, ICML: vol. 3: 2. 2013: 896.

[46] BERTHELOT D, CARLINI N, GOODFELLOW I, et al. Mixmatch: A holistic approach to semi-supervised learning[J]. Advances in neural information processing systems, 2019, 32.

[47] BERTHELOT D, CARLINI N, CUBUK E D, et al. Remixmatch: Semi-supervised learning with distribution alignment and augmentation anchoring[J]. arXiv preprint arXiv:1911.09785, 2019.

[48] SELLARS P, AVILES-RIVERO A I, SCHÖNLIEB C B. Laplacenet: A hybrid energy-neural model for deep semi-supervised classification[J]. arXiv preprint arXiv:2106.04527, 2021.

[49] ZHANG B, WANG Y, HOU W, et al. Flexmatch: Boosting semi-supervised learning with curriculum pseudo labeling[J]. Advances in Neural Information Processing Systems, 2021, 34: 18408-18419.

[50] ZHOU D, BOUSQUET O, LAL T, et al. Learning with local and global consistency[J]. Advances in neural information processing systems, 2003, 16.

[51] JOHNSON J, DOUZE M, JÉGOU H. Billion-Scale Similarity Search with GPUs[J]. IEEE Transactions on Big Data, 2021, 7(3): 535-547.

[52] ZHANG H, CISSE M, DAUPHIN Y N, et al. mixup: Beyond empirical risk minimization[J]. arXiv preprint arXiv:1710.09412, 2017.

[53] WANG F, JIANG M, QIAN C, et al. Residual attention network for image classification[C]// Proceedings of the IEEE conference on computer vision and pattern recognition. 2017: 3156-3164.

[54] MCDAID A F, GREENE D, HURLEY N. Normalized mutual information to evaluate overlapping community finding algorithms[J]. arXiv preprint arXiv:1110.2515, 2011.

[55] HUBERT L, ARABIE P. Comparing partitions[J]. Journal of Classification, 1985, 2(1): 193-218.